Laborator 1:

Introducere în Java

Întocmit de: Adina Neculai

Îndrumător: Asist. Drd. Gabriel Danciu

2 octombrie 2011

I. NOŢIUNI TEORETICE

A. Ce este Java?

Java nu este doar un limbaj de programare ci este şi mediu de programare ce oferă utilizatorului atât cadrul cât şi uneltele necesare pentru a dezvolta aplicaţii Java. Limbajul Java poate fi caracterizat ca un limbaj:

1. simplu: tehnologia Java conţine aşa numitul *Garbage Colector* care eliberează programatorul de grija dezalocării memoriei.

2. familiar: limbajul Java respectă o mare parte a gramaticii şi a sintaxei de programare C/C++.

3. robust: un program Java este mai puţin supus erorilor datorită celor două nivele de verificare ale acestuia. Un nivel de verificare este la compilare şi unul este la rulare.

4. orientat pe obiecte: spre deosebire de limbajul C++, Java este în întregime orientat pe obiecte.

5. dinamic: multe decizii privind evoluţia programului se iau in momentul rulării, la *runtime*.

6. ce asigură un nivel ridicat de securitate: programele Java sunt verificate pas cu pas în timpul rulării, astfel evitându-se accesul la zone nepermise.

7. independent de platformă: un program Java poate fi rulat pe orice platformă pe care a fost instalată o maşină virtuală Java (*Java Virtual Machine*).

8. adaptat pentru multithreading: tehnologia Java permite ca un program să execute mai multe sarcini aparent în acelaşi timp, utilizând mai multe fire de execuţie (*thread-uri*).

9. adaptat pentru aplicaţii distribuite

B. Pachetul JDK

Mediul JDK (*Java Developers Kit*) conţine o serie de biblioteci de clase Java necesare scrierii unui program şi un set de utilitare necesare compilării, execuţiei şi documentării unei aplicaţii

Java. În Java o bibliotecă este cunoscută sub numele de *package*. Package-urile Java incluse în JDK formează API (*Application Programming Interface*). Mai multe informaţii se găsesc aici.

C. Dezvoltarea şi execuţia unei aplicaţii Java

Paşii ce trebuie urmaţi pentru a putea crea un program Java sunt următorii:

1. scrierea codului;
2. compilarea;
3. interpretarea şi lansarea în execuţie.

Aceste operaţii sunt prezentate în figura următoare:

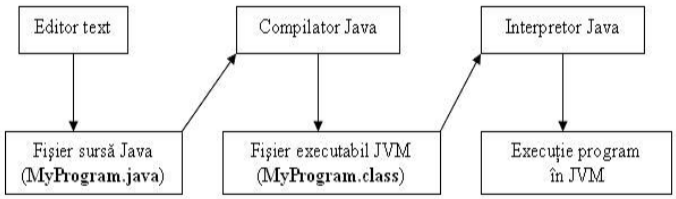

Figura 1: Etapele dezvoltării şi execuţiei unei aplicaţii Java stand-alone.

D. Structura unui program Java

```
1 package main;
2
3 public class MyProgram {
4    public static void main(String args[]) {
5       //intructiuni
6    }
7 }
```

Listing 1: Descriptive Caption Text

- linia 1: *main* reprezintă pachetul din care face parte clasa MyProgram. Fizic, un pachet este un director al proiectului.

- linia 3: clasa MyProgram are modificatorul de acces *public*;
- linia 4: metoda *main* are modificatorul de acces public, are ca parametru şirul de caractere *args* şi nu returnează nimic. De asemenea, cuvântul rezervat *static* determină ca metoda main să depindă de clasa MyProgram şi nu de o instanţă a acesteia.

II. PREZENTAREA LUCRĂRII DE LABORATOR

A. Editarea unei aplicaţii Java

Codul sursă Java constă din unul sau mai multe fişiere text ce au extensia *.java*. Pentru scrierea codului se poate folosi orice editor de text. Ca exemplu se foloseşte codul următor:

```
1 public class HelloWorld {
2    public static void main(String args[]) {
3       System.out.println("Hello World!!!");
4    }
5 }
```

Listing 2: Descriptive Caption Text

De reţinut că numele clasei trebuie să coincidă cu numele fişierului a cărui extensie este *.java*.

B. Compilarea unei aplicaţii Java

Pasul următor este compilarea programului. Acest lucru se realizează dintr-o fereastră de sistem cu ajutorul comenzii: *javac HelloWorld.java*

Dacă operaţia de compilare s-a desfăşurat cu succes, în acelaşi director ar trebui să apară un fişier cu acelaşi nume, dar cu extensia *.class*. În cazul în care fişierul amintit anterior nu s-a generat, înseamnă că s-au întâmpinat probleme la compilare.

O posibilă problemă ar fi ca sistemul de operare să nu poată lansa în execuţie utilitarul *javac*. Mesajul de eroare este, în acest caz, " **'javac' is not recognized as an internal or external command, operable program or batch file**". Pentru a rezolva această eroare trebuie setată variabila de sistem *path* pentru a cuprinde şi directorul în care se află *javac* (Exemplu: C:\Program Files \Java \jdk1.6.0_23 \bin).

O altă cauză ar putea fi lipsa pachetului JDK din sistem. Soluţia ar fi instalarea acestuia. Pachetul poate fi descarcat de la această adresă.

C. Lansarea în execuţie a unei aplicaţii Java

După compilare s-a obţinut *HelloWorld.class*. Pentru a-l lansa în execuţie se foloseşte comanda: *java HelloWorld*. În urma acestei comenzi va apărea în consolă mesajul "Hello World!!!".

D. Preluarea parametrilor din linie de comandă

Următorul exemplu este o aplicaţie care spune "Hello" utilizatorilor ce îşi dau numele ca parametru de apel al acesteia. Aplicaţia va prelua toţi aceşti parametri din argumentul metodei *main* numit *args*. Acesta este un tablou de şiruri de caractere ce va conţine toţi parametrii din linie de comandă ce urmează după numele programului.

```
1  public class HelloUsers {
2     public static void main(String args[]) {
3        if (args.length == 0){//daca nu sunt argumente
4           System.out.println("Introduceti cel putin un nume");
5        }
6        for(int i=0; i<args.length; i++){
7           System.out.println("Hello, "+args[i]+"!");//se afiseaza fiecare element din tabloul args
8        }
9     }
10 }
```

Listing 3: Descriptive Caption Text

După ce programul a fost compilat şi a fost obţinut fişierul *class*, se lansează în execuţie astfel: *java Hello Mihai Radu Ana*. În consolă se afişează:

Hello, Mihai!

Hello, Radu!

Hellor, Ana!

III. TEMĂ

Editaţi, compilaţi şi lansaţi în execuţie aplicaţiile:

1. HelloWorld din secţiunea II A;
2. HelloUsers din secţiunea II D.