Przykładowe klasy biblioteczne

Klasa java.lang.String

- Służy do reprezentowania niemodyfikowalnych ciągów znaków Unicode
- Metody klasy `String`
  - `length(), charAt(int), equals(String), compareTo(String), substring(int beginIndex, int endIndex)`, ...
- Każda klasa posiada metodę `toString()`, konwertującą jej obiekty do tekstów

```
String s1 = "Ala";
String s2 = new String ("ma kota");
string s3 = s1 + " " + s2;
```

```
String s1 = "Ala";
String s2 = "Ala";
if (s1 == s2) // false (!)
{  ... }
```

```
String s1 = "Ala";
String s2 = "Ala";
if (s1.equals(s2)) // true
{  ... }
```


Klasa java.lang.StringBuffer

- Zachowuje funkcjonalność `String`: `length(),charAt(int),`...
- Reprezentuje modyfikowalne ciągi znaków:
  - dodatkowe metody: `append (String s), delete(int st, int end), insert(int offset, String s), reverse(), append(StringBuffer sb), delete(int st, int end), insert(int offset, StringBuffer sb), reverse()`, ...
- Konwersja do `String`:
  - wywołanie metody `toString()`
  - przekazanie jako parametr konstruktora `String`
- Porównanie wydajności `StringBuffer` i `String`:
  - kosztowniejsza alokacja `StringBuffer`
  - operacje na `StringBuffer` wydajniejsze od konkatenacji obiektów `String`

```
String x = "a" + 4;
String y =
       new StringBuffer().append("a").append(4).toString();
```


Klasa java.lang.Math

- Klasa `java.lang.Math` pełni funkcję biblioteki matematycznej; wszystkie jej metody są typu static (nie jest konieczne tworzenie obiektów)

`Math.abs(x)`	`Math.acos(x)`	`Math.asin(x)`
`Math.atan(x)`	`Math.ceil(x)`	`Math.cos(x)`
`Math.exp(x)`	`Math.floor(x)`	`Math.log(x)`
`Math.max(x,y)`	`Math.min(x,y)`	`Math.pow(x,y)`
`Math.random()`	`Math.round(x)`	`Math.sin(x)`
`Math.sqrt(x)`	`Math.tan(x)`	

- Klasa ta zawiera również stałe reprezentujące liczby π i e

`Math.PI`	`Math.E`

```
double pierwiastek;
pierwiastek = Math.sqrt(2.0);
```

```
double dwaPi;
dwaPi = 2 * Math.PI;
```

Klasa java.util.Date

- Klasa `java.util.Date` służy do tworzenia obiektów reprezentujących datę i czas.

`Date()`	konstruktor tworzący obiekt na podstawie czasu aktualnego
`Date(year,month,day,hour,min,sec)`	konstruktor tworzący obiekt dla podanej daty
`getYear(), setYear()`	odczytuje/ustawia numer roku od 1900
`getMonth(), setMonth()`	odczytuje/ustawia numer miesiąca
`getDate(), setDate()`	odczytuje/ustawia numer dnia w miesiącu
`getDay(), setDay()`	odczytuje/ustawia numer dnia w tygodniu (od niedzieli)
`getHours(), setHours()`	odczytuje/ustawia godzinę
`getMinutes(), setMinutes()`	odczytuje/ustawia minuty
`getSeconds(), setSeconds()`	odczytuje/ustawia sekundy

Klasa java.util.Date

`after(d)`	zwraca true, jeżeli reprezentowana data jest późniejsza od d
`before(d)`	zwraca true, jeżeli reprezentowana data jest wcześniejsza od d
`equals(d)`	zwraca true, jeżeli daty są równe

```
Date teraz;
Date potem;

teraz = new Date();
potem = new Date(102,02,13,10,0,0);

System.out.println(teraz.getHours());
if (potem.after(teraz)) System.out.println("Czas p□ynie");
```


Klasa java.util.Vector

- Klasa `java.util.Vector` służy do tworzenia tablic o zmiennym rozmiarze; elementy tablicy mogą być obiektami dowolnych klas; pozycje tablicy są numerowane od zera

