STRUMIENIE TEKSTOWE WEJŚCIOWE – WPROWADZANIE DANYCH
STRUMIENIE BAJTOWE, STRUMIENIE TEKSTOWE

1) Strumienie tekstowe wejściowe – wprowadzanie danych

Procedura korzystania ze strumieni tekstowych wejściowych powiązanych ze standardowym wejściem (konsola)

1) Należy utworzyć obiekt (np. typu *InputStreamReader*), powiązany ze źródłem danych (np. *System.in*)

   Przykład
   *InputStreamReader wejscie = **new** InputStreamReader(System.in);*

2) W celu buforowania znaków pochodzących z obiektu powiązanego ze źródłem danych np. typu *InputStreamReader* należy utworzyć obiekt klasy *BufferedReader*

   Przykład
   *BufferedReader bufor = **new** BufferedReader(wejscie);*
   lub
   *BufferedReader bufor = **new** BufferedReader(wejscie, **int**);* //**int**-rozmiar bufora

3) Znaki mogą być odczytywane metodami obiektu buforującego:
   ***int** read()*
   *np. **int** dana = bufor.read()*
4) Całe ciągi znaków mogą być odczytane za pomocą:
   ***int** read(**char**[] cbuf, **int** off, **int** len)* – metoda, która czyta do tablicy *cbuf* od indeksu *off* liczbę *len* znaków i zwraca przez **return** liczbę faktycznie odczytanych znaków
5) Całe wiersze mogą być odczytywane za pomocą metody obiektu buforującego:
   String readLine()
   np. String dana = bufor.readLine(),

   która czyta łańcuch typu String lub jego wartość **null**, gdy będzie osiągnięty koniec łańcucha (czyli koniec linii oznaczony znakami np.: '\n','\r')
6) Analiza odczytanego łańcucha może być przeprowadzona za pomocą obiektu klasy typu *StringTokenizer*

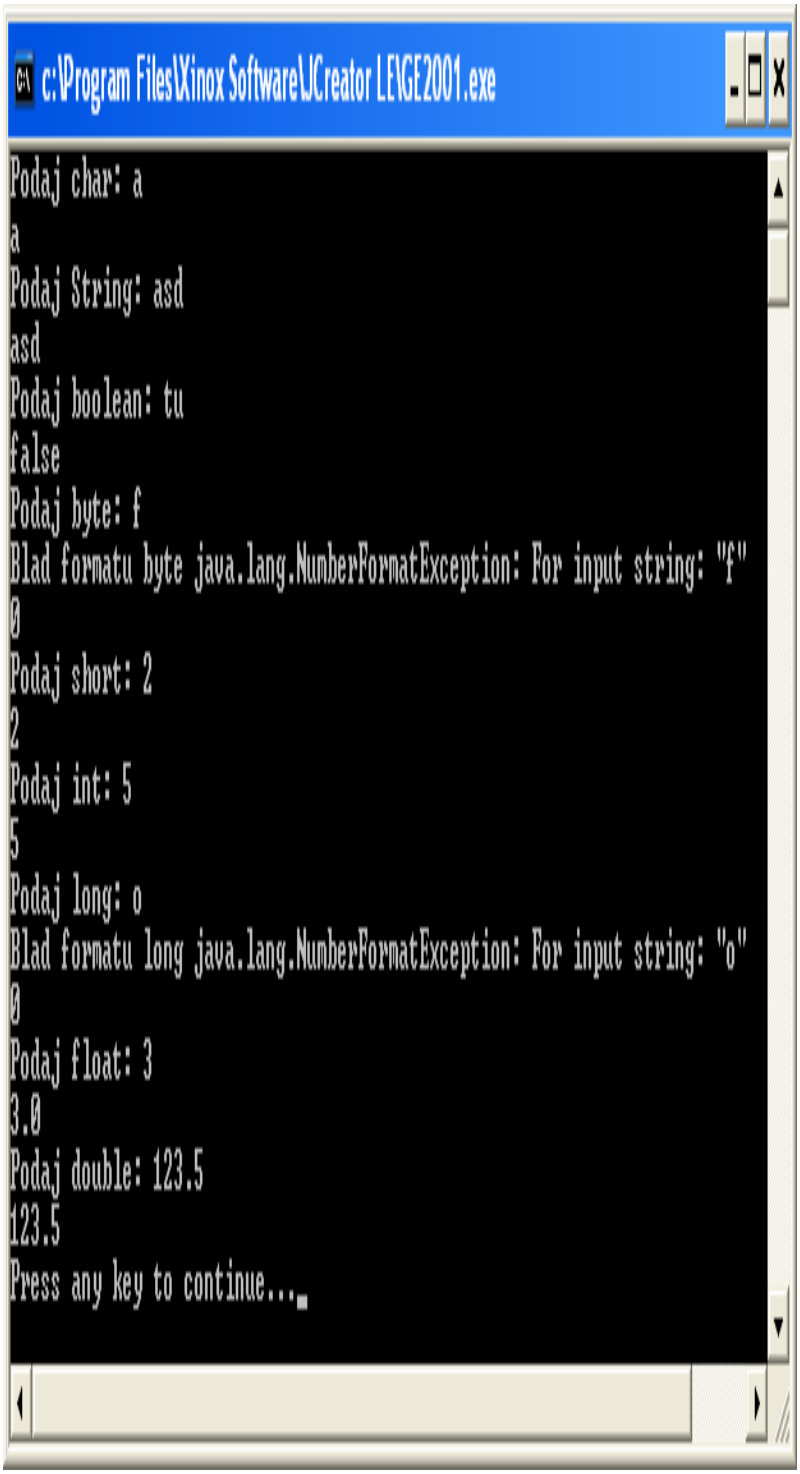
   *StringTokenizer bon = **new** StringTokenizer(bufor.readLine());*
7) Innym sposobem jest analiza odczytanego łańcucha typu *String* bezpośrednio za pomocą metod klasy *String*

