TEHNICI AVANSATE DE PROGRAMARE
LABORATORUL 1

Bibliografie:
- http://java.sun.com/docs
- Horia Georgescu – Introducere in Universul Java

1. Caracteristicile limbajului:

- Simplitate: Sunt eliminate supraîncărcarea operatorilor, moștenirea multiplă, lucrul cu pointerii.
- Complet orientat pe obiect.
- Securitate: Limbajul obligă programatorul să prevadă acțiunile ce trebuie întreprinse la apariția diferitelor erori posibile.
- Portabilitate: Java este un limbaj independent de platforma de lucru.

2. Java este un limbaj compilat si interpretat.

Un program sursă Java trebuie mai întâi compilat; ca rezultat se obține un fișier "*byte-code*" (o secvență de instrucțiuni de asamblare pentru o masină imaginară, numită ***masina virtuală Java* - JVM**), care nu depinde de masina gazdă pe care va fi executat programul. Programul *byte-code* poate fi apoi transferat pe orice mașină. Fiecare mașină gazdă capabilă să execute programe Java dispune de un *interpretor*, care convertește reprezentarea "*bytecode*" (cod de octeți) în instrucțiuni mașină proprii, care apoi sunt executate; conversia are loc la lansarea executării. În acest mod este asigurată portabilitatea si independența de platformă.

3. Platforme JAVA

Limbajul Java a permis dezvoltarea unor tehnologii dedicate rezolvării unor probleme din cele mai diverse domenii. Aceste tehnologii au fost grupate în aşa numitele platforme de lucru:

- Java SE (Standard Edition) - platforma standard de lucru ce oferă suport pentru crearea de aplicaţii independente şi appleturi
- ME (Micro Edition) – pentru programarea dispozitivelor mobile
- Java EE (Enterprise Edition) - această platformă oferă API-ul necesar dezvoltării de aplicaţii complexe, formate din componente ce trebuie să ruleze în sisteme eterogene, cu informaţiile memorate în baze de date distribuite, etc, precum şi suportul necesar pentru crearea de aplicaţii şi servicii Web

4. Unitățile de compilare

Fiecare aplicație Java conține o clasă principală în care trebuie să existe metoda main. Clasele aplicației se pot găsi fie într-un singur fisier (**unitate de compilare**), fie în mai multe. Este recomandat ca fisierul care conține codul sursă al clasei primare să aibă același nume cu cel al clasei, deși acest lucru nu este obligatoriu. Fișierele vor avea obligatoriu extensia **.java.**
Un fișier poate să conțină cel mult o clasă publică (cu modificatorul public). Dacă un fișier conține o clasă publică, atunci numele fișierului trebuie să coincidă cu cel al clasei publice.
Un fisier cu extensia .java poate conține opțional, în ordine: declarații de pachete(**package**), instrucținui de includere (**import**), definiții de clase.

 **EXE:** Salvați în fișierul PrimulExemplu.java codul:

```
public class PrimulExemplu{
      public static void main(String args[]){
      // prima aplicație
            System.out.print("Salut");
      }
}
```

5. Compilarea aplicației

Pentru a compila fișierul PrimulExemplu.java utilizați compilatorul Java **javac**. Compilatorul creează câte un fișier separat pentru fiecare clasă a programului; acestea au extensia **.class** și sunt plasate implicit în același director cu fișierele sursă. Compilatorul poate fi apelat pentru orice fișier care are extensia .java.

 **EXE:** Executați comanda:

```
javac PrimulExemplu.java
```

6. Executarea aplicației

Executați aplicația prin apelul interpretorului **java**
Interpretorul primește ca argument numele clasei principale (nu numele unui fișier).

 **EXE:** Poziționați-va în directorul care conține fișierul PrimulExemplu.class și executați

```
java PrimulExemplu
```

7. Argumentele aplicației

O aplicație Java poate primi oricâte argumente de la linia de comandă în momentul lansării ei. Aceste argumente se specifică în apelul interpretorului, după numele aplicației si sunt separate prin spațiu. Argumentele sunt trimise sub forma unui vecor de șiruri de caractere (obiecte de tip String) ca parametru al metodei main.

 **EXE:** Scrieți o aplicație care să primească ca argumente două numere întregi a, b și care să calculeze a^b. Afișați rezultatul în baza de numerație 2.

Atenție:
Clasa `Math` posedă metoda:

```
static double pow(double a, double b)
```

Clasa `Integer` posedă o metoda pentru convertirea unui String către un întreg:

```
static int parseInt(String s) throws NumberFormatException.
```

Consultați documentația
http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Integer.html
pentru a gasi o metodă pentru conversia în baza 2 a unui numă întreg.

