

**Lista pytań i zadań określających zakres materiału z Javy SE,  
przewidzianego na egzamin z przedmiotu**

**Języki i metody programowania – Java**

**Na podstawie „Tutorial Java SE”**

**<https://docs.oracle.com/javase/tutorial/index.html>**

**(<https://docs.oracle.com/javase/tutorial/java/index.html>,**

**<https://docs.oracle.com/javase/tutorial/essential/index.html>**

**<https://docs.oracle.com/javase/tutorial/collections/index.html>**

**<https://docs.oracle.com/javase/tutorial/uiswing/index.html>)**

**Wkrótce podam przykładowy zestaw pytań i zadań do rozwiązania i  
opracowania podczas egzaminu z podanego przedmiotu, oparty na tym  
materiale**

## Questions and Exercises: Object-Oriented Programming Concepts

### Questions

1. Real-world objects contain \_\_\_ and \_\_\_.
2. A software object's state is stored in \_\_\_.
3. A software object's behavior is exposed through \_\_\_.
4. Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data \_\_\_.
5. A blueprint for a software object is called a \_\_\_.
6. Common behavior can be defined in a \_\_\_ and inherited into a \_\_\_ using the \_\_\_ keyword.
7. A collection of methods with no implementation is called an \_\_\_.
8. A namespace that organizes classes and interfaces by functionality is called a \_\_\_.
9. The term API stands for \_\_\_?

### Exercises

1. Create new classes for each real-world object that you observed at the beginning of this trail. Refer to the Bicycle class if you forget the required syntax.
2. For each new class that you've created above, create an interface that defines its behavior, then require your class to implement it. Omit one or two methods and try compiling. What does the error look like?

## Answers to Questions and Exercises: Object-Oriented Programming Concepts

### Answers to Questions

1. Real-world objects contain **state** and **behavior**.
2. A software object's state is stored in **fields**.
3. A software object's behavior is exposed through **methods**.
4. Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data **encapsulation**.
5. A blueprint for a software object is called a **class**.
6. Common behavior can be defined in a **superclass** and inherited into a **subclass** using the **extends** keyword.
7. A collection of methods with no implementation is called an **interface**.
8. A namespace that organizes classes and interfaces by functionality is called a **package**.
9. The term API stands for **Application Programming Interface**.

### Answers to Exercises

1. Your answers will vary depending on the real-world objects that you are modeling.
2. Your answers will vary here as well, but the error message will specifically list the required methods that have not been implemented.

## 2. Answers to Questions and Exercises: Variables

### Questions and Exercises: Variables

#### Questions

1. The term "instance variable" is another name for \_\_\_\_.
2. The term "class variable" is another name for \_\_\_\_.
3. A local variable stores temporary state; it is declared inside a \_\_\_\_.
4. A variable declared within the opening and closing parenthesis of a method signature is called a \_\_\_\_.
5. What are the eight primitive data types supported by the Java programming language?
6. Character strings are represented by the class \_\_\_\_.
7. An \_\_\_\_ is a container object that holds a fixed number of values of a single type.

#### Exercises

1. Create a small program that defines some fields. Try creating some illegal field names and see what kind of error the compiler produces. Use the naming rules and conventions as a guide.
2. In the program you created in Exercise 1, try leaving the fields uninitialized and print out their values. Try the same with a local variable and see what kind of compiler errors you can produce. Becoming familiar with common compiler errors will make it easier to recognize bugs in your code.

#### Answers to Questions

1. The term "instance variable" is another name for **non-static field**.
2. The term "class variable" is another name for **static field**.
3. A local variable stores temporary state; it is declared inside a **method**.
4. A variable declared within the opening and closing parenthesis of a method is called a **parameter**.
5. What are the eight primitive data types supported by the Java programming language? **byte, short, int, long, float, double, boolean, char**
6. Character strings are represented by the class **java.lang.String**.
7. An **array** is a container object that holds a fixed number of values of a single type.

#### Answers to Exercises

1. Create a small program that defines some fields. Try creating some illegal field names and see what kind of error the compiler produces. Use the naming rules and conventions as a guide.  
There is no single correct answer here. Your results will vary depending on your code.
2. In the program you created in Exercise 1, try leaving the fields uninitialized and print out their values. Try the same with a local variable and see what kind of compiler errors you can produce. Becoming familiar with common compiler errors will make it easier to recognize bugs in your code.  
Again, there is no single correct answer for this exercise. Your results will vary depending on your code.

### 3. Questions and Exercises: Operators

#### Questions

1. Consider the following code snippet.  
2. `arrayOfInts[j] > arrayOfInts[j+1]`  
Which operators does the code contain?
3. Consider the following code snippet.  
`int i = 10;`  
`int n = i++%5;`
  - a. What are the values of i and n after the code is executed?
  - b. What are the final values of i and n if instead of using the postfix increment operator (i++), you use the prefix version (++i)?
4. To invert the value of a boolean, which operator would you use?
5. Which operator is used to compare two values, = or == ?
6. Explain the following code sample: `result = someCondition ? value1 : value2;`

#### Exercises

1. Change the following program to use compound assignments:  

```
class ArithmeticDemo {
    public static void main (String[] args){
        int result = 1 + 2; // result is now 3
        System.out.println(result);
        result = result - 1; // result is now 2
        System.out.println(result);
        result = result * 2; // result is now 4
        System.out.println(result);
        result = result / 2; // result is now 2
        System.out.println(result);
        result = result + 8; // result is now 10
        result = result % 7; // result is now 3
        System.out.println(result);
    }
}
```
2. In the following program, explain why the value "6" is printed twice in a row:  

```
class PrePostDemo {
    public static void main(String[] args){
        int i = 3;
        i++;
        System.out.println(i); // "4"
        ++i;
        System.out.println(i); // "5"
        System.out.println(++i); // "6"
        System.out.println(i++); // "6"
        System.out.println(i); // "7"
    }
}
```

### Answers to Questions and Exercises: Operators

#### Answers to Questions

1. Consider the following code snippet:  
`arrayOfInts[j] > arrayOfInts[j+1]`  
**Question:** What operators does the code contain?  
**Answer:** >, +
2. Consider the following code snippet:  
`int i = 10;`  
`int n = i++%5;`
  - a. **Question:** What are the values of i and n after the code is executed?  
**Answer:** i is 11, and n is 0.
  - b. **Question:** What are the final values of i and n if instead of using the postfix increment operator (i++), you use the prefix version (++i)?  
**Answer:** i is 11, and n is 1.

3. **Question:** To invert the value of a boolean, which operator would you use?  
**Answer:** The logical complement operator "!".
4. **Question:** Which operator is used to compare two values, = or == ?  
**Answer:** The == operator is used for comparison, and = is used for assignment.
5. **Question:** Explain the following code sample: result = someCondition ? value1 : value2;  
**Answer:** This code should be read as: "If someCondition is true, assign the value of value1 to result. Otherwise, assign the value of value2 to result."

### Exercises

1. Change the following program to use compound assignments:

```
class ArithmeticDemo {
    public static void main (String[] args){
        int result = 1 + 2; // result is now 3
        System.out.println(result);
        result = result - 1; // result is now 2
        System.out.println(result);
        result = result * 2; // result is now 4
        System.out.println(result);
        result = result / 2; // result is now 2
        System.out.println(result);
        result = result + 8; // result is now 10
        result = result % 7; // result is now 3
        System.out.println(result);
    }
}
```

Here is one solution:

```
class ArithmeticDemo {
    public static void main (String[] args){
        int result = 3;
        System.out.println(result);
        result -= 1; // result is now 2
        System.out.println(result);
        result *= 2; // result is now 4
        System.out.println(result);
        result /= 2; // result is now 2
        System.out.println(result);
        result += 8; // result is now 10
        result %= 7; // result is now 3
        System.out.println(result);
    }
}
```

2. In the following program, explain why the value "6" is printed twice in a row:

```
class PrePostDemo {
    public static void main(String[] args){
        int i = 3;
        i++;
        System.out.println(i); // "4"
        ++i;
        System.out.println(i); // "5"
        System.out.println(++i); // "6"
        System.out.println(i++); // "6"
        System.out.println(i); // "7"
    }
}
```

The code `System.out.println(++i);` evaluates to 6, because the prefix version of `++` evaluates to the incremented value. The next line, `System.out.println(i++);` evaluates to the current value (6), then increments by one. So "7" doesn't get printed until the next line.

#### 4. Questions and Exercises: Expressions, Statements, and Blocks

##### Questions

1. Operators may be used in building \_\_\_\_, which compute values.
2. Expressions are the core components of \_\_\_\_.
3. Statements may be grouped into \_\_\_\_.
4. The following code snippet is an example of a \_\_\_\_ expression.  
`1 * 2 * 3`
5. Statements are roughly equivalent to sentences in natural languages, but instead of ending with a period, a statement ends with a \_\_\_\_.
6. A block is a group of zero or more statements between balanced \_\_\_\_ and can be used anywhere a single statement is allowed.

##### Exercises

Identify the following kinds of expression statements:

- `aValue = 8933.234;`
- `aValue++;`
- `System.out.println("Hello World!");`
- `Bicycle myBike = new Bicycle();`

##### Answers to Questions and Exercises: Expressions, Statements, and Blocks

##### Answers to Questions

1. Operators may be used in building **expressions**, which compute values.
2. Expressions are the core components of **statements**.
3. Statements may be grouped into **blocks**.
4. The following code snippet is an example of a **compound** expression.  
`1 * 2 * 3`
5. Statements are roughly equivalent to sentences in natural languages, but instead of ending with a period, a statement ends with a **semicolon**.
6. A block is a group of zero or more statements between balanced **braces** and can be used anywhere a single statement is allowed.