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

public class WEWY                    //plik WEWY.java
 {
            //wejściowy strumień tekstowy odczytuje strumien stdin (System.in)
 static InputStreamReader wejscie = new InputStreamReader( System.in );
                  //bufor – klasa odczytuje wejściowy strumień znakowy wejscie i
                  // przechowuje odczytywane znaki w buforze
  static BufferedReader bufor = new BufferedReader( wejscie );
            //klasa do analizy składniowej jednostek lerksykalnych tzw. leksemów
            //(tokens) pobieranych metodą nextToken()
  StringTokenizer bon;

  boolean weBoolean()
   { try
     { bon = new StringTokenizer(bufor.readLine());
      return new Boolean(bon.nextToken()).booleanValue();
      }
    catch (IOException e)
     { System.err.println("Blad IO Boolean "+e);
      return false;  }
  }

  char weChar()
   { try
     { String s = bufor.readLine();
      return s.charAt(0);
      }
     catch (IOException e)
      { System.err.println("Blad IO char "+e);
       return 0;  }
   }

  String weString()
   { try
     { return bufor.readLine(); }
     catch (IOException e)
      { System.err.println("Blad IO String");
       return ""; }
   }
```

```
 byte weByte()
 { try
    { bon = new StringTokenizer(bufor.readLine());
     return Byte.parseByte(bon.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;   }
 }

 short weShort()
  { try
     { bon = new StringTokenizer(bufor.readLine());
      return Short.parseShort(bon.nextToken());
     }
    catch (IOException e)
     { System.err.println("Blad IO short: "+e);
        return 0;   }
    catch (NumberFormatException e)
     { System.err.println( "Blad formatu short "+e);
       return 0;   }
  }

 int weInt()
  { try
     { bon = new StringTokenizer(bufor.readLine());
      return Integer.parseInt(bon.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;   }
}
```

```
long weLong()
   { try
     { bon = new StringTokenizer(bufor.readLine());
       return Long.parseLong(bon.nextToken());
      }
     catch (IOException e)
     { System.err.println("Blad IO "+e);
       return 0L; }
     catch (NumberFormatException e)
     { System.err.println( "Blad formatu long "+e);
       return 0L;   }
   }

 float weFloat()
   { try
     { bon = new StringTokenizer(bufor.readLine());
      return new Float(bon.nextToken()).floatValue();
     }
     catch (IOException e)
     { System.err.println("Blad IO float "+e);
      return 0.0F;  }
     catch (NumberFormatException e)
     { System.err.println( "Blad formatu float "+e);
      return 0.0F;   }
   }

  double weDouble()
  { try
    { bon = new StringTokenizer(bufor.readLine());
     return new Double(bon.nextToken()).doubleValue();
    }
    catch (IOException e)
    { System.err.println("Blad IO double "+e);
     return 0.0; }
    catch (NumberFormatException e)
    { System.err.println( "Blad formatu double "+e);
     return 0;   }
  }
```

