STRUMIENIE DANYCH, SERIALIZACJA OBIEKTÓW

1. Procedura korzystania ze strumieni danych

Aby utworzyć plik:

1) Należy utworzyć obiekt (np. typu *FileOutputStream*), powiązany ze plikiem danych binarnych (np. "plik3.dat");
   *FileOutputStream plik = **new** FileOutputStream("plik3.dat");*

2) W celu buforowania bajtów pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileOutputStream* należy utworzyć obiekt klasy *BufferedOutputStream*

   *BufferedOutputStream bufor = **new** BufferedOutputStream(plik);*

3) W celu reprezentowania danych typu ***boolean**, byte, **double**, **float long**, **short*** należy utworzyć strumień danych typu *DataOutputStream* powiązanego z obiektem buforującym typu *BufferedOutputStream*

   *DataOutputStream dana= **new** DataOutputStream (bufor);*

Dalej podano metody strumienia danych do zapisu danych do pliku:

4) Pojedyncze bajty mogą być zapisywane do pliku za pomocą metody:

   ***void** write(**int** b)*

5) Całe ciągi bajtów mogą być zapisywane do pliku za pomocą metody:

   ***void** write(**byte[]** cbuf, **int** off, **int** len)* – metoda, która czyta z tablicy *cbuf* od indeksu *off* liczbę *len* bajtów i zapisuje do pliku

6) **void** writeBoolean(**boolean** v) – zapisuje do pliku 1-bajtową wartość

7) **void** writeByte(**int** v) – zapisuje do pliku 1-bajtową wartość

8) **void** writeChar(**int** v) – zapisuje znak jako 2-bajtową wartość –starszy bajt jako pierwszy (Unicode)

9) **void** writeDouble(**double** v) – zapisuje 8-bajtową wartość do pliku

10) **void** writeFloat(**float** v) – zapisuje 4-bajtową wartość do pliku

11) **void** writeInt(**int** v) – zapisuje 4 bajty do pliku

12) **void** writeLong(**long** v) – zapisuje 8 bajtów do pliku

13) **void** writeShort(**int** v) – zapisuje 2 bajty do pliku

Aby odczytać plik:

14) Należy utworzyć obiekt (np. typu *FileInputStream*), powiązany ze plikiem danych binarnych (np. "plik3.dat");

*FileInputStream plik = **new** FileInputStream("plik3.dat");*

15) W celu buforowania bajtów pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileInputStream* należy utworzyć obiekt klasy *BufferedInputStream*

*BufferedInputStream bufor = **new** BufferedInputStream (plik);*

16) W celu reprezentowania danych typu ***boolean**, byte, **double**, **float long**, **short*** należy utworzyć strumień danych typu *DataInputStream* powiązanego z obiektem buforującym typu *BufferedInputStream*

*DataInputStream dana= **new** DataInputStream (bufor);*

Dalej podano metody strumienia danych do odczytu danych z pliku:

17) Pojedyncze bajty mogą być odczytywane z pliku za pomocą metody:

***int** read()*

18) Całe ciągi bajtów mogą być odczytywane z pliku za pomocą metody:

***int** read(**byte[]** cbuf, **int** off, **int** len)* – metoda, która czyta plik i zapisuje do tablicy *cbuf* od indeksu *off* liczbę *len* bajtów i zwraca przez **return** liczbę faktycznie odczytanych bajtów

19) **boolean** readBoolean() – czyta z pliku 1 bajt i wraca wartość true lub false

20) **byte** readByte() – czyta z pliku 1 bajt i zwraca wartość typu byte

21) **char** readChar() – czyta 1 znak (2 bajty) i zwraca 1 znak (Unicode)

22) **double** readDouble() – czyta 8 bajtów z pliku i zwraca wartość **double**

23) **float** readFloat() – czyta 4 bajtów z pliku i zwraca wartość **float**

24) **int** readInt() – czyta 4 bajty z pliku i zwraca wartość typu **int**

25) **long** readLong() – czyta 8 bajtów z pliku i zwraca wartość typu **long**

26) **short** readShort() – czyta 2 bajty z pliku i zwraca wartość typu **short**

