Laborator 2 – Bazele Programarii Orientate pe Obiect

Consideratii teoretice.

1. Cum ne organizam un proiect Java

Înainte de a trece la implementare trebuie sa ne gandim cum de structuram din punct de vedere logic programul pe unitati (grupuri). Elementele care fac parte din aceeasi grup trebuie să fie **conectate în mod logic**, pentru o ușoară implementare și înțelegere ulterioară a codului. În cazul limbajului Java, aceste grupuri care contin entitati (ex. clase, interfete).conectate din punct de vedere logic intre ele poarta numele de pachete. In catrul unui proicet pachetele sunt organizate intr-o structura ierahica, putand avea o ierarhie pe mai multe nivele (ex. pachete in pachete).

Clasele pot fi create fiecare separat in cate un fisier, sau puteam avea definite mai multe clase in cadrul aceluiasi fisier respeectand urmatoarea reguli:

- dacă dorim ca această clasă să fie vizibilă din întreg proiectul, îi vom pune specificatorul **public** (vom vorbi despre specificatori de acces mai în detaliu în cursurile urmatoare); acest lucru implică însă 2 restricții:
  - fișierul și clasa publică trebuie să aibă același nume
  - nu poate exista o altă clasă/interfață publică în același fișier (vom vedea în laboratoarele următoare ce sunt interfețele)
- pot exista mai multe clase în același fișier sursă, cu condiția ca **maxim una** să fie publică

2. **Clase in Java**

Clasele reprezintă tipuri de date definite de utilizator sau deja existente (definite in pachete, parte a API Java). O clasă poate conține:

- **campuri sau atribute**
- **constructori**
- **metode**

Exemplu de declararea a unei clase:

```
class Book
{
   String name;
    int id;
    Book(String name, int id)
     {
          this.name=name;
          this.id=id;
          }
    public String toString()
     {
          return "Name: " +name;
     }
}
```

Prin **instanțierea** unei clase se înțelege crearea unui obiect care corespunde unui șablon definit. Procesul de definire a unui obiect implică: declarare, instanțiere și intializare. Instantiere se face prin intermediul cuvântului cheie **new**.

Un exemplu este următorul:
Book myZero = new Book(1, "Mandrie si Prejudecata");

Un alt exemplu este definirea unui obiect instanta a clasei **String.** Se poate instanția in doua moduri :

String s;
s = "My first string";

sau

String s = new String("My first string ");

Câmpuri (atribute). Campurile definesc starea obiectului. Un câmp este un obiect având tipul unei clase sau o variabilă de tip primitiv. Dacă este un obiect atunci trebuie inițializat înainte de a fi folosit (folosind cuvântul cheie **new**).

Sintaxa declarare campuri:

tip_acces tip nume_camp

- **tip_acces** - specifică modificator de acces ca public, privat, protejat
- **tip** – specifica tipul campului care poate fi fie primitiv, fie referinta
- **nume_camp** -specifica numele campului

Exemplu de declararea a campurilor intr-o clasa:

```
class Persoana
{
        int varsta;
        String nume;
        ......
}
```

Exemplu de declarare si initializare a unui obiect de tip persoana:

```
Persoana p = new Persoana();

// set valoarea campului varsta la 20
p.varsta = 20;
p.nume="Ana";

// folosim aceasta valoare
System.out.println("Campul varsta a obiectului p este " + p. varsta);
```

Observăm că pentru a utiliza un câmp/funcție membru dintr-o funcție care nu aparține clasei respective, folosim sintaxa:

classInstance.memberName

Constructorul. Constructorul unei clase este o metoda speciala a acelei clase care este apelata in momentul in care se creeaza o instanta a acelei clase. La crearea unui obiect al unei clase se apelează automat **constructor**. Constructorul are numele clasei, nu returnează explicit un tip anume (nici măcar void) și poate avea oricâți parametri.

Exemplu de declararea a unui constructor:

```
class Persoana
{
	int varsta;
	String nume;
	Persoana (int v, String n)
	{
		varsta=v;
		nume=n;
	}
}
```

Crearea unui obiect se face cu sintaxa:

```
MyClass instanceObject = new MyClass(param1, param2, ..., paramn);
```

In cazul exemplului nostru:

```
Persona p = new Persoana(20, "Ana");
```

În cazul in care intr-o clasa nu am declarant nici un constructor, Java crează automat un **constructor implicit** (**default constructor**) care face iniţializarea câmpurilor neinițializate, astfel:

- referințele la obiecte se inițializează cu null
- variabilele de tip numeric se inițializează cu 0
- variabilele de tip logic (boolean) se inițializează cu false

Daca avem declarat un constructor in clasa, constructorul default nu se mai creaza!!!
Metode. Metodele reprezintă operații asupra stării. Metodele în sunt un bloc de cod care conține instrucțiuni de execuție și operații.

Sintaxa declararii unei metode:

```
access_type return_type method_name(arguments) {
//method body
}
```

- tip_acces: specifică modificator de acces ca public, privat, protejat.
- return_type: Specifică dacă metoda returnează orice valoare. Poate fi nul (fără tip de returnare) sau poate specifica un tip ca Șir sau int
- nume_metodă: Orice nume valid care se potrivește comportamentului metodei
- argumente: Dacă metoda necesită parametri sau argumente pentru a efectua o operație. Este opțional.
- corpul metodei: Funcționalitatea metodei

Invocarea unei metode in Java:

O metodă este utilizabilă numai atunci când invocăm un apel o metodă. Putem apela o metodă folosind numele metodei urmat de argumente dacă există între paranteze.
method_name(arguments);