```
public static void main (String args[])
 {
   WEWY we = new WEWY();
   System.out.print("Podaj char: ");
   System.out.println(we.weChar());

   System.out.print("Podaj String: ");
   System.out.println(we.weString());

   System.out.print("Podaj boolean: ");
   System.out.println(we.weBoolean());

   System.out.print("Podaj byte: ");
   System.out.println(we.weByte());

   System.out.print("Podaj short: ");
   System.out.println(we.weShort());

   System.out.print("Podaj int: ");
   System.out.println(we.weInt());

   System.out.print("Podaj long: ");
   System.out.println(we.weLong());

   System.out.print("Podaj float: ");
   System.out.println(we.weFloat());

   System.out.print("Podaj double: ");
   System.out.println(we.weDouble());
  }
}
```



2) Strumienie tekstowe, strumienie bajtowe

2. Strumienie tekstowe plikowe

2.1. Procedura korzystania ze strumieni tekstowych powiązanych z plikami tekstowymi

Aby utworzyć plik:

1) Należy utworzyć obiekt (np. typu *FileWriter*), powiązany ze plikiem danych tekstowych (np. "plik1.txt");
   *FileWriter plik = **new** FileWriter("plik1.txt");*

2) Znaki mogą być zapisywane do pliku za pomocą metody strumienia:
   void write(**int** c)

3) Całe ciągi znaków mogą być zapisywane do pliku za pomocą metody strumienia:
   ***void** write(**char[]** cbuf, **int** off, **int** len)* – metoda, która czyta z tablicy *cbuf* od indeksu *off* liczbę *len* znaków i zapisuje do pliku

4) Część łańcucha można zapisać do pliku za pomocą metody strumienia:
   void write(String str, **int** off, **int** len) – metoda, która czyta z łańcucha *str* od indeksu *off* liczbę *len* znaków i zapisuje do pliku

Aby odczytać plik:

5) Należy utworzyć obiekt (np. typu *FileReader*), powiązany ze plikiem danych tekstowych (np. "plik1.txt");

   *FileReader plik = **new** FileReader("plik1.txt");*

6) Znaki mogą być odczytywane metodą strumienia:
   ***int** read ();*
   np. ***int** dane = plik.read()*

7) Całe ciągi znaków mogą być odczytane za pomocą metody strumienia:
   ***int** read(**char**[] cbuf, **int** off, **int** len)* – metoda, która czyta z pliku i zapisuje do tablicy *cbuf* od indeksu *off* liczbę *len* znaków i zwraca przez **return** liczbę faktycznie odczytanych znaków

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

```
import java.io.*;
import java.util.*;
public class WEWY4
 {
   static String weString()                    //metoda pobiera z klawiatury łańcuchy
    { InputStreamReader wejscie = new InputStreamReader( System.in );
      BufferedReader bufor = new BufferedReader( wejscie );
      try
       {  return bufor.readLine(); }
      catch (IOException e)
       {  System.err.println("Blad IO String");  return ""; }
     }
    static void Zapiszplik4()                    //metoda zapisuje plik
     { String dane="1";
       try
        {  FileWriter plik =  new FileWriter ("plik1.txt");
           while (!dane.equals(""))
            { System.out.print("Podaj dane: ");
              dane=weString();
              if (!dane.equals(""))   plik.write(dane,0,dane.length());  }
           plik.close();
         } catch (IOException e)
             {  System.out.println ("Blad zapisu pliku tekstowego"+e);  }
       }
   static void Odczytajplik4()                   //metoda odczytuje plik
    {  int dane = 0;
       try
        {  FileReader plik = new FileReader ("plik1.txt");
           dane=plik.read();
           while (dane!=-1)
            { System.out.print((char)dane);
              dane=plik.read(); }
           plik.close();
           System.out.println();
         } catch (IOException e)
             {  System.out.println ("Blad odczytu pliku  tekstowego"+e);  }
     }