##### Answers to Exercises

Identify the following kinds of expression statements:

- `aValue = 8933.234;` // **assignment statement**
- `aValue++;` // **increment statement**
- `System.out.println("Hello World!");` // **method invocation statement**
- `Bicycle myBike = new Bicycle();` // **object creation statement**

## 5. Questions and Exercises: Control Flow Statements

### Questions

1. The most basic control flow statement supported by the Java programming language is the \_\_\_ statement.
2. The \_\_\_ statement allows for any number of possible execution paths.
3. The \_\_\_ statement is similar to the while statement, but evaluates its expression at the \_\_\_ of the loop.
4. How do you write an infinite loop using the for statement?
5. How do you write an infinite loop using the while statement?

### Exercises

1. Consider the following code snippet.

```
if (aNumber >= 0)
    if (aNumber == 0)
        System.out.println("first string");
else System.out.println("second string");
System.out.println("third string");
```

  - a. What output do you think the code will produce if aNumber is 3?
  - b. Write a test program containing the previous code snippet; make aNumber 3. What is the output of the program? Is it what you predicted? Explain why the output is what it is; in other words, what is the control flow for the code snippet?
  - c. Using only spaces and line breaks, reformat the code snippet to make the control flow easier to understand.
  - d. Use braces, { and }, to further clarify the code.

## Answers to Questions and Exercises: Control Flow Statements

### Answers to Questions

1. The most basic control flow statement supported by the Java programming language is the **if-then** statement.
2. The **switch** statement allows for any number of possible execution paths.
3. The **do-while** statement is similar to the while statement, but evaluates its expression at the **bottom** of the loop.
4. **Question:** How do you write an infinite loop using the for statement?  
**Answer:**

```
for ( ; ; ) {
}
```
5. **Question:** How do you write an infinite loop using the while statement?  
**Answer:**

```
while (true) {
}
```

### Answers to Exercises

1. Consider the following code snippet.

```
if (aNumber >= 0)
    if (aNumber == 0)
        System.out.println("first string");
else
    System.out.println("second string");
System.out.println("third string");
```

  - a. **Exercise:** What output do you think the code will produce if aNumber is 3?  
**Solution:**  
second string  
third string
  - b. **Exercise:** Write a test program containing the previous code snippet; make aNumber 3. What is the output of the program? Is it what you predicted? Explain why the output is what it is. In other words, what is the control flow for the code snippet?  
**Solution:** [NestedIf](#)  
second string  
third string

3 is greater than or equal to 0, so execution progresses to the second if statement. The second if statement's test fails because 3 is not equal to 0. Thus, the else clause executes (since it's attached to the second if statement). Thus, second string is displayed. The final println is completely outside of any if statement, so it always gets executed, and thus third string is always displayed.

- c. **Exercise:** Using only spaces and line breaks, reformat the code snippet to make the control flow easier to understand.

**Solution:**

```
if (aNumber >= 0)
    if (aNumber == 0)
        System.out.println("first string");
    else
        System.out.println("second string");
System.out.println("third string");
```

- d. **Exercise:** Use braces { and } to further clarify the code and reduce the possibility of errors by future maintainers of the code.

**Solution:**

```
if (aNumber >= 0) {
    if (aNumber == 0) {
        System.out.println("first string");
    } else {
        System.out.println("second string");
    }
}
System.out.println("third string");
```



## 6. Questions and Exercises: Classes

### Questions

1. Consider the following class:

```
public class IdentifyMyParts {
    public static int x = 7;
    public int y = 3;
}
```

  - a. What are the class variables?
  - b. What are the instance variables?
  - c. What is the output from the following code:

```
IdentifyMyParts a = new IdentifyMyParts();
IdentifyMyParts b = new IdentifyMyParts();
a.y = 5;
b.y = 6;
a.x = 1;
b.x = 2;
System.out.println("a.y = " + a.y);
System.out.println("b.y = " + b.y);
System.out.println("a.x = " + a.x);
System.out.println("b.x = " + b.x);
System.out.println("IdentifyMyParts.x = " + IdentifyMyParts.x);
```

### Exercises

1. Write a class whose instances represent a single playing card from a deck of cards. Playing cards have two distinguishing properties: rank and suit. Be sure to keep your solution as you will be asked to rewrite it in [Enum Types](#).

#### Hint:

You can use the assert statement to check your assignments. You write:

```
assert (boolean expression to test);
```

If the boolean expression is false, you will get an error message. For example,

```
assert toString(ACE) == "Ace";
```

should return true, so there will be no error message.

If you use the assert statement, you must run your program with the ea flag:

```
java -ea YourProgram.class
```

2. Write a class whose instances represent a **full** deck of cards. You should also keep this solution.
3. Write a small program to test your deck and card classes. The program can be as simple as creating a deck of cards and displaying its cards.

### Answers to Questions and Exercises: Classes

#### Answers to Questions

1. Consider the following class:

```
public class IdentifyMyParts {
    public static int x = 7;
    public int y = 3;
}
```

  - a. **Question:** What are the class variables?  
**Answer:** x
  - b. **Question:** What are the instance variables?  
**Answer:** y
  - c. **Question:** What is the output from the following code:

```
IdentifyMyParts a = new IdentifyMyParts();
IdentifyMyParts b = new IdentifyMyParts();
a.y = 5;
b.y = 6;
a.x = 1;
b.x = 2;
System.out.println("a.y = " + a.y);
System.out.println("b.y = " + b.y);
System.out.println("a.x = " + a.x);
```

```
System.out.println("b.x = " + b.x);
System.out.println("IdentifyMyParts.x = " + IdentifyMyParts.x);
```

**Answer:** Here is the output:

```
a.y = 5
b.y = 6
a.x = 2
b.x = 2
IdentifyMyParts.x = 2
```

Because x is defined as a public static int in the class IdentifyMyParts, every reference to x will have the value that was last assigned because x is a static variable (and therefore a class variable) shared across all instances of the class. That is, there is only one x: when the value of x changes in any instance it affects the value of x for all instances of IdentifyMyParts.


This is covered in the Class Variables section of [Understanding Instance and Class Members](#).

#### Answers to Exercises

1. **Exercise:** Write a class whose instances represent a single playing card from a deck of cards. Playing cards have two distinguishing properties: rank and suit. Be sure to keep your solution as you will be asked to rewrite it in [Enum Types](#).

**Answer:** [Card.java](#) .

2. **Exercise:** Write a class whose instances represents a **full** deck of cards. You should also keep this solution.

**Answer:** See [Deck.java](#) .

3. **Exercise:** Write a small program to test your deck and card classes. The program can be as simple as creating a deck of cards and displaying its cards.

**Answer:** See [DisplayDeck.java](#) .

## 7. Questions and Exercises: Objects

### Questions

1. What's wrong with the following program?

```
public class SomethingIsWrong {
    public static void main(String[] args) {
        Rectangle myRect;
        myRect.width = 40;
        myRect.height = 50;
        System.out.println("myRect's area is " + myRect.area());
    }
}
```

2. The following code creates one array and one string object. How many references to those objects exist after the code executes? Is either object eligible for garbage collection?

```
...
String[] students = new String[10];
String studentName = "Peter Parker";
students[0] = studentName;
studentName = null;
```

3. How does a program destroy an object that it creates?

### Exercises

1. Fix the program called SomethingIsWrong shown in Question 1.
2. Given the following class, called [NumberHolder](#), write some code that creates an instance of the class, initializes its two member variables, and then displays the value of each member variable.

```
public class NumberHolder {
    public int anInt;
    public float aFloat;
}
```

### Answers to Questions and Exercises: Objects

#### Answers to Questions

1. **Question:** What's wrong with the following program?

```
public class SomethingIsWrong {
    public static void main(String[] args) {
        Rectangle myRect;
        myRect.width = 40;
        myRect.height = 50;
        System.out.println("myRect's area is " + myRect.area());
    }
}
```

**Answer:** The code never creates a [Rectangle](#) object. With this simple program, the compiler generates an error. However, in a more realistic situation, myRect might be initialized to null in one place, say in a constructor, and used later. In that case, the program will compile just fine, but will generate a `NullPointerException` at runtime.

2. **Question:** The following code creates one array and one string object. How many references to those objects exist after the code executes? Is either object eligible for garbage collection?

```
...
String[] students = new String[10];
String studentName = "Peter Smith";
students[0] = studentName;
studentName = null;
```

**Answer:** There is one reference to the students array and that array has one reference to the string Peter Smith. Neither object is eligible for garbage collection. The array students is not eligible for garbage collection because it has one reference to the object studentName even though that object has been assigned the value null. The object studentName is not eligible either because students[0] still refers to it.

3. **Question:** How does a program destroy an object that it creates?

**Answer:** A program does not explicitly destroy objects. A program can set all references to an object to null so that it becomes eligible for garbage collection. But the program does not actually destroy objects.

#### Answers to Exercises

1. **Exercise:** Fix the program called `SomethingIsWrong` shown in Question 1.

**Answer:** See [SomethingIsRight](#):

```
public class SomethingIsRight {
    public static void main(String[] args) {
        Rectangle myRect = new Rectangle();
        myRect.width = 40;
        myRect.height = 50;
        System.out.println("myRect's area is " + myRect.area());
    }
}
```

2. **Exercise:** Given the following class, called [NumberHolder](#), write some code that creates an instance of the class, initializes its two member variables, and then displays the value of each member variable.

```
public class NumberHolder {
    public int anInt;
    public float aFloat;
}
```

**Answer:** See [NumberHolderDisplay](#):

```
public class NumberHolderDisplay {
    public static void main(String[] args) {
        NumberHolder aNumberHolder = new NumberHolder();
        aNumberHolder.anInt = 1;
        aNumberHolder.aFloat = 2.3f;
        System.out.println(aNumberHolder.anInt);
        System.out.println(aNumberHolder.aFloat);
    }
}
```

## 8. Questions and Exercises: Nested Classes

### Questions

1. The program [Problem.java](#) doesn't compile. What do you need to do to make it compile? Why?
2. Use the Java API documentation for the [Box](#) class (in the javax.swing package) to help you answer the following questions.
  - a. What static nested class does Box define?
  - b. What inner class does Box define?
  - c. What is the superclass of Box's inner class?
  - d. Which of Box's nested classes can you use from any class?
  - e. How do you create an instance of Box's Filler class?