8. Funcții și date membru

EXE: Cum se poate apela din funcția main funcția mesaj?

```
public class ExempluFunctie {
	int dataMembru;
	void mesaj(){
		System.out.println("S-a apelat metoda mesaj a clasei
				ExempleVariabile. dataMembru = " + dataMembru);
	}
	ExempluFunctie(){
	}
	ExempluFunctie(int dataMembru){
		this.dataMembru = dataMembru;
	}
	public static void main(String args[])
	{
		/*completați cu aplelul metodei mesaj*/
	}
}
```

9. Variabile si literali

Variabilele pot conține:

- primitive
- referințe
- null

Variabilele pot fi

- date membre
- variabile locale
- parametrii metodelor
- parametrii de la tratarea erorilor

Primitive:

Tip	Nr de octeți	Clasă înfășurătoare
byte	1	Byte
short	2	Short
int	4	Integer
long	8	Long
float	4	Float
double	8	Double
boolean	1 bit	Boolean
char	2 codificare UNICODE	Character

EXP: Compilați și rulați Lietrali.java și observați exemple de literali precum și valorile implicite atribuite datelor membru.

Pentru fiecare tip primitiv există o clasă „înfășurătoare”. Clasele înfășurătoare conțin constantele `MIN_VALUE, MAX_VALUE.` Clasele de tip numeric conțin constantele `POSITIVE_INFINITY, NEGATIVE_INFINITY, NaN,` și metodele:

```
boolean isNaN(double a)
boolean isInfinite(double a)
```

10. Variabile de tip referință

O referinţă Java este un identificator al unui obiect şi este, de fapt un pointer ascuns, asemănător cu o referinţă C++, de care se deosebeşte în primul rând din punct de vedere sintactic (nu se mai foloseşte operatorul & la definiţia unei referinţe). O referință se poate obține prin utilizarea operatorului **new.**

Variabilele care rețin obiecte, tablouri sunt referințe.

Referințele au valoarea implicită null.

Operatori care acționează asupra referințelor

- operatori relaționali == !=
- operator pentru atribuire =
- operatorul de apartenență la o clasă `instanceof`

EXP:

```
class Intreg{
	private int i = 1;
	void setI(int i){
		this.i = i;
	}

	int getI(){
		return i;
	}
}

public class Referinte {
	public static void main(String[] args) {
		Intreg ObjUnu = new Intreg();
		Intreg ObjDoi = ObjUnu;
		if (ObjDoi == ObjUnu)
			System.out.println("sunt obiecte identice");
		ObjDoi.setI(2);
		System.out.println("s-a modificat ObjUnu " + ObjDoi.getI());
		if (ObjDoi instanceof Intreg)
			System.out.println("ObjDoi este intreg");
	}

}
```

EXP: Clase înfășurătoare

```
int i=1;
Integer wi=new Integer(i);
int j=wi.intValue();
System.out.println(j);
```

EXP: Tablouri

```
int[][] a = new int[3][];
a[0] = new int[3];
a[1] = new int[4];
a[2] = new int[2];
```

11. Convenții de notație

Este de preferat utilizarea următoarelor convenții de notație:
NumeClasa, numeVariabila, numeVariablaCompus, NUME_CONSTANTA
http://www.oracle.com/technetwork/java/codeconventions-135099.html

12. Citirea datelor de la tastatură

Clasa `Scanner` din pachetul `java.util` se poate folosi pentru citirea din diferite surse: de la tastatură, din fișier, din obiecte de tip `String`, în funcție de tipul obiectului trimis ca parametru constructorului clasei: `InputStream, File, String.`

EXP: Se citește de la tastatură un șir de caractere:

```
import java.util.*;
public class CitireTastatura {
	public static void main(String args[]){
		Scanner sc = new Scanner(System.in);
		String text = sc.next();
	}
}
```

EXP: Se citesc dintr-un fișier numere:

```
import java.io.File;
import java.io.FileNotFoundException;
public class CitireFisier {
	public static void main (String args[]) throws
	FileNotFoundException {
		java.util.Scanner sc =new java.util.Scanner(new File("numere.in"));
		while (sc.hasNextInt()){
			System.out.print(sc.nextInt());
		}
	}
}
```

EXE 1: Să se verifice dacă un text introdus de la tastatură conține doar litere și să se numere câte vocale apar în textul citit.

EXE 2: Scrieți un program care citește de la tastatură un întreg N și tipărește secvența Thue-Morse de rang N. Primele cuvinte din șirul Thue-Morse sunt 0, 01, 0110, 01101001. Fiecare cuvânt este obținut inversând biții cuvântului precedent și concatenând rezultatul la sfârșitul cuvântului anterior.

EXE 3: Creați o clasa student care va avea:

- doua câmpuri de tip String nume, prenume
- un câmp de tip float medie
- metoda clasei `Object public String toString()` suprascrisă astfel încât să returneze numele concatenate cu prenumele și cu media.
- doi constructori (student(String nume, String prenume)
și student(String nume, String prenume, float medie))

Creați un program care să citească din fișierul studenți numele, prenumele și mediile studenților (fiecare linie din fisier conține în ordine numele, prenumele și media unui student). Afișati studenții în ordinea crescătoare a mediilor.

EXE 4: Scrieți un program care să citească de la tastatură un întreg cod $=d_2d_3 \ldots d_{10}$ de 9 cifre și tipărește codul ISBN obținut prin adaugarea la cod a cifrei de control d_1. Codul ISBN obținut va fi $d_1d_2d_3 \ldots d_{10}$ astef încât $d_1 + 2d_2 + 3d_3 + \ldots + 10d_{10}$ este multiplu de 11. Dacă d_1 este 10 se va afișa x