   public static void main(String[] args)
    {  Zapiszplik4();                 //metody typu static
       Odczytajplik4();  }
}
```

2.2. Procedura korzystania ze strumieni tekstowych buforowanych powiązanych z plikami tekstowymi

Aby utworzyć plik:

1) Należy utworzyć obiekt (np. typu *FileWriter*), powiązany ze plikiem danych tekstowych (np. "plik2.txt");
   *FileWriter plik = **new** FileWriter("plik2.txt");*

2) W celu buforowania znaków pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileWriter* należy utworzyć obiekt klasy *BufferedWriter*
   *BufferedWriter bufor = **new** BufferedWriter(plik);*

3) Znaki mogą być zapisywane do pliku za pomocą metody bufora:
   ***void** write(**int** c)*

4) Całe ciągi znaków mogą być zapisywane do pliku za pomocą metody bufora:

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

5) Część łańcucha można zapisać do pliku za pomocą metody bufora:

   ***void** write(String str, **int** off, **int** len)* – metoda, która czyta z łańcucha *str* od indeksu *off* liczbę *len* znaków i zapisuje do pliku

Aby odczytać plik:

6) Należy utworzyć obiekt (np. typu *FileReader*), powiązany ze plikiem danych tekstowych (np. "plik1.txt");
   *FileReader plik = **new** FileReader("plik2.txt");*

7) W celu buforowania znaków pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileWriter* należy utworzyć obiekt klasy *BufferedReader*
   *BufferedReader bufor = **new** BufferedReader(plik);*

8) Znaki mogą być odczytywane metodą bufora:
   ***int** read ();*
   np. ***int** dane = plik.read()*

9) Całe ciągi znaków mogą być odczytane za pomocą metody bufora:

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

10) Całe wiersze mogą być odczytywane za pomocą metody obiektu buforującego:
    String readLine()
    np. String dana = bufor.readLine(),

    która czyta w pliku łańcuch typu *String* lub jego wartość **null**, gdy będzie osiągnięty koniec łańcucha (czyli koniec linii oznaczony znakami np.: '\n','\r')

11) Po zapisie i odczycie *bufor* należy zamknąć metodą *close()*

```
import java.io.*;
import java.util.*;
public class WEWY5
 {
  static String weString()
   { InputStreamReader wejscie = new InputStreamReader( System.in );
     BufferedReader bufor = new BufferedReader( wejscie );
     try
      { return bufor.readLine(); }
     catch (IOException e)
      { System.err.println("Blad IO String");  return ""; }
   }
   static void Zapiszplik5()
    { String dane="1";
      try
       { FileWriter plik = new FileWriter ("plik2.txt");
         BufferedWriter bufor = new BufferedWriter (plik);
         while (!dane.equals(""))
          { System.out.print("Podaj dane: ");
            dane=weString();
            if (!dane.equals(""))
                bufor.write(dane, 0, dane.length()); }
          bufor.close();
         } catch (IOException e)
               {  System.out.println ("Blad zapisu pliku tekstowego"+e);  }
      }
    static void Odczytajplik5()
     { String dane="0";
       try
        { FileReader plik = new FileReader ("plik2.txt");
          BufferedReader bufor = new BufferedReader (plik);
          dane=bufor.readLine();
          while (dane!=null)
                { System.out.print(dane);
                  dane=bufor.readLine();  }
          bufor.close();
          System.out.println();
         } catch (IOException e)
             {  System.out.println ("Blad odczytu pliku tekstowego"+e); }
      }
   public static void main(String[] args)
   { Zapiszplik5();
     Odczytajplik5(); }
 }
```

3. Strumienie bajtowe plikowe

3.1. Procedura korzystania ze strumieni bajtowych powiązanych z plikami binarnymi

Aby utworzyć plik:

1) Należy utworzyć obiekt (np. typu *FileOutputStream*), powiązany ze plikiem danych binarnych (np. "plik1.dat");
   *FileOutputStream plik = **new** FileOutputStream("plik1.dat");*

2) Pojedyncze bajty mogą być zapisywane do pliku za pomocą metody strumienia:
   void write(**int** c)
3) Całe ciągi bajtów mogą być zapisywane do pliku za pomocą metody strumienia:
   ***void** write(**byte[]** cbuf)* – metoda, która czyta z tablicy *cbuf* od indeksu *off* liczbę *cbuf.length* bajtów i zapisuje do pliku
4) Całe ciągi bajtów mogą być zapisywane do pliku za pomocą metody strumienia:
   ***void** write(**byte[]** cbuf, **int** off, **int** len)* – metoda, która czyta z tablicy *cbuf* od indeksu *off* liczbę *len* bajtów i zapisuje do pliku

Aby odczytać plik:

5) Należy utworzyć obiekt (np. typu *FileInputStream*), powiązany ze plikiem danych binarnych (np. "plik1.dat");

   *FileInputStream plik = **new** FileInputStream("plik1.dat");*

6) Pojedyncze bajty mogą być odczytywane metodą strumienia:
   ***int** read ();*
   np. ***int** dane = plik.read()*
7) Całe ciągi bajtów mogą być odczytane za pomocą metody strumienia:
   ***int** read (**byte**[] cbuf)* – metoda, która czyta plik i zapisuje do tablicy *cbuf* liczbę *cbuf.length* bajtów i zwraca przez **return** liczbę faktycznie odczytanych bajtów
8) Całe ciągi bajtów mogą być odczytane za pomocą metody strumienia:
   ***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
9) Po zapisie i odczycie strumień należy zamknąć metodą *close()*