### Exercises

1. Get the file [Class1.java](#). Compile and run Class1. What is the output?
2. The following exercises involve modifying the class [DataStructure.java](#), which the section [Inner Class Example](#) discusses.
  1. Define a method named print(DataStructureIterator iterator). Invoke this method with an instance of the class EvenIterator so that it performs the same function as the method printEven.
  2. Invoke the method print(DataStructureIterator iterator) so that it prints elements that have an odd index value. Use an anonymous class as the method's argument instead of an instance of the interface DataStructureIterator.
  3. Define a method named print(java.util.Function<Integer, Boolean> iterator) that performs the same function as print(DataStructureIterator iterator). Invoke this method with a lambda expression to print elements that have an even index value. Invoke this method again with a lambda expression to print elements that have an odd index value.
  4. Define two methods so that the following two statements print elements that have an even index value and elements that have an odd index value:

```
DataStructure ds = new DataStructure()
// ...
ds.print(DataStructure::isEvenIndex);
ds.print(DataStructure::isOddIndex);
```

### Answers to Questions and Exercises: Nested Classes

#### Answers to Questions

1. **Question:** The program [Problem.java](#) doesn't compile. What do you need to do to make it compile? Why?  
**Answer:** Delete static in front of the declaration of the Inner class. An static inner class does not have access to the instance fields of the outer class. See [ProblemSolved.java](#).
2. Use the Java API documentation for the [Box](#) class (in the javax.swing package) to help you answer the following questions.
  - a. **Question:** What static nested class does Box define?  
**Answer:** Box.Filler
  - b. **Question:** What inner class does Box define?  
**Answer:** Box.AccessibleBox
  - c. **Question:** What is the superclass of Box's inner class?  
**Answer:** [java.awt.]Container.AccessibleAWTContainer
  - d. **Question:** Which of Box's nested classes can you use from any class?  
**Answer:** Box.Filler
  - e. **Question:** How do you create an instance of Box's Filler class?  
**Answer:** new Box.Filler(minDimension, prefDimension, maxDimension)

#### Answers to Exercises

1. **Exercise:** Get the file [Class1.java](#). Compile and run Class1. What is the output?  
**Answer:** InnerClass1: getString invoked.  
InnerClass1: getAnotherString invoked.
2. **Exercise:** The following exercises involve modifying the class [DataStructure.java](#), which the section [Inner Class Example](#) discusses.
  1. Define a method named print(DataStructureIterator iterator). Invoke this method with an instance of the class EvenIterator so that it performs the same function as the method printEven.

**Hint:** These statements do not compile if you specify them in the main method:

```
DataSet ds = new DataSet();  
ds.print(new EvenIterator());
```

The compiler generates the error message, "non-static variable this cannot be referenced from a static context" when it encounters the expression `new EvenIterator()`. The class `EvenIterator` is an inner, non-static class. This means that you can create an instance of `EvenIterator` only inside an instance of the outer class, `DataSet`.

You can define a method in `DataSet` that creates and returns a new instance of `EvenIterator`.

2. Invoke the method `print(DataStructureIterator iterator)` so that it prints elements that have an odd index value. Use an anonymous class as the method's argument instead of an instance of the interface `DataStructureIterator`.

**Hint:** You cannot access the private members `SIZE` and `arrayOfInts` outside the class `DataSet`, which means that you cannot access these private members from an anonymous class defined outside `DataSet`.

You can define methods that access the private members `SIZE` and `arrayOfInts` and then use them in your anonymous class.

3. Define a method named `print(java.util.Function<Integer, Boolean> iterator)` that performs the same function as `print(DataStructureIterator iterator)`. Invoke this method with a lambda expression to print elements that have an even index value. Invoke this method again with a lambda expression to print elements that have an odd index value.

**Hint:** In this `print` method, you can step through the elements contained in the array `arrayOfInts` with a `for` expression. For each index value, invoke the method `function.apply`. If this method returns a true value for a particular index value, print the element contained in that index value.

To invoke this `print` method to print elements that have an even index value, you can specify a lambda expression that implements the method `Boolean Function.apply(Integer t)`. This lambda expression takes one `Integer` argument (the index) and returns a `Boolean` value (`Boolean.TRUE` if the index value is even, `Boolean.FALSE` otherwise).

4. Define two methods so that these statements print elements that have an even index value and then elements that have an odd index value:

```
DataSet ds = new DataSet()  
// ...  
ds.print(DataStructure::isEvenIndex);  
    ds.print(DataStructure::isOddIndex);
```

**Hint:** Create two methods in the class `DataSet` named `isEvenIndex` and `isOddIndex` that have the same parameter list and return type as the abstract method `Boolean Function<Integer, Boolean>.apply(Integer t)`. This means that the methods take one `Integer` argument (the index) and return a `Boolean` value.

**Answer:** See the file [DataSet.java](#).

## 9. Questions and Exercises: Enum Types

### Questions

1. True or false: an Enum type can be a subclass of java.lang.String.

### Exercises

1. Rewrite the class Card from the exercise in [Questions and Exercises: Classes](#) so that it represents the rank and suit of a card with enum types.
2. Rewrite the Deck class.

### Answers to Questions and Exercises: Enum Types

#### Answers to Questions

1. **Question:** True or false: an Enum type can be a subclass of java.lang.String.  
**Answer:** False. All enums implicitly extend java.lang.Enum. Because a class can only extend one parent, the Java language does not support multiple inheritance of state, and therefore an enum cannot extend anything else.

#### Answers to Exercises

1. **Exercise:** Rewrite the class Card from the exercise in [Questions and Exercises: Classes](#) so that it represents the rank and suit of a card with enum types.  
**Answer:** See [Card3.java](#), [Suit.java](#), and [Rank.java](#).
2. **Exercise:** Rewrite the Deck class.  
**Answer:** See [Deck3.java](#).

## 10. Questions and Exercises: Interfaces

### Questions

1. What methods would a class that implements the `java.lang.CharSequence` interface have to implement?
2. What is wrong with the following interface?

```
public interface SomethingIsWrong {  
    void aMethod(int aValue){  
        System.out.println("Hi Mom"); }  
}
```

3. Fix the interface in question 2.
4. Is the following interface valid?

```
public interface Marker {  
}
```

### Exercises

1. Write a class that implements the `CharSequence` interface found in the `java.lang` package. Your implementation should return the string backwards. Select one of the sentences from this book to use as the data. Write a small main method to test your class; make sure to call all four methods.
2. Suppose you have written a time server that periodically notifies its clients of the current date and time. Write an interface the server could use to enforce a particular protocol on its clients.

### Answers to Questions and Exercises: Interfaces

#### Answers to Questions

**Question 1:** What methods would a class that implements the `java.lang.CharSequence` interface have to implement?

**Answer 1:** `charAt`, `length`, `subSequence`, and `toString`.

**Question 2:** What is wrong with the following interface?

```
public interface SomethingIsWrong {  
    void aMethod(int aValue) {  
        System.out.println("Hi Mom"); }  
}
```

**Answer 2:** It has a method implementation in it. Only default and static methods have implementations.

**Question 3:** Fix the interface in Question 2.

**Answer 3:**

```
public interface SomethingIsWrong {  
    void aMethod(int aValue); }
```

Alternatively, you can define `aMethod` as a default method:

```
public interface SomethingIsWrong {  
    default void aMethod(int aValue) {  
        System.out.println("Hi Mom"); }  
}
```

**Question 4:** Is the following interface valid?

```
public interface Marker {  
}
```

**Answer 4:** Yes. Methods are not required. Empty interfaces can be used as types and to mark classes without requiring any particular method implementations. For an example of a useful empty interface, see `java.io.Serializable`.

#### Answers to Exercises

**Exercise 1:** Write a class that implements the `CharSequence` interface found in the `java.lang` package. Your implementation should return the string backwards. Select one of the sentences from this book to use as the data. Write a small main method to test your class; make sure to call all four methods.

Answer 1: See [CharSequenceDemo.java](#)

**Exercise 2:** Suppose that you have written a time server, which periodically notifies its clients of the current date and time. Write an interface that the server could use to enforce a particular protocol on its clients.

Answer 2: See [TimeClient.java](#).



## 11. Questions and Exercises: Inheritance

### Questions

1. Consider the following two classes:

```
public class ClassA {
    public void methodOne(int i) { }
    public void methodTwo(int i) { }
    public static void methodThree(int i) { }
    public static void methodFour(int i) { }
}
public class ClassB extends ClassA {
    public static void methodOne(int i) { }
    public void methodTwo(int i) { }
    public void methodThree(int i) { }
    public static void methodFour(int i) { }
}
```

- Which method overrides a method in the superclass?
- Which method hides a method in the superclass?
- What do the other methods do?

2. Consider the [Card](#), [Deck](#), and [DisplayDeck](#) classes you wrote in [Questions and Exercises: Classes](#). What Object methods should each of these classes override?

### Exercises

1. Write the implementations for the methods that you answered in question 2.

### Answers to Questions and Exercises: Inheritance

#### Questions

**Question 1:** Consider the following two classes:

```
public class ClassA {
    public void methodOne(int i) { }
    public void methodTwo(int i) { }
    public static void methodThree(int i) { }
    public static void methodFour(int i) { }
}
public class ClassB extends ClassA {
    public static void methodOne(int i) { }
    public void methodTwo(int i) { }
    public void methodThree(int i) { }
    public static void methodFour(int i) { }
}
```

**Question 1a:** Which method overrides a method in the superclass?

**Answer 1a:** methodTwo

**Question 1b:** Which method hides a method in the superclass?

**Answer 1b:** methodFour

**Question 1c:** What do the other methods do?

**Answer 1c:** They cause compile-time errors.

**Question 2:** Consider the [Card](#), [Deck](#), and [DisplayDeck](#) classes you wrote in the previous exercise. What Object methods should each of these classes override?