27) Po zapisie i odczycie strumień danych należy zamknąć metodą *close()*

```
import java.io.*;//zapis za pomocą writeInt i odczyt za pomocą readInt – z klawiatury są podawane 1-bajtowe wartości
import java.util.*;
public class WEWY3
 {   static byte weByte()
      {  InputStreamReader wejscie = new InputStreamReader( System.in );
         BufferedReader bufor = new BufferedReader( wejscie );
         StringTokenizer zeton;
         try
            {  zeton = new StringTokenizer(bufor.readLine());
               return Byte.parseByte(zeton.nextToken()); }
         catch (IOException e)
            {  System.err.println("Blad IO byte "+e); return 0;  }
         catch (NumberFormatException e)
            {  System.err.println( "Blad formatu byte "+e); return 0;   }
       }
 static void Zapiszplik3()
   { byte dane=0;
     try
      {  FileOutputStream plik = new FileOutputStream ("plik3.dat");
         BufferedOutputStream bufor = new BufferedOutputStream (plik);
         DataOutputStream dana= new DataOutputStream (bufor);
         while (dane!=-1)
           {  System.out.print("Podaj dane: "); dane=weByte(); //odczyt wartości 1-bajtowej
              if (dane!=-1)   dana.writeByte(dane);  }       //zapis do pliku 1 bajtu
         dana.close();
       }  catch (IOException e)
            {  System.out.println ("Blad zapisu pliku bajtowego"+e);   }
     }
 static void Odczytajplik3()
   { byte dane=0;
     try
      {  FileInputStream plik = new FileInputStream ("plik3.dat");
         BufferedInputStream bufor = new BufferedInputStream (plik);
         DataInputStream dana= new DataInputStream (bufor);
          try
           {  while (true)
              { dane=dana.readByte();    System.out.print(dane);}//odczyt z pliku 1 bajtu
           } catch  (EOFException eof)
              { bufor.close();  } //zamknięcie bufora przez obsługę wyjątku od czytania poza plikiem
          System.out.println();
         } catch (IOException e)
          {  System.out.println ("Blad odczytu pliku bajtowego"+e); }
   }
 public static void main(String[] args)
   {Zapiszplik3(); Odczytajplik3();  } }
```

```
import java.io.*; //zapis za pomocą writeInt i odczyt za pomocą readInt – z klawiatury są podawane 4-bajtowe wartości
import java.util.*;
public class WEWY3_
{ static int weInt()
   { InputStreamReader wejscie = new InputStreamReader( System.in );
     BufferedReader bufor = new BufferedReader( wejscie );
     StringTokenizer zeton;
     try
       { zeton = new StringTokenizer(bufor.readLine());
         return Integer.parseInt(zeton.nextToken());  }
     catch (IOException e)
       {  System.err.println("Blad IO int "+e);   return 0;        }
     catch (NumberFormatException e)
       {  System.err.println( "Blad formatu int "+e);   return 0;   }
    }
 static void Zapiszplik3_()
   { int dane=0;
     try
       {  FileOutputStream plik =  new FileOutputStream ("plik2.dat");
          BufferedOutputStream bufor = new BufferedOutputStream (plik);
          DataOutputStream dana= new DataOutputStream (bufor);
          while (dane!=-1)
              { System.out.print("Podaj dane: "); dane=weInt(); //odczyt wartości 4-bajtowej
                if (dane!=-1)   dana.writeInt(dane); }  //zapis do pliku 4 bajtów
          dana.close();
       } catch (IOException e)
              {  System.out.println ("Blad zapisu pliku bajtowego"+e);   }
   }
static void Odczytajplik3_()
   { int dane=0;
     try
      {  FileInputStream plik = new FileInputStream ("plik2.dat");
         BufferedInputStream bufor = new BufferedInputStream (plik);
         DataInputStream dana= new DataInputStream (bufor);
         try
          { while (true)
              { dane=dana.readInt();     System.out.print(dane); } //odczyt z pliku 4 bajtów
           } catch  (EOFException eof)
                { bufor.close(); }      //zamknięcie bufora przez obsługę wyjątku od czytania poza plikiem
        System.out.println();
      } catch (IOException e)
         {  System.out.println ("Blad odczytu pliku bajtowego"+e);  }
     }
  public static void main(String[] args)
   { Zapiszplik3_();  Odczytajplik3_();   } }
```