```
import java.io.*;
import java.util.*;
public class WEWY1
 {
   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 Zapiszplik1()
    { int dane=0;
      try
       {  FileOutputStream plik = new FileOutputStream ("plik1.dat");
          while (dane!=-1)
           { System.out.print("Podaj dane: ");
             dane=weByte();
             if (dane!=-1)   plik.write(dane);  }
          plik.close();
        }  catch (IOException e)
          {  System.out.println ("Blad zapisu pliku bajtowego"+e); }
      }
  static void Odczytajplik1()
    {  int dane=0;
       try
        {  FileInputStream plik = new FileInputStream ("plik1.dat");
           dane=plik.read();
           while (dane!=-1)
            { System.out.print(dane);
              dane=plik.read(); }
           System.out.println();
           plik.close();
        } catch (IOException e)
           {  System.out.println ("Blad odczytu pliku bajtowego"+e);   }
      }
  public static void main(String[] args)
    {  Zapiszplik1();  Odczytajplik1(); }
}
```

3.2. Procedura korzystania ze strumieni bajtowych buforowanych powiązanych z plikami binarnymi

Aby utworzyć plik:

1) Należy utworzyć obiekt (np. typu *FileOutputStream*), powiązany ze plikiem danych binarnych (np. "plik2.dat");
   *FileOutputStream plik = **new** FileOutputStream("plik2.dat");*
2) W celu buforowania bajtów pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileOutputStream* należy utworzyć obiekt klasy *BufferedOutputStream*

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

3) Pojedyncze bajty mogą być zapisywane do pliku za pomocą metody bufora:
   ***void** write(**int** c)*
4) Całe ciągi bajtów mogą być zapisywane do pliku za pomocą metody bufora:

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

Aby odczytać plik:

5) Należy utworzyć obiekt (np. typu *FileInputStream*), powiązany ze plikiem danych binarnych (np. "plik2.dat");

   *FileInputStream plik = **new** FileInputStream("plik2.dat");*

6) W celu buforowania bajtów pochodzących z obiektu powiązanego ze źródłem danych np. typu *FileInputStream* należy utworzyć obiekt klasy *BufferedInputStream*
   *BufferedInputStream bufor = **new** BufferedInputStream (plik);*

7) Pojedyncze bajty mogą być odczytywane metodą bufora:
   ***int** read ();*
   np. ***int** dane = plik.read()*
8) Całe ciągi bajtów mogą być odczytane za pomocą metody bufora:

   ***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
9) Po zapisie i odczycie bufor należy zamknąć metodą *close()*

```
import java.io.*;
import java.util.*;
public class WEWY2
 {  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 short "+e);   return 0;   }
    }
 static void Zapiszplik2()
   {  int dane=0;
      try
       {  FileOutputStream plik = new FileOutputStream ("plik2.dat");
          BufferedOutputStream bufor = new BufferedOutputStream (plik);
          while (dane!=-1)
            {  System.out.print("Podaj dane: ");
               dane=weByte();
               if (dane!=-1)   bufor.write(dane);  }
          bufor.close();
       } catch (IOException e)
          {  System.out.println ("Blad zapisu pliku bajtowego"+e);   }
     }
 static void Odczytajplik2()
   { int dane=0;
     try
       {  FileInputStream plik = new FileInputStream ("plik2.dat");
          BufferedInputStream bufor = new BufferedInputStream (plik);
          dane=plik.read();
          while (dane!=-1)
             { System.out.print(dane);
               dane=bufor.read();}
          System.out.println();
          bufor.close();
        } catch (IOException e)
          {  System.out.println ("Blad odczytu pliku bajtowego"+e);   }
    }
 public static void main(String[] args)
   { Zapiszplik2();   Odczytajplik2();  }
}
```