**Answer 2:** Card and Deck should override equals, hashCode, and toString.

#### Answers to Exercises

**Exercise 1:** Write the implementations for the methods that you answered in question 2.

**Answer 1:** See [Card2](#).

## 12. Questions and Exercises: Numbers

### Questions

- Use the API documentation to find the answers to the following questions:
  - What Integer method can you use to convert an int into a string that expresses the number in hexadecimal? For example, what method converts the integer 65 into the string "41"?
  - What Integer method would you use to convert a string expressed in base 5 into the equivalent int? For example, how would you convert the string "230" into the integer value 65? Show the code you would use to accomplish this task.
  - What Double method can you use to detect whether a floating-point number has the special value Not a Number (NaN)?
- What is the value of the following expression, and why?  
`Integer.valueOf(1).equals(Long.valueOf(1))`

### Exercises

- Change [MaxVariablesDemo](#) to show minimum values instead of maximum values. You can delete all code related to the variables `aChar` and `aBoolean`. What is the output?
- Create a program that reads an unspecified number of integer arguments from the command line and adds them together. For example, suppose that you enter the following:  
`java Adder 1 3 2 10`  
The program should display 16 and then exit. The program should display an error message if the user enters only one argument. You can base your program on [ValueOfDemo](#).
- Create a program that is similar to the previous one but has the following differences:
  - Instead of reading integer arguments, it reads floating-point arguments.
  - It displays the sum of the arguments, using exactly two digits to the right of the decimal point.For example, suppose that you enter the following:  
`java FPAdder 1 1e2 3.0 4.754`  
The program would display 108.75. Depending on your locale, the decimal point might be a comma (,) instead of a period (.).

### Answers to Questions and Exercises: Numbers

#### Answers to Questions

- Use the API documentation to find the answers to the following questions:
  - Question:** What Integer method can you use to convert an int into a string that expresses the number in hexadecimal? For example, what method converts the integer 65 into the string "41"?  
**Answer:** `toHexString`
  - Question:** What Integer method would you use to convert a string expressed in base 5 into the equivalent int? For example, how would you convert the string "230" into the integer value 65? Show the code you would use to accomplish this task.  
**Answer:** `valueOf`. Here's how:  

```
String base5String = "230";
int result = Integer.valueOf(base5String, 5);
```
  - Question:** What Double method can you use to detect whether a floating-point number has the special value Not a Number (NaN)?  
**Answer:** `isNaN`
- Question:** What is the value of the following expression, and why?  
`Integer.valueOf(1).equals(Long.valueOf(1))`
- Question:** What is the value of the following expression, and why?  
`Integer.valueOf(1).equals(Long.valueOf(1))`  
**Answer:** `False`. The two objects (the Integer and the Long) have different types.

#### Answers to Exercises

- Exercise:** Change [MaxVariablesDemo](#) to show minimum values instead of maximum values. You can delete all code related to the variables `aChar` and `aBoolean`. What is the output?  
**Answer:** See [MinVariablesDemo](#). Here is the output:  
The smallest byte value is -128  
The smallest short value is -32768  
The smallest integer value is -2147483648  
The smallest long value is -9223372036854775808  
The smallest float value is 1.4E-45  
The smallest double value is 4.9E-324

2. **Exercise:** Create a program that reads an unspecified number of integer arguments from the command line and adds them together. For example, suppose that you enter the following:
- ```
java Adder 1 3 2 10
```
- The program should display 16 and then exit. The program should display an error message if the user enters only one argument. You can base your program on [ValueOfDemo](#).

**Answer:** See [Adder](#).

3. **Exercise:** Create a program that is similar to the previous one but has the following differences:
- Instead of reading integer arguments, it reads floating-point arguments.
  - It displays the sum of the arguments, using exactly two digits to the right of the decimal point.

For example, suppose that you enter the following:

```
java FPAdder 1 1e2 3.0 4.754
```

The program would display 108.75. Depending on your locale, the decimal point might be a comma (,) instead of a period (.).

**Answer:** See [FPAdder](#).

### 13. Questions and Exercises: Characters and Strings

#### Questions

1. What is the initial capacity of the following string builder?  
`StringBuilder sb = new StringBuilder("Able was I ere I saw Elba.");`
2. Consider the following string:
3. `String hannah = "Did Hannah see bees? Hannah did.";`
  1. What is the value displayed by the expression `hannah.length()`?
  2. What is the value returned by the method call `hannah.charAt(12)`?
  3. Write an expression that refers to the letter b in the string referred to by hannah.
4. How long is the string returned by the following expression? What is the string?
5. `"Was it a car or a cat I saw?".substring(9, 12)`
6. In the following program, called [ComputeResult](#), what is the value of result after each numbered line executes?

```
public class ComputeResult {
    public static void main(String[] args) {
        String original = "software";
        StringBuilder result = new StringBuilder("hi");
        int index = original.indexOf('a');
        /*1*/ result.setCharAt(0, original.charAt(0));
        /*2*/ result.setCharAt(1, original.charAt(original.length()-1));
        /*3*/ result.insert(1, original.charAt(4));
        /*4*/ result.append(original.substring(1,4));
        /*5*/ result.insert(3, (original.substring(index, index+2) + " "));
        System.out.println(result);}}

```

#### Exercises

1. Show two ways to concatenate the following two strings together to get the string "Hi, mom.":  
`String hi = "Hi, ";`  
`String mom = "mom.";`
2. Write a program that computes your initials from your full name and displays them.
3. An anagram is a word or a phrase made by transposing the letters of another word or phrase; for example, "parliament" is an anagram of "partial men," and "software" is an anagram of "swear oft." Write a program that figures out whether one string is an anagram of another string. The program should ignore white space and punctuation.

#### Answers to Questions and Exercises: Characters and Strings

##### Answers to Questions

1. **Question:** What is the initial capacity of the following string builder?  
`StringBuilder sb = new StringBuilder("Able was I ere I saw Elba.");`  
**Answer:** It's the length of the initial string + 16:  $26 + 16 = 42$ .
2. Consider the following string:  
`String hannah = "Did Hannah see bees? Hannah did.";`
  1. **Question:** What is the value displayed by the expression `hannah.length()`?  
**Answer:** 32.
  2. **Question:** What is the value returned by the method call `hannah.charAt(12)`?  
**Answer:** e.
  3. **Question:** Write an expression that refers to the letter b in the string referred to by hannah.  
**Answer:** `hannah.charAt(15)`.
3. **Question:** How long is the string returned by the following expression? What is the string?  
`"Was it a car or a cat I saw?".substring(9, 12)`  
**Answer:** It's 3 characters in length: car. It does not include the space after car.
4. **Question:** In the following program, called [ComputeResult](#), what is the value of result after each numbered line executes?

```
public class ComputeResult {
    public static void main(String[] args) {
        String original = "software";
        StringBuilder result = new StringBuilder("hi");
        int index = original.indexOf('a');
```

```

/*1*/ result.setCharAt(0, original.charAt(0));
/*2*/ result.setCharAt(1, original.charAt(original.length()-1));
/*3*/ result.insert(1, original.charAt(4));
/*4*/ result.append(original.substring(1,4));
/*5*/ result.insert(3, (original.substring(index, index+2) + " "));
    System.out.println(result);}}

```

**Answer:**

1. si
2. se
3. swe
4. swoeft
5. swear oft

**Answers to Exercises**

1. **Exercise:** Show two ways to concatenate the following two strings together to get the string "Hi, mom.":

```
String hi = "Hi, ";
```

```
String mom = "mom.";
```

**Answer:** hi.concat(mom) and hi + mom.

2. **Exercise:** Write a program that computes your initials from your full name and displays them.

**Answer:** [ComputeInitials](#)

```

public class ComputeInitials {
    public static void main(String[] args) {
        String myName = "Fred F. Flintstone";
        StringBuffer myInitials = new StringBuffer();
        int length = myName.length();
        for (int i = 0; i < length; i++) {
            if (Character.isUpperCase(myName.charAt(i))) {
                myInitials.append(myName.charAt(i)); } }
        System.out.println("My initials are: " + myInitials); }
}

```

3. **Exercise:** An anagram is a word or a phrase made by transposing the letters of another word or phrase; for example, "parliament" is an anagram of "partial men," and "software" is an anagram of "swear oft." Write a program that figures out whether one string is an anagram of another string. The program should ignore white space and punctuation.

**Answer:** [Anagram](#)

```

public class Anagram {
    public static boolean areAnagrams(String string1, String string2) {

        String workingCopy1 = removeJunk(string1);
        String workingCopy2 = removeJunk(string2);

        workingCopy1 = workingCopy1.toLowerCase();
        workingCopy2 = workingCopy2.toLowerCase();

        workingCopy1 = sort(workingCopy1);
        workingCopy2 = sort(workingCopy2);

        return workingCopy1.equals(workingCopy2);
    }

    protected static String removeJunk(String string) {
        int i, len = string.length();
        StringBuilder dest = new StringBuilder(len);
        char c;
        for (i = (len - 1); i >= 0; i--) {
            c = string.charAt(i);
            if (Character.isLetter(c)) {
                dest.append(c); }
        }
        return dest.toString(); }
}

```

```
protected static String sort(String string) {
    char[] charArray = string.toCharArray();
    java.util.Arrays.sort(charArray);
    return new String(charArray);
}

public static void main(String[] args) {
    String string1 = "Cosmo and Laine:";
    String string2 = "Maid, clean soon!";

    System.out.println();
    System.out.println("Testing whether the following " + "strings are anagrams:");
    System.out.println("  String 1: " + string1);
    System.out.println("  String 2: " + string2);
    System.out.println();

    if (areAnagrams(string1, string2)) {
        System.out.println("They ARE anagrams!");
    } else {
        System.out.println("They are NOT anagrams!");
    }
    System.out.println();
}
}
```

#### 14. Questions and Exercises: Generics

1. Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).