2) Serializacja obiektów

- Jest to mechanizm szeregowego zapisu do pliku związanego ze strumieniem wyjściowym ciągu bajtów po wykonaniu konwersji obiektu do postaci szeregowej i
- Odczytu szeregowego ciągu bajtów z pliku związanego ze strumieniem wejściowym i dokonanie konwersji do postaci danej (obiektu, typu podstawowego: **int**, **float** itp.)
- Mechanizm ten pozwala zachować całe obiekty w pliku po zakończeniu programu
- Obiekty zapisywane do pliku muszą implementować pusty interfejs *Serializable* (obiekty są serializowane)
- Obiekty z zagnieżdżonymi obiektami są w całości zapisywane do pliku pod warunkiem, że zagnieżdżone obiekty też są serializowane
- Obiekty zagnieżdżone w serializowanych klasach mogą być pomijane przy zapisie do strumienia, jeśli to konieczne, za pomocą słowa kluczowego **transient**
  Np. **public transient** String s = ”Kowalski”;

Procedura korzystania ze strumieni obiektowych powiązanych z plikami binarymi

Aby utworzyć plik:

1) Należy utworzyć obiekt (np. typu *FileOutputStream*), powiązany ze plikiem danych binarnych (np. "Wiadomosc.obj");
   FileOutputStream plikobiektow = **new** FileOutputStream("Wiadomosc.obj");
2) W celu utworzenia wyjściowego strumienia obiektowego powiązanego z obiektem związanym ze źródłem danych np. typu *FileOutputStream* należy utworzyć obiekt klasy *ObjectOutputStream*
   ObjectOutputStream strumienobiektow=**new** BufferedOutputStream (plikobiektow);

3) obiekty dziedziczące po *Object* i implementujace interfejs *Serializable* są zapisywane do pliku w postaci szeregowej za pomocą metody
   void writeObject(Object ob)

Dalej podano część metod strumienia obiektów do zapisu różnych danych do pliku:

4) Pojedyncze bajty mogą być zapisywane do pliku za pomocą metody:
   void write(**int** b)
5) Całe ciągi bajtów mogą być zapisywane do pliku za pomocą metody:
   void write(**byte**[]cbuf) – metoda, która czyta zawartość tablicy bajtów (cbuf.length) i zapisuje ją do strumienia
6) Całe ciągi bajtów mogą być zapisywane do pliku za pomocą metody:
   void write(**byte[]** cbuf, **int** off, **int** len) – metoda, która czyta z tablicy *cbuf* od indeksu *off* liczbę *len* bajtów i zapisuje do pliku
7) **void** writeBoolean(**boolean** v) – zapisuje do pliku 1-bajtową wartość
8) **void** writeByte(**int** v) – zapisuje do pliku 1-bajtową wartość
9) **void** writeChar(**int** v) – zapisuje znak jako 2-bajtową wartość – pierwszy starszy bajt (Unicode)
10) **void** writeDouble(**double** v) – zapisuje 8-bajtową wartość do pliku
11) **void** writeFloat(**float** v) – zapisuje 4-bajtową wartość do pliku
12) **void** writeInt(**int** v) – zapisuje 4 bajty do pliku
13) **void** writeLong(**long** v) – zapisuje 8 bajtów do pliku
14) **void** writeShort(**int** v) – zapisuje 2 bajty do pliku

Aby odczytać plik:

1) Należy utworzyć obiekt (np. typu *FileInputStream*), powiązany ze plikiem danych binarnych (np. "Wiadomosc.obj");

```
FileInputStream plik = new FileInputStream("Wiadomosc.obj");
```

2) W celu odczytu obiektow pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileInputStream* należy utworzyć obiekt klasy *ObjectInputStream*

```
ObjectInputStream bufor = new ObjectInputStream (plik);
```

3) Odczytu obiektów z strumienia należy wykonać za pomocą metody

```
Object readObject()
```

Dalej podano niektóre metody strumienia obiektów do odczytu danych z pliku:

4) Pojedyncze bajty mogą być odczytywane z pliku za pomocą metody:

   int *read()*

5) Całe ciągi bajtów mogą być odczytywane z pliku za pomocą metody:

