Programowanie obiektowe

Laboratorium 1. Wstęp do programowania w języku Java.

Narzędzia

1. Aby móc tworzyć programy w języku Java, potrzebny jest zestaw narzędzi **Java Development Kit**, który można ściągnąć ze strony java.sun.com. JDK zawiera między innymi:

- kompilator **javac**

- interpreter **java**

- przeglądarkę apletów **appletviewer**

- generator dokumentacji **javadoc**

- debuger **jdb**

- narzędzie do tworzenia archiwów **jar**

Wraz z Java Development Kit zainstalowane zostanie środowisko uruchomieniowe Java Runtime Environment (które można ściągnąć jako osobny pakiet JRE, jeżeli nie chcemy tworzyć, a jedynie uruchamiać programy napisane w języku Java), którego podstawowym elementem jest maszyna wirtualna Javy.

Aby móc uruchamiać w/w narzędzia z linii poleceń bez konieczności każdorazowego podawania ich pełnej ścieżki , należy zaktualizować zmienną środowiskową **Path** w następujący sposób:

```
set Path=%Path%;<pełna ścieżka do katalogu jdk\bin>
```

2.Nieodzownym narzędziem, jakim posługuje się programista Java jest dokumentacja języka. Można ściągnąć wersję offline, bądź też otwierać dokumentację pod adresem:
http://java.sun.com/javase/6/docs/api/

3. Programy pisać można nawet w notatniku. Dobrym edytorem przeznaczonym dla programistów (w szczególności tych, którzy piszą w języku Java) jest **jedit**, który można ściągnąć TUTAJ.

Pierwszy program w języku Java

Aby aplikacja została uruchomiona, główna klasa musi zawierać metodę main(). Maszyna wirtualna Javy jest wywoływana za pomocą polecenia Java, z argumentami: nazwa pliku o rozszerzeniu „class" zawierającego metodę main() oraz argumenty wywołania tej metody, np.:

```
java nazwa_pliku arg1 arg2
```

Plik z rozszerzeniem *.java musi mieć taką samą nazwę jak główna klasa w nim opisana.

Podstawowa struktura programu:

```
class aplikacja001 {
      public static void main (String args[]) {
            System.out.println("Hello World");
      }
}
```

Po załadowaniu klasy przez JVM sterowanie zostaje przekazane do metody main() i tu zaczyna sie właściwe działanie programu: tworzenie obiektów, odwołania do innych klas aplikacji.

- ➔ Stwórz program, który wyświetli wszystkie argumenty podane wraz z jego uruchomieniem. Posłuż się pętlą for.

Pierwszy aplet w języku Java

Aplety w odróżnieniu od aplikacji nie posiadają metody main(). Główna klasa każdego apletu (np. MyApps) musi być klasa publiczna, dziedziczyć z predefiniowanej klasy Applet z pakietu java.applet, albo JApplet z pakietu javax.swing.

Do uruchomienia apletu trzeba utworzyć plik HTML, zawierający znacznik wywołania tej klasy, np.:

```
<applet code = ”MyApps.class”
width = ”300” height = ”300”>
</applet>
```

Po napotkaniu tego znacznika przeglądarka ładuje plik MyApps.class, następnie wywoływany jest konstruktor tej klasy oraz metoda inicjalizacyjna, itd.

Podstawowa struktura apletu:

```
import java.awt.*;
import java.applet.Applet;
public class aplet001 extends Applet {
      public void paint (Graphics g) {
            g.drawString("To jest pierwszy aplet", 20, 20);
      }
}
```

- ➔ Spróbuj skompilować aplet bez załączania pakietów (import)
- ➔ Spróbuj uruchomić aplet jako aplikację – jaki będzie rezultat?

Aplet uruchamiany jako aplikacja

Istnieje możliwość napisania apletu, który będzie mógł pracować zarówno jako aplet, jak i aplikacja. Wystarczy dodać do niego metodę main(), która utworzy okno (obiekt klasy Frame), a w nim egzemplarz apletu.

Przykład:

```
import java.awt.*;
import java.applet.Applet;
public class aplet002 extends Applet {
      public void paint (Graphics g) {
            g.drawString("To jest pierwszy aplet", 20, 20);
      }

      public static void main (String args[]) {
            Frame frame = new Frame ("Drugi aplet");
            //utworz obiekt klasy aplet002
            aplet002 hello = new aplet002 ();
            frame.setSize(300, 300);
            frame.add(hello);
            frame.show();
            hello.init();
      }
}
```

Wczytywanie i wyświetlanie danych

Pakiet graficzny **swing** zawiera szereg klas, które ułatwiają obsługę operacji wejścia / wyjścia za pomocą typowych „okienek”.

Klasa `JOptionPane` posiada m.in. dwie przeznaczone do tego metody:

`showInputDialog(tekst zachęty);` - pobieranie tekstu

`showMessageDialog(null, wyświetlany tekst);` - wyświetlanie tekstu

- ➔ Napisz program, który stworzy obiekt klasy JOptionPane, wczyta tekst, a następnie wyświetli go na ekran.
- ➔ Rozbuduj program, aby możliwe było podanie kilku danych wejściowych, a następnie wyświetlenie ich razem w jednym oknie.

Parsowanie tekstu

Możliwe jest dokonanie konwersji danych tekstowych wprowadzanych przez użytkownika do odpowiedniego typu danych. Należy w tym celu wykorzystać odpowiednie metody z interesujących nas klas:

`Integer.parseInt(String tekst)` – zamiana tekstu na liczbę całkowitą

`Float.parseFloat(String tekst)` – zamiana tekstu na liczbę rzeczywistą

`Zmienna_tekstowa.charAt(n)` – pobranie n-tego znaku tekstu jako zmienna typu char.

Podejście obiektowe

W języku Java wszystko traktujemy jako obiekt, niezależnie od stopnia złożoności danych czy metod na nich wywoływanych. Identyfikator obiektu który modyfikujemy jest referencją do obiektu. Przykładowo:

```
String tekst; // tworzymy referencję typu String
```

```
String tekst = ”jakiś tekst”; // tworzymy i inicjalizujemy referencję typu String,
```

Powyższy zapis jest jednak uproszczony dla typu String. Pełen mechanizm tworzenia obiektu danej klasy wygląda tak (posługujemy się operatorem **new**):

```
String tekst;
tekst  = new String (”jakiś tekst)”;
```

Typy podstawowe

Istnieje grupa typów, które nie wymagają użycia operatora **new**, a co za tym idzie, nie są tworzone obiekty, a jedynie zmienne automatyczne, nie będące referencją – jest to uzasadnione względami pamięciowymi. Typy te wymienione są poniżej, wraz z odpowiadającymi im nazwami klas, o ile istnieje potrzeba stworzenia obiektu danej klasy (z reguły można obejść się bez tego):

```
boolean    Boolean
char       Character
byte       Byte
short      Short
int        Integer
long       Long
float      Float
double     Double
void       Void
```

Przykładowo, stworzyć możemy obiekt typu podstawowego char:

```
char c = ‘x’;
```

lub też stworzyć obiekt klasy Character:

```
char c = new Character (‘x’);
```