2. Will the following class compile? If not, why?

```
public final class Algorithm {
    public static <T> T max(T x, T y) {
        return x > y ? x : y;
    }
}
```

3. Write a generic method to exchange the positions of two different elements in an array.

4. If the compiler erases all type parameters at compile time, why should you use generics?

5. What is the following class converted to after type erasure?

```
public class Pair<K, V> {
    public Pair(K key, V value) {
        this.key = key;
        this.value = value; }
    public K getKey() ;           { return key; }
    public V getValue() ;         { return value; }
    public void setKey(K key)     { this.key = key; }
    public void setValue(V value) { this.value = value; }
    private K key;
    private V value;
}
```

6. What is the following method converted to after type erasure?

```
public static <T extends Comparable<T>>
    int findFirstGreaterThan(T[] at, T elem) {
    // ...
}
```

7. Will the following method compile? If not, why?

```
public static void print(List<? extends Number> list) {
    for (Number n : list)
        System.out.print(n + " ");
    System.out.println();}
}
```

8. Write a generic method to find the maximal element in the range [begin, end) of a list.

9. Will the following class compile? If not, why?

```
public class Singleton<T> {
    public static T getInstance() {
        if (instance == null)
            instance = new Singleton<T>();
        return instance; }
    private static T instance = null;
}
```

10. Given the following classes:

```
class Shape { /* ... */ }
class Circle extends Shape { /* ... */ }
class Rectangle extends Shape { /* ... */ }
class Node<T> { /* ... */ }
```

Will the following code compile? If not, why?

```
Node<Circle> nc = new Node<>();
Node<Shape> ns = nc;
```

11. Consider this class:

```
class Node<T> implements Comparable<T> {
    public int compareTo(T obj) { /* ... */ }
    // ...
}
```

12. Will the following code compile? If not, why?

```
Node<String> node = new Node<>();
Comparable<String> comp = node;
```

13. How do you invoke the following method to find the first integer in a list that is relatively prime to a list of specified integers?

```
public static <T>
```

```
    int findFirst(List<T> list, int begin, int end, UnaryPredicate<T> p)
```

Note that two integers  $a$  and  $b$  are relatively prime if  $\text{gcd}(a, b) = 1$ , where  $\text{gcd}$  is short for greatest common divisor.

### Answer to Questions and Exercises: Generics

#### Answers to Questions

1. Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).

**Answer:**

```
public final class Algorithm {
    public static <T> int countIf(Collection<T> c, UnaryPredicate<T> p) {
        int count = 0;
        for (T elem : c)
            if (p.test(elem))
                ++count;
        return count;
    }
}
```

where the generic `UnaryPredicate` interface is defined as follows:

```
public interface UnaryPredicate<T> {
    public boolean test(T obj);
}
```

For example, the following program counts the number of odd integers in an integer list:

```
import java.util.*;
class OddPredicate implements UnaryPredicate<Integer> {
    public boolean test(Integer i) { return i % 2 != 0; }
}
public class Test {
    public static void main(String[] args) {
        Collection<Integer> ci = Arrays.asList(1, 2, 3, 4);
        int count = Algorithm.countIf(ci, new OddPredicate());
        System.out.println("Number of odd integers = " + count);
    }
}
```

The program prints:

Number of odd integers = 2

2. Will the following class compile? If not, why?

```
public final class Algorithm {
    public static <T> T max(T x, T y) {
        return x > y ? x : y;
    }
}
```

**Answer:** No. The greater than ( $>$ ) operator applies only to primitive numeric types.

3. Write a generic method to exchange the positions of two different elements in an array.

**Answer:**

```
public final class Algorithm {
    public static <T> void swap(T[] a, int i, int j) {
        T temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
}
```



4. If the compiler erases all type parameters at compile time, why should you use generics?

**Answer:** You should use generics because:

- The Java compiler enforces tighter type checks on generic code at compile time.
- Generics support programming types as parameters.
- Generics enable you to implement generic algorithms.

5. What is the following class converted to after type erasure?

```
public class Pair<K, V> {
    public Pair(K key, V value) {
        this.key = key;
        this.value = value; }
    public K getKey(); { return key; }
    public V getValue(); { return value; }
    public void setKey(K key) { this.key = key; }
    public void setValue(V value) { this.value = value; }
    private K key;
    private V value;
}
```

**Answer:**

```
public class Pair {
    public Pair(Object key, Object value) {
        this.key = key;
        this.value = value; }
    public Object getKey() { return key; }
    public Object getValue() { return value; }
    public void setKey(Object key) { this.key = key; }
    public void setValue(Object value) { this.value = value; }
    private Object key;
    private Object value;
}
```

6. What is the following method converted to after type erasure?

```
public static <T extends Comparable<T>>
    int findFirstGreaterThan(T[] at, T elem) {
    // ...
}
```

**Answer:**

```
public static int findFirstGreaterThan(Comparable[] at, Comparable elem) {
    // ...
}
```

7. Will the following method compile? If not, why?

```
public static void print(List<? extends Number> list) {
    for (Number n : list)
        System.out.print(n + " ");
    System.out.println();
}
```

**Answer:** Yes.

8. Write a generic method to find the maximal element in the range [begin, end) of a list.

**Answer:**

```
import java.util.*;
public final class Algorithm {
    public static <T extends Object & Comparable<? super T>>
        T max(List<? extends T> list, int begin, int end) {
        T maxElem = list.get(begin);
        for (++begin; begin < end; ++begin)
            if (maxElem.compareTo(list.get(begin)) < 0)
                maxElem = list.get(begin);
        return maxElem; }}
```

9. Will the following class compile? If not, why?

```
public class Singleton<T> {
    public static T getInstance() {
        if (instance == null)
            instance = new Singleton<T>();
        return instance;
    }
    private static T instance = null;
}
```

**Answer:** No. You cannot create a static field of the type parameter T.

10. Given the following classes:

```
class Shape { /* ... */ }
class Circle extends Shape { /* ... */ }
class Rectangle extends Shape { /* ... */ }
class Node<T> { /* ... */ }
```

Will the following code compile? If not, why?

```
Node<Circle> nc = new Node<>();
Node<Shape> ns = nc;
```

**Answer:** No. Because Node<Circle> is not a subtype of Node<Shape>.

11. Consider this class:

```
class Node<T> implements Comparable<T> {
    public int compareTo(T obj) { /* ... */ }
    // ...
}
```

Will the following code compile? If not, why?

```
Answer: Yes.
Node<String> node = new Node<>();
Comparable<String> comp = node;
```

12. How do you invoke the following method to find the first integer in a list that is relatively prime to a list of specified integers?

```
public static <T>
```

```
int findFirst(List<T> list, int begin, int end, UnaryPredicate<T> p)
```

Note that two integers  $a$  and  $b$  are relatively prime if  $\text{gcd}(a, b) = 1$ , where gcd is short for greatest common divisor.

**Answer:**

```
import java.util.*;
public final class Algorithm {
    public static <T>
        int findFirst(List<T> list, int begin, int end, UnaryPredicate<T> p) {
        for (; begin < end; ++begin)
            if (p.test(list.get(begin)))
                return begin;
        return -1;
    }
    // x > 0 and y > 0
    public static int gcd(int x, int y) {
        for (int r; (r = x % y) != 0; x = y, y = r) {}
        return y;
    }
}
```

The generic UnaryPredicate interface is defined as follows:

```
public interface UnaryPredicate<T> {
    public boolean test(T obj);
}
```

The following program tests the findFirst method:

```
import java.util.*;
class RelativelyPrimePredicate implements UnaryPredicate<Integer> {
```

```

public RelativelyPrimePredicate(Collection<Integer> c) {
    this.c = c;
}
public boolean test(Integer x) {
    for (Integer i : c)
        if (Algorithm.gcd(x, i) != 1)
            return false;
    return c.size() > 0;
}
private Collection<Integer> c;
}
public class Test {
    public static void main(String[] args) throws Exception {
        List<Integer> li = Arrays.asList(3, 4, 6, 8, 11, 15, 28, 32);
        Collection<Integer> c = Arrays.asList(7, 18, 19, 25);
        UnaryPredicate<Integer> p = new RelativelyPrimePredicate(c);
        int i = Algorithm.findFirst(li, 0, li.size(), p);
        if (i != -1) {
            System.out.print(li.get(i) + " is relatively prime to ");
            for (Integer k : c)
                System.out.print(k + " ");
            System.out.println();
        }
    }
}

```

**The program prints:**

11 is relatively prime to 7 18 19 25

## 15 Questions and Exercises: Creating and Using Packages

### Questions

Assume you have written some classes. Belatedly, you decide they should be split into three packages, as listed in the following table. Furthermore, assume the classes are currently in the default package (they have no package statements).

| Destination Packages |            |
|----------------------|------------|
| Package Name         | Class Name |
| mygame.server        | Server     |
| mygame.shared        | Utilities  |
| mygame.client        | Client     |

1. Which line of code will you need to add to each source file to put each class in the right package?
2. To adhere to the directory structure, you will need to create some subdirectories in the development directory and put source files in the correct subdirectories. What subdirectories must you create? Which subdirectory does each source file go in?
3. Do you think you'll need to make any other changes to the source files to make them compile correctly? If so, what?

### Exercises

Download the source files as listed here.

- [Client](#)
  - [Server](#)
  - [Utilities](#)
1. Implement the changes you proposed in questions 1 through 3 using the source files you just downloaded.
  2. Compile the revised source files. (*Hint*: If you're invoking the compiler from the command line (as opposed to using a builder), invoke the compiler from the directory that contains the mygame directory you just created.)