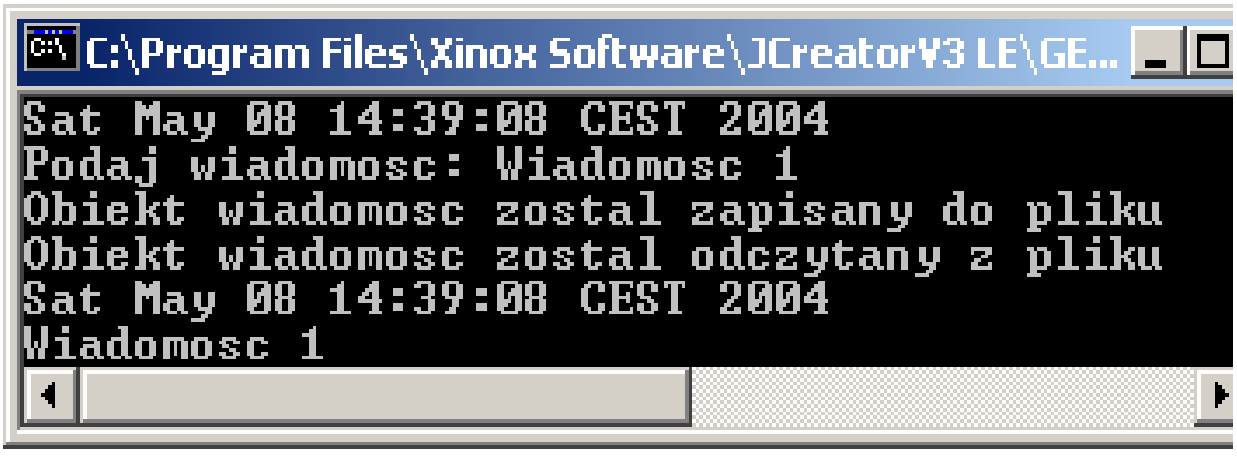
   int *read(**byte[]** cbuf, **int** off, **int** len)* – metoda, która czyta plik i zapisuje do tablicy *cbuf* od indeksu *off* liczbę *len* bajtów i zwraca przez **return** liczbę faktycznie odczytanych bajtów

6) **boolean** readBoolean() – czyta z pliku 1 bajt i wraca wartość true lub false
7) **byte** readByte() – czyta z pliku 1 bajt i zwraca wartość typu byte
8) **char** readChar() – czyta 1 znak (2 bajty - Unicode) i zwraca 1 znak
9) **double** readDouble() – czyta 8 bajtów z pliku i zwraca wartość **double**
10) **float** readFloat() – czyta 4 bajtów z pliku i zwraca wartość **float**
11) **int** readInt() – czyta 4 bajty z pliku i zwraca wartość typu **int**
12) **long** readLong() – czyta 8 bajtów z pliku i zwraca wartość typu **long**
13) **short** readShort() – czyta 2 bajty z pliku i zwraca wartość typu **short**
14) Po zapisie i odczycie strumień obiektów należy zamknąć metodą *close()*

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

 class Wiadomosc implements Serializable
  { String dane;
    Date data;
    static String weString()
     { InputStreamReader wejscie = new InputStreamReader( System.in );
       BufferedReader bufor = new BufferedReader( wejscie );
       System.out.print("Podaj wiadomosc: ");
       try
        { return bufor.readLine(); }
       catch (IOException e)
        { System.err.println("Blad IO String");
          return ""; }
    }
   public void zapiszWiadomosc()
    { data = new Date();
      System.out.println(data);
      dane =weString();}

   public void odczytajWiadomosc()
    { System.out.println(data);
      System.out.println(dane);}
}
```



```
public class p6_6
 {
  static void Zapiszobiektydopliku()
   {  Wiadomosc wiadomosc = new Wiadomosc();
      wiadomosc.zapiszWiadomosc();
      try
      { FileOutputStream plikobiektow = new FileOutputStream ("Wiadomosc.obj");
        ObjectOutputStream  strumienobiektow =
                                              new ObjectOutputStream (plikobiektow);
       strumienobiektow.writeObject(wiadomosc);
       strumienobiektow.close();
       System.out.println("Obiekt wiadomosc zostal zapisany do pliku");
       } catch (IOException e)
           { System.out.println ("Blad zapisu pliku obiektowego"+e);}
   }

static void Odczytajobiektyzpliku()
   { Wiadomosc wiadomosc = null;
     try
     { FileInputStream plikobiektow =  new FileInputStream ("Wiadomosc.obj");
       ObjectInputStream  strumienobiektow =
                                              new   ObjectInputStream (plikobiektow);
       wiadomosc = (Wiadomosc)strumienobiektow.readObject();
       System.out.println("Obiekt wiadomosc zostal odczytany z pliku");
       if (wiadomosc != null)
          wiadomosc.odczytajWiadomosc();
       strumienobiektow.close();
      } catch (Exception e)
        { System.out.println ("Blad odczytu pliku obiektowego"+e);        }
   }

  public static void main(String[] args)
   { Zapiszobiektydopliku();
     Odczytajobiektyzpliku();
   }
 }
```