`addElement(x)`	dołącza na koniec tablicy nową pozycję - obiekt x
`elementAt(i)`	zwraca obiekt znajdujący się na pozycji i
`remove(i)`	usuwa z tablicy element na pozycji i
`firstElement()`	zwraca pierwszy obiekt w tablicy
`lastElement()`	zwraca ostatni obiekt w tablicy
`elements()`	zwraca obiekty jako wystąpienie klasy Enumeration
`size()`	zwraca liczbę pozycji tablicy

Klasa java.util.Vector

```
String ja = "Zbigniew";
String ty = "Jan";
String on = "Jerzy"

Vector ludzie = new Vector();

ludzie.addElement(ja);
ludzie.addElement(ty);
ludzie.addElement(on);

ludzie.remove(1);

System.out.println(ludzie.elementAt(1));
```

Klasa java.util.Hashtable

- Klasa `java.util.Vector` służy do tworzenia tablic haszowych; umożliwia odwzorowanie kluczy w wartości; klucze i wartości tablicy mogą być obiektami dowolnych klas;

`put(klucz, wartość)`	wstawia wartość adresowaną kluczem
`get(klucz)`	zwraca wartość adresowaną kluczem
`contains(wartość)`	testuje istnienie podanej wartości
`containsKey(klucz)`	testuje istnienie podanego klucza
`elements()`	zwraca wartości jako wystąpienie klasy Enumeration
`isEmpty()`	testuje czy tablica haszowa jest pusta
`remove(klucz)`	usuwa wartość adresowaną kluczem
`size()`	zwraca liczbę kluczy

Klasa java.util.Hashtable

```
String ja = "Zbigniew";
String ty = "Jan";
String on = "Jerzy";

Hashtable ludzie = new Hashtable();

ludzie.put("ja", ja);
ludzie.put("ty", ty);
ludzie.put("on", on);

ludzie.remove("ty");

System.out.println(ludzie.get("ja"));
```


Klasa java.util.Enumeration

- Klasa `java.util.Enumeration` służy do iterowania po serii elementów; elementy mogą być obiektami dowolnych klas;

`hasMoreElements()`	testuje czy są jeszcze elementy do iterowania
`nextElement()`	zwraca kolejny element

Klasa java.util.Enumeration

```
Vector ludzie = new Vector();
ludzie.addElement("Zbigniew");
ludzie.addElement("Jan");
ludzie.addElement("Jerzy");

Enumeration e=ludzie.elements();
while (e.hasMoreElemets())
        System.out.println(e.nextElement());

Hashtable zwierzeta= new Hashtable();
ludzie.put("Zbigniew", "pies");
ludzie.put("Jan", "kot");
ludzie.put("Jerzy", "kanarek");

e=zwierzeta.elements();
while (e.hasMoreElemets())
        System.out.println(e.nextElement());
```

Pakiet java.io

- Pakiet `java.io` grupuje klasy służące do obsługi plikowego wejścia/wyjścia:
  - `java.io.FileInputStream`
    Służy do odczytywania plików binarnych
  - `java.io.InputStreamReader`
    Służy do odczytywania plików tekstowych; odpowiednio konwertuje bajty odczytywane przez `FileInputStream` zgodnie ze wskazanym zestawem znaków narodowych
  - `java.io.FileOutputStream`
    Służy do zapisywania plików binarnych
  - `java.io.OutputStreamWriter`
    Służy do zapisywania plików tekstowych; odpowiednio konwertuje zapisywane znaki do bajtów dla `FileOutputStream` zgodnie ze wskazanym zestawem znaków narodowych
- Nazwy przykładowych metod: `int read()` i `write(int)`

Odczyt pliku - przykłady

binarnego

```
int value;
FileInputStream fStream;

fStream = new FileInputStream("/home/data.txt");
while ((value = fStream.read()) != -1)
  System.out.write((char)value);
fStream.close();
```

tekstowego

```
int value;
FileInputStream fStream;
InputStreamReader fReader;

fStream = new FileInputStream("/home/data.txt");
fReader = new InputStreamReader(fStream,"ISO-8859-2");
while ((value = fReader.read()) != -1)
  System.out.print((char)value);
fReader.close();
fStream.close();
```