## Answers to Questions and Exercises: Creating and Using Packages

### Answers to Questions

Question 1: Assume that you have written some classes. Belatedly, you decide that they should be split into three packages, as listed in the table below. Furthermore, assume that the classes are currently in the default package (they have no package statements).

| Package Name  | Class Name |
|---------------|------------|
| mygame.server | Server     |
| mygame.shared | Utilities  |
| mygame.client | Client     |

a. What line of code will you need to add to each source file to put each class in the right package?

Answer 1a: The first line of each file must specify the package:

In Client.java add:

```
package mygame.client;
```

In Server.java add:

```
package mygame.server;;
```

In Utilities.java add:

```
package mygame.shared;
```

b. To adhere to the directory structure, you will need to create some subdirectories in your development directory, and put source files in the correct subdirectories. What subdirectories must you create? Which subdirectory does each source file go in?

Answer 1b: Within the mygame directory, you need to create three subdirectories: client, server, and shared.

In mygame/client/ place:

```
Client.java
```

In mygame/server/ place:

```
Server.java
```

In mygame/shared/ place:

```
Utilities.java
```

c. Do you think you'll need to make any other changes to the source files to make them compile correctly? If so, what?

Answer 1c: Yes, you need to add import statements. Client.java and Server.java need to import the Utilities class, which they can do in one of two ways:

```
import mygame.shared.*;
```

--or--

```
import mygame.shared.Utilities;
```

Also, Server.java needs to import the Client class:

```
import mygame.client.Client;
```

### Answers to Exercises

Exercise 1: Download three source files:

- [Client](#)
- [Server](#)
- [Utilities](#)

a. Implement the changes you proposed in question 1, using the source files you just downloaded.

b. Compile the revised source files. (*Hint*: If you're invoking the compiler from the command line (as opposed to using a builder), invoke the compiler from the directory that contains the mygame directory you just created.)

Answer 1: Download this zip file with the solution: [mygame.zip](#)

You might need to change your proposed import code to reflect our implementation.

## 16. Questions and Exercises

### Questions

1. Is the following code legal?  

```
try {  
} finally {}
```
2. What exception types can be caught by the following handler?  

```
catch (Exception e) {}
```

What is wrong with using this type of exception handler?
3. Is there anything wrong with the following exception handler as written? Will this code compile?  

```
try {  
} catch (Exception e) {  
} catch (ArithmeticException a) {}
```
4. Match each situation in the first list with an item in the second list.
  - a. 

```
int[] A;  
A[0] = 0;
```
  - b. The JVM starts running your program, but the JVM can't find the Java platform classes. (The Java platform classes reside in classes.zip or rt.jar.)
  - c. A program is reading a stream and reaches the end of stream marker.
  - d. Before closing the stream and after reaching the end of stream marker, a program tries to read the stream again.
  - 1 \_\_error
  - 2 \_\_checked exception
  - 3 \_\_compile error
  - 4 \_\_no exception

### Exercises

5. Add a readList method to [ListOfNumbers.java](#). This method should read in int values from a file, print each value, and append them to the end of the vector. You should catch all appropriate errors. You will also need a text file containing numbers to read in.
6. Modify the following cat method so that it will compile.

```
public static void cat(File file) {  
    RandomAccessFile input = null;  
    String line = null;  
    try {  
        input = new RandomAccessFile(file, "r");  
        while ((line = input.readLine()) != null) {  
            System.out.println(line); }  
        return;  
    } finally {  
        if (input != null) {  
            input.close(); }  
    }  
}
```

### Answers to Questions and Exercises

#### Answers to Questions

1. **Question:** Is the following code legal?  

```
try {  
} finally {  
}
```

**Answer:** Yes, it's legal — and very useful. A try statement does not have to have a catch block if it has a finally block. If the code in the try statement has multiple exit points and no associated catch clauses, the code in the finally block is executed no matter how the try block is exited. Thus it makes sense to provide a finally block whenever there is code that *must always* be executed. This includes resource recovery code, such as the code to close I/O streams.

2. **Question:** What exception types can be caught by the following handler?

```
catch (Exception e) {  
}
```

What is wrong with using this type of exception handler?

**Answer:** This handler catches exceptions of type Exception; therefore, it catches any exception. This can be a poor implementation because you are losing valuable information about the type of exception being thrown and making your code less efficient. As a result, your program may be forced to determine the type of exception before it can decide on the best recovery strategy.

3. **Question:** Is there anything wrong with this exception handler as written? Will this code compile?

```
try {  
} catch (Exception e) {  
} catch (ArithmeticException a) {}
```

**Answer:** This first handler catches exceptions of type Exception; therefore, it catches any exception, including ArithmeticException. The second handler could never be reached. This code will not compile.

4. **Question:** Match each situation in the first list with an item in the second list.

```
int[] A;
```

```
A[0] = 0;
```

- The JVM starts running your program, but the JVM can't find the Java platform classes. (The Java platform classes reside in classes.zip or rt.jar.)
- A program is reading a stream and reaches the end of stream marker.
- Before closing the stream and after reaching the end of stream marker, a program tries to read the stream again.

1 \_\_error

2 \_\_checked exception

3 \_\_compile error

4 \_\_no exception

**Answer:**

d. 3 (compile error). The array is not initialized and will not compile.

e. 1 (error).

f. 4 (no exception). When you read a stream, you expect there to be an end of stream marker. You should use exceptions to catch unexpected behavior in your program.

g. 2 (checked exception).

#### Answers to Exercises

1. **Exercise:** Add a readList method to [ListOfNumbers.java](#). This method should read in int values from a file, print each value, and append them to the end of the vector. You should catch all appropriate errors. You will also need a text file containing numbers to read in.

**Answer:** See [ListOfNumbers2.java](#).

2. **Exercise:** Modify the following cat method so that it will compile:

```
public static void cat(File file) {  
    RandomAccessFile input = null;  
    String line = null;  
    try {  
        input = new RandomAccessFile(file, "r");  
        while ((line = input.readLine()) != null) {  
            System.out.println(line); }  
        return;  
    } finally {  
        if (input != null) {  
            input.close(); }  
    }  
}
```

**Answer:** The code to catch exceptions is shown in bold:

```
public static void cat(File file) {
    RandomAccessFile input = null;
    String line = null;
    try {
        input = new RandomAccessFile(file, "r");
        while ((line = input.readLine()) != null) {
            System.out.println(line);
        }
        return;
    } catch(FileNotFoundException fnf) {
        System.err.format("File: %s not found%n", file);
    } catch(IOException e) {
        System.err.println(e.toString());
    } finally {
        if (input != null) {
            try {
                input.close();
            } catch(IOException io) {
            }
        }
    }
}
```



## 17. Questions and Exercises: Basic I/O

### Questions

1. What class and method would you use to read a few pieces of data that are at known positions near the end of a large file?
2. When invoking `format`, what is the best way to indicate a new line?
3. How would you determine the MIME type of a file?
4. What method(s) would you use to determine whether a file is a symbolic link?

### Exercises

1. Write an example that counts the number of times a particular character, such as `e`, appears in a file. The character can be specified at the command line. You can use [xanadu.txt](#) as the input file.
2. The file [datafile](#) begins with a single long that tells you the offset of a single `int` piece of data within the same file. Write a program that gets the `int` piece of data. What is the `int` data?

### Answers to Questions and Exercises: Basic I/O

#### Answers to Questions

Question 1. What class and method would you use to read a few pieces of data that are at known positions near the end of a large file?

Answer 1. `Files.newByteChannel` returns an instance of `SeekableByteChannel` which allows you to read from (or write to) any position in the file.

Question 2. When invoking `format`, what is the best way to indicate a new line?

Answer 2. Use the `%n` conversion — the `\n` escape is not platform independent!

Question 3. How would you determine the MIME type of a file?

Answer 3. The `Files.probeContentType` method uses the platform's underlying file type detector to evaluate and return the MIME type.

Question 4. What method(s) would you use to determine whether a file is a symbolic link?

Answer 4. You would use the `Files.isSymbolicLink` method.

#### Answers to Exercises

Exercise 1. Write an example that counts the number of times a particular character, such as `e`, appears in a file. The character can be specified at the command line. You can use [xanadu.txt](#) as the input file.

Answer 1. See [CountLetter.java](#) for the solution.

Exercise 2. The file [datafile](#) begins with a single long that tells you the offset of a single `int` piece of data within the same file. Write a program that gets the `int` piece of data. What is the `int` data?

Answer 2. 123. See [FindInt.java](#) for the solution.

## 18. Questions and Exercises: Collections, Interfaces

### Questions

1. At the beginning of this lesson, you learned that the core collection interfaces are organized into two distinct inheritance trees. One interface in particular is not considered to be a true Collection, and therefore sits at the top of its own tree. What is the name of this interface?
2. Each interface in the collections framework is declared with the <E> syntax, which tells you that it is generic. When you declare a Collection instance, what is the advantage of specifying the type of objects that it will contain?
3. What interface represents a collection that does not allow duplicate elements?
4. What interface forms the root of the collections hierarchy?
5. What interface represents an ordered collection that may contain duplicate elements?
6. What interface represents a collection that holds elements prior to processing?
7. What interface represents a type that maps keys to values?
8. What interface represents a double-ended queue?
9. Name three different ways to iterate over the elements of a List.
10. True or False: Aggregate operations are mutative operations that modify the underlying collection.

### Exercises

1. Write a program that prints its arguments in random order. Do not make a copy of the argument array. Demonstrate how to print out the elements using both streams and the traditional enhanced for statement.
2. Take the [FindDups](#) example and modify it to use a SortedSet instead of a Set. Specify a Comparator so that case is ignored when sorting and identifying set elements.
3. Write a method that takes a List<String> and applies [String.trim](#) to each element.
4. Consider the four core interfaces, Set, List, Queue, and Map. For each of the following four assignments, specify which of the four core interfaces is best-suited, and explain how to use it to implement the assignment.
  1. Whimsical Toys Inc (WTI) needs to record the names of all its employees. Every month, an employee will be chosen at random from these records to receive a free toy.
  2. WTI has decided that each new product will be named after an employee but only first names will be used, and each name will be used only once. Prepare a list of unique first names.
  3. WTI decides that it only wants to use the most popular names for its toys. Count up the number of employees who have each first name.
  4. WTI acquires season tickets for the local lacrosse team, to be shared by employees. Create a waiting list for this popular sport.

### Answers to Questions and Exercises:

#### Answers to Questions

1. Question: At the beginning of this lesson, you learned that the core collection interfaces are organized into two distinct inheritance trees. One interface in particular is not considered to be a true Collection, and therefore sits at the top of its own tree. What is the name of this interface?  
Answer: Map
2. Question: Each interface in the collections framework is declared with the <E> syntax, which tells you that it is generic. When you declare a Collection instance, what is the advantage of specifying the type of objects that it will contain?  
Answer: Specifying the type allows the compiler to verify (at compile time) that the type of object you put into the collection is correct, thus reducing errors at runtime.
3. Question: What interface represents a collection that does not allow duplicate elements?  
Answer: Set
4. Question: What interface forms the root of the collections hierarchy?  
Answer: Collection
5. Question: What interface represents an ordered collection that may contain duplicate elements?  
Answer: List
6. Question: What interface represents a collection that holds elements prior to processing?  
Answer: Queue
7. Question: What interface represents a type that maps keys to values?  
Answer: Map
8. Question: What interface represents a double-ended queue?  
Answer: Deque

9. Question: Name three different ways to iterate over the elements of a List.  
 Answer: You can iterate over a List using streams, the enhanced for statement, or iterators.
10. Question: True or False: Aggregate operations are mutative operations that modify the underlying collection.  
 Answer: False. Aggregate operations do not mutate the underlying collection. In fact, you must be careful to never mutate a collection while invoking its aggregate operations. Doing so could lead to concurrency problems should the stream be changed to a parallel stream at some point in the future.

### Answers to Exercises

1. Exercise: Write a program that prints its arguments in random order. Do not make a copy of the argument array. Demonstrate how to print out the elements using both streams and the traditional enhanced for statement.

Answer:

```
import java.util.*;
public class Ran {
    public static void main(String[] args) {
        // Get and shuffle the list of arguments
        List<String> argList = Arrays.asList(args);
        Collections.shuffle(argList);
        // Print out the elements using JDK 8 Streams
        argList.stream()
            .forEach(e->System.out.format("%s ",e));
        // Print out the elements using for-each
        for (String arg: argList) {
            System.out.format("%s ", arg); }
        System.out.println(); }
}
```

2. Exercise: Take the [FindDups](#) example and modify it to use a SortedSet instead of a Set. Specify a Comparator so that case is ignored when sorting and identifying set elements.

Answer:

```
import java.util.*;
public class FindDups {
    public static void main(String[] args) {
        Set<String> s = new HashSet<String>();
        for (String a : args)
            s.add(a);
        System.out.println(s.size() + " distinct words: " + s); }}
```

3. Exercise: Write a method that takes a List<String> and applies [String.trim](#) to each element.

Answer:

The enhanced for statement does not allow you to modify the List. Using an instance of the Iterator class allows you to delete elements, but not replace an existing element or add a new one. That leaves ListIterator:

```
import java.util.*;
public class ListTrim {
    static void listTrim(List<String> strings) {
        for (ListIterator<String> lit = strings.listIterator(); lit.hasNext(); ) {
            lit.set(lit.next().trim()); }
    }
    public static void main(String[] args) {
        List<String> l = Arrays.asList(" red ", " white ", " blue ");
        listTrim(l);
        for (String s : l) {
            System.out.format("\'%s\'%n", s); }
    }
}
```

4. Exercise: Consider the four core interfaces, Set, List, Queue, and Map. For each of the following four assignments, specify which of the four core interfaces is best-suited, and explain how to use it to implement the assignment.

Answers:

- Whimsical Toys Inc (WTI) needs to record the names of all its employees. Every month, an employee will be chosen at random from these records to receive a free toy.  
Use a List. Choose a random employee by picking a number between 0 and `size()-1`.
- WTI has decided that each new product will be named after an employee — but only first names will be used, and each name will be used only once. Prepare a list of unique first names.  
Use a Set. Collections that implement this interface don't allow the same element to be entered more than once.
- WTI decides that it only wants to use the most popular names for its toys. Count up the number of employees who have each first name.  
Use a Map, where the keys are first names, and each value is a count of the number of employees with that first name.
- WTI acquires season tickets for the local lacrosse team, to be shared by employees. Create a waiting list for this popular sport.  
Use a Queue. Invoke `add()` to add employees to the waiting list, and `remove()` to remove them.

## 19. Questions and Exercises: Collections, Implementations

### Questions

1. You plan to write a program that uses several basic collection interfaces: Set, List, Queue, and Map. You're not sure which implementations will work best, so you decide to use general-purpose implementations until you get a better idea how your program will work in the real world. Which implementations are these?
2. If you need a Set implementation that provides value-ordered iteration, which class should you use?
3. Which class do you use to access wrapper implementations?

### Exercises

1. Write a program that reads a text file, specified by the first command line argument, into a List. The program should then print random lines from the file, the number of lines printed to be specified by the second command line argument. Write the program so that a correctly-sized collection is allocated all at once, instead of being gradually expanded as the file is read in. Hint: To determine the number of lines in the file, use [java.io.File.length](#) to obtain the size of the file, then divide by an assumed size of an average line.

### Answers to Questions and Exercises:

#### Answers to Questions

1. Question: You plan to write a program that uses several basic collection interfaces: Set, List, Queue, and Map. You're not sure which implementations will work best, so you decide to use general-purpose implementations until you get a better idea how your program will work in the real world. Which implementations are these?

Answer:

Set: HashSet

List: ArrayList

Queue: LinkedList

Map: HashMap

2. Question: If you need a Set implementation that provides value-ordered iteration, which class should you use?

Answer:

TreeSet guarantees that the sorted set is in ascending element order, sorted according to the natural order of the elements or by the Comparator provided.

3. Question: Which class do you use to access wrapper implementations?

Answer:

You use the Collections class, which provides static methods that operate on or return collections.

#### Answers to Exercises

1. Exercise: Write a program that reads a text file, specified by the first command line argument, into a List. The program should then print random lines from the file, the number of lines printed to be specified by the second command line argument. Write the program so that a correctly-sized collection is allocated all at once, instead of being gradually expanded as the file is read in. Hint: To determine the number of lines in the file, use [java.io.File.length](#) to obtain the size of the file, then divide by an assumed size of an average line.

Answer:

Since we are accessing the List randomly, we will use ArrayList. We estimate the number of lines by taking the file size and dividing by 50. We then double that figure, since it is more efficient to overestimate than to underestimate.

```

import java.util.*;
import java.io.*;

public class FileList {
    public static void main(String[] args) {
        final int assumedLineLength = 50;
        File file = new File(args[0]);
        List<String> fileList =
            new ArrayList<String>((int)(file.length() / assumedLineLength) * 2);
        BufferedReader reader = null;
        int lineCount = 0;
        try {
            reader = new BufferedReader(new FileReader(file));
            for (String line = reader.readLine(); line != null;
                line = reader.readLine()) {
                fileList.add(line);
                lineCount++;
            }
        } catch (IOException e) {
            System.err.format("Could not read %s: %s%n", file, e);
            System.exit(1);
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {}
            }
        }
        int repeats = Integer.parseInt(args[1]);
        Random random = new Random();
        for (int i = 0; i < repeats; i++) {
            System.out.format("%d: %s%n", i,
                fileList.get(random.nextInt(lineCount - 1)));
        }
    }
}

```

This program actually spends most of its time reading in the file, so pre-allocating the ArrayList has little affect on its performance. Specifying an initial capacity in advance is more likely to be useful when your program repeatly creates large ArrayList objects without intervening I/O.

## 20. Questions and Exercises: Using Swing Components

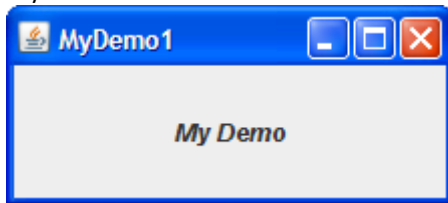
Use the information in this lesson and the component [how-to sections](#) to help you complete these questions and exercises.

### Questions

- Find the component that best fits each of the following needs. Write down both the component's common name (such as "frame") and find the component's how-to page online.
  - A component that lets the user pick a color.
  - A component that displays an icon, but that doesn't react to user
  - A component that looks like a button and that, when pressed, brings up a menu of items for the user to choose from.
  - A container that looks like a frame, but that appears (usually with other, similar containers) within a real frame.
  - A container that lets the user determine how two components share a limited amount of space.
- Which method do you use to add a menu bar to a top-level container such as a JFrame?
- Which method do you use to specify the default button for a top-level container such as a JFrame or JDialog?
- Which method do you use to enable and disable components such as JButtons? What class is it defined in?
- Which Swing components use [ListSelectionModel](#)? [Hint: The "Use" link at the top of the specification for each interface and class takes you to a page showing where in the API that interface or class is referenced.]
  - Do those components use any other models to handle other aspects of the components' state? If so, list the other models' types
- Which type of model holds a text component's content?

### Exercises

- Implement a program with a GUI that looks like the one shown below. Put the main method in a class named MyDemo1.



- Make a copy of MyDemo1.java named MyDemo2.java. Add a menu bar to MyDemo2.
- Copy MyDemo1.java to MyDemo3.java. Add a button (JButton) to MyDemo3.java. Make it the default button.

### Answers: Using Swing Components

Use the information in this lesson and the component [how-to sections](#) to help you complete these questions and exercises.

#### Answers to Questions

**Question 1:** Find the component that best fits each of the following needs. Write down both the component's common name (such as "frame") and find the component's how-to page online.

**Question 1a:** A component that lets the user pick a color.

**Answer 1a:** [color chooser](#)

**Question 1b:** A component that displays an icon, but that doesn't react to user clicks.

**Answer 1b:** [label](#)

**Question 1c:** A component that looks like a button and that, when pressed, brings up a menu of items for the user to choose from.

**Answer 1c:** [uneditable combo box](#)

**Question 1d:** A container that looks like a frame, but that appears (usually with other, similar containers) within a real frame.

**Answer 1d:** [internal frame](#)

**Question 1e:** A container that lets the user determine how two components share a limited amount of space.

**Answer 1e:** [split pane](#)

**Question 2:** Which method do you use to add a menu bar to a top-level container such as a JFrame?

**Answer 2:** [setJMenuBar](#)

**Question 3:** Which method do you use to specify the default button for a top-level container such as a JFrame or JDialog?

**Answer 3:** JRootPane's [setDefaultButton](#) method. (You get the top-level container's root pane using the [getRootPane](#) method defined by the RootPaneContainer interface, which every top-level container implements.)

**Question 4:** Which method do you use to enable and disable components such as JButtons? What class is it defined in?

**Answer 4:** [setEnabled](#), which is defined in the Component class

**Question 5a:** Which Swing components use [ListSelectionModel](#)? [Hint: The "Use" link at the top of the specification for each interface and class takes you to a page showing where in the API that interface or class is referenced.]

**Answer 5a:** [JList](#) and [JTable](#)

**Question 5b:** Do those components use any other models to handle other aspects of the components' state? If so, list the other models' types.

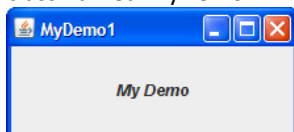
**Answer 5b:** JList also uses a ListModel, which holds the list's data. JTable uses a TableModel to hold its data and a TableColumnModel to manage the table's columns.

**Question 6:** Which type of model holds a text component's content?

**Answer 6:** [Document](#)

## Answers to Exercises

**Exercise 1.** Implement a program with a GUI that looks like the one shown below. Put the main method in a class named MyDemo1.



**Answer 1:** See [MyDemo1.java](#). Here's the code that adds the bold, italicized text:

```
JLabel label = new JLabel("My Demo");
frame.getContentPane().add(BorderLayout.CENTER, label);
label.setFont(label.getFont().deriveFont(Font.ITALIC | Font.BOLD));
label.setHorizontalAlignment(JLabel.CENTER)
```

**Exercise 2.** Make a copy of MyDemo1.java named MyDemo2.java. Add a menu bar to MyDemo2.

**Answer 2:** See [MyDemo2.java](#). The menu bar can be implemented with this code:

```
JMenu menu = new JMenu("Menu");
JMenuBar mb = new JMenuBar();
mb.add(menu);
frame.setJMenuBar(mb);
```

**Exercise 3.** Copy MyDemo1.java to MyDemo3.java. Add a button (JButton) to MyDemo3.java. Make it the default button.

**Answer 3:** See [MyDemo3.java](#). Here's the code that adds the button and makes it the default button:

```
JButton b = new JButton("A button");
frame.getContentPane().add(BorderLayout.PAGE_END, b);
frame.getRootPane().setDefaultButton(b);
```

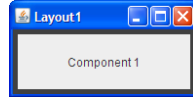
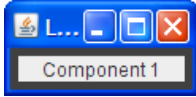


## 21. Questions and Exercises: Laying Out Components within a Container

### Questions

In each of the following questions, choose the layout manager(s) most naturally suited for the described layout. Assume that the container controlled by the layout manager is a JPanel.

1. The container has one component that should take up as much space as possible



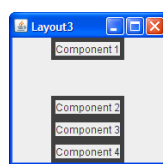
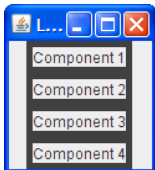
- a. BorderLayout
- b. GridLayout
- c. GridBagLayout
- d. a and b
- e. b and c

2. The container has a row of components that should all be displayed at the same size, filling the container's entire area.



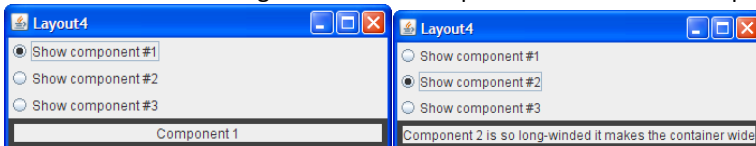
- a. FlowLayout
- b. GridLayout
- c. BoxLayout
- d. a and b

3. The container displays a number of components in a column, with any extra space going between the first two components.



- a. FlowLayout
- b. BoxLayout
- c. GridLayout
- d. BorderLayout

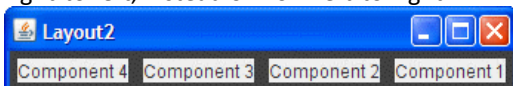
4. The container can display three completely different components at different times, depending perhaps on user input or program state. Even if the components' sizes differ, switching from one component to the next shouldn't change the amount of space devoted to the component.



- a. SpringLayout
- b. BoxLayout
- c. CardLayout
- d. GridBagLayout

### Exercises

1. Implement the layout described and shown in question 1.
2. Implement the layout described and shown in question 2.
3. Implement the layout described and shown in question 3.
4. Implement the layout described and shown in question 4.
5. By adding a single line of code, make the program you wrote for Exercise 2 display the components from right-to-left, instead of from left-to-right.



## Answers: Laying Out Components within a Container

### Answers to Questions

In each of the following questions, choose the layout manager(s) most naturally suited for the described layout. Assume that the container controlled by the layout manager is a JPanel.

**Question 1.** The container has one component that should take up as much space as possible



- a. BorderLayout
- b. GridLayout
- c. GridBagLayout
- d. a and b
- e. b and c

**Answer 1:** d. BorderLayout and GridLayout easily deal with this situation. Although you could use GridBagLayout, it's much more complex than necessary.

**Question 2.** The container has a row of components that should all be displayed at the same size, filling the container's entire area.



- a. FlowLayout
- b. GridLayout
- c. BoxLayout
- d. a and b

**Answer 2:** b. This type of same-size layout — whether in a row, a column, or a grid — is what GridLayout is best at.

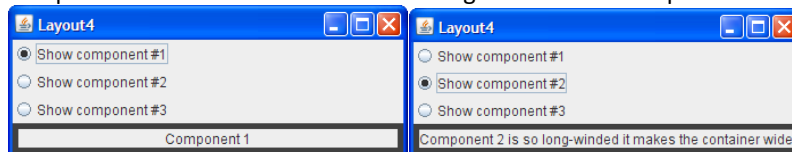
**Question 3.** The container displays a number of components in a column, with any extra space going between the first two components.



- a. FlowLayout
- b. BoxLayout
- c. GridLayout
- d. BorderLayout

**Answer 3:** b. BoxLayout lays out components in either a column or a row. You can specify extra space using an invisible component.

**Question 4.** The container can display three completely different components at different times, depending perhaps on user input or program state. Even if the components' sizes differ, switching from one component to the next shouldn't change the amount of space devoted to the component.



- a. SpringLayout
- b. BoxLayout
- c. CardLayout
- d. GridBagLayout

**Answer 4:** c. CardLayout exists to allow components to share the same space. Although it's simpler to use a JTabbedPane component to control an area, CardLayout is the solution when you don't want tabs.

### Answers to Exercises

**Exercise 1.** Implement the layout described and shown in question 1.

**Answer 1:** See [Layout1.java](#). Here's the code that implements the layout:

```
JPanel p = new JPanel(new BorderLayout());
p.add(createComponent("Component 1"),
      BorderLayout.CENTER);
frame.setContentPane(p);
```

**Exercise 2.** Implement the layout described and shown in question 2.

**Answer 2:** See [Layout2.java](#). Here's the code that implements the layout:

```
JPanel p = new JPanel(new GridLayout(1,0));
p.add(createComponent("Component 1"));
p.add(createComponent("Component 2"));
p.add(createComponent("Component 3"));
p.add(createComponent("Component 4"));
frame.setContentPane(p);
```

**Exercise 3.** Implement the layout described and shown in question 3.

**Answer 3:** See [Layout3.java](#). Here's the code that implements the layout:

```
JPanel p = new JPanel();
p.setLayout(new BoxLayout(p, BoxLayout.PAGE_AXIS));
p.add(createComponent("Component 1"));
p.add(Box.createVerticalGlue());
p.add(createComponent("Component 2"));
p.add(createComponent("Component 3"));
p.add(createComponent("Component 4"));
frame.setContentPane(p);
```

**Exercise 4.** Implement the layout described and shown in question 4.

**Answer 4:** See [Layout4.java](#). Here's the code that implements the layout:

*...//Where the radio buttons are set up:*

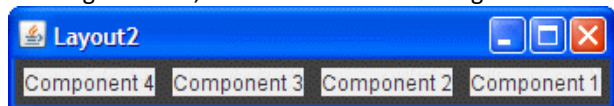
```
for (int i = 0; i < strings.length; i++) {
    ...
    rb[i].setActionCommand(String.valueOf(i));
    ...
}
```

*...//Where the panel to contain the shared-space components is set up:*

```
cards = new JPanel(new CardLayout());
for (int i = 0; i < strings.length; i++) {
    cards.add(createComponent(strings[i]), String.valueOf(i));}
...//In the action listener for the radio buttons:
```

```
public void actionPerformed(ActionEvent evt) {
    CardLayout cl = (CardLayout)(cards.getLayout());
    cl.show(cards, (String)evt.getActionCommand());}
```

**Exercise 5.** By adding a single line of code, make the program you wrote for Exercise 2 display the components from right-to-left, instead of from left-to-right.



**Answer 5:** You can change the horizontal orientation using the [setComponentOrientation](#) method defined by the Component class. For example:

```
p.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
```