Mai jos este un exemplu simplu de invocare a unei metode. Deoarece metodele sunt definite în cadrul unei clase, putem invoca metoda folosind doar instanța clasei. În acest exemplu, adăuga este metoda și m este obiectul clasei.

```
public class Exemplu
{
	int x;
	Exemplu(int x)
	{
		this.x = x;
	}
	public void increment()
	{
		System.out.println("Valoare incremenata este: " + x++);
	}
	public static void main(String[] args)
	{
		Exemplu a = new Exemplu(5);
		System.out.println(a.increment());
	}
}
```

3. Informatii ajutatoare pentru implementarea exercitilor:

3.1 Generare numere aleatorii.
Exista doua variante prin intermediul carora se pot genera numere aleatorii in java:

3.1. **1. Folosind clasa Random**:

Pentru a utiliza această clasă pentru a genera numere aleatorii, trebuie mai întâi să creăm o instanță a clase și apoi să invocăm metode precum nextInt (), nextDouble (), nextLong () etc. folosind acea instanță.

Putem genera numere aleatorii de tip int, float, double, short, boolean folosind această clasă.

Putem transmite argumente metodelor care reprezinta limita superioara corespunzatoare numerelor care urmeaza a fi generate. De exemplu, nextInt (6) va genera numere cuprinse între 0 și 5, inclusiv 0 si 5.

Exemplu

```
import java.util.Random;
…….
Random rand = new Random();
// Generate random integers in range 0 to 999
int rand_int1 = rand.nextInt(1000);
```

```
int rand_int2 = rand.nextInt(1000);
// Generate Random doubles
double rand_dub1 = rand.nextDouble();
double rand_dub2 = rand.nextDouble();
```

3.1.2. Folosind Math.random():

Clasa **Math** conține diverse metode pentru efectuarea de diferite operații numerice, cum ar fi, calculul ridicarii la putere, logarithm, etc. Una dintre aceste metode este **random** (). Această metodă returnează o valoare de tip **double**, mai mare sau egal cu 0,0 și mai mic decât 1,0. Valorile returnate sunt alese pseudoaleatoriu.

Exemplu

```
// Generating random doubles
System.out.println("Random doubles: " + Math.random());
System.out.println("Random doubles: " + Math.random());
```

3.2. Citire date de la tastatura:

Pentru citirea datelor de la tastatura se poate

- Utiliza clasa **Scanner**

Aceasta este probabil cea mai preferată metodă de preluare a datelor. Scopul principal al clasei **Scanner** este de a analiza tipurile și șirurile primitive folosind expresii regulate, cu toate acestea, se poate folosi și pentru a citi intrările date de utilizator în linia de comandă.

Exemplu:

```
import java.util.Scanner;
// Using Scanner for Getting Input from User
Scanner in = new Scanner(System.in);
String s = in.nextLine();
System.out.println("You entered string " + s);
int a = in.nextInt();
System.out.println("You entered integer " + a);
float b = in.nextFloat();
System.out.println("You entered float " + b);
```

- Folosind clasa **Console**

A devenit o modalitate preferată de citire a informațiilor date de utilizator din linia de comandă

Exemplu:

```
// Using Console to input data from user
String name = System.console().readLine();
System.out.println("You entered string " + name);
```

Pentru conversia unui string in int se poate folosi metoda statica parseInt din clasa Integer:

Exemplu:

```
int i=Integer.parseInt("200");
```

Exemplu de conversie din String in double:

```
double d=Double.parseDouble("23.6");
```

3.3. Metode recursive

În Java, o metodă care se numește se apeleaza pe ea insasi se numeste metodă recursivă. Acest proces este cunoscut sub numele de recursivitate. Folosind algoritmul recursiv, anumite probleme pot fi rezolvate destul de ușor.

```
public static void main(String[] args) {
……
method1();
……
}
static void method1 (){
……
method1();
…….. }
```

In exemplul de mai sus, am apelat metoda method1 () din interiorul metodei main(). (apel nromal). In acelasi timp in interiorul metode method1 (), este apleata acceasi metoda method1 () . Acesta este un exemplu de apel recursiv.

Pentru a iesi din recursiv, trebuie să pune niste conditii suplimentare în cadrul metodei. În caz contrar, metoda va rula la infinit. O modalitate este folosirea structurii if ... else (sau o abordare similară) pentru a termina apelul recursiv din interiorul metodei.

Exemplu:

```
public class RecursionExample2 {
static int count=0;
static void p(){
count++;
if(count<=5){
System.out.println("hello "+count);
p();
}
}
public static void main(String[] args) {
p();
}
}
```

4. Teme de implementat in cadrul laboratorului

4.1. Scrieti o clasa TestAritmetic care genereaza doua numere aleatoare intregi si afiseaza la consola: cele doua numere, suma/ diferenta celor doua numere, inmultirea, impartirea intrega, restul impartirii celor doua numere.

4.2. Scrieti o clasa Java care genereaza numere pseudoaleatoare din domeniul 0.....65535. Presupunand ca avem o valoare initiala r0 pentru generarea urmatorului numar aleator din domeniu 0....65535 utilizam formula rnou = ((rvechi*25173) + 13849) % 65536

4.3. Scrieti o clasa java care defineste doua metode: (1) metoda java care sa returneze minimul a trei numere de acelasi tip primitiv (folosind operatorul conditional? :), (2) o metoda java care returneaza minimul dintre patru numere si care foloseste in implementare metoda anterior definita.

4.4. Presupunem ca depunem o suma (depozit la temen) intr-o banca care ofera dobanda de 25% pe an. Sa se calculeze suma finala dupa un anumit numar de ani (se va face capitalizarea contului, adica se va tine cont de “dobanda la dobanda”).

4.5. Să se scrie un program care citeşte valoarea reală a lui x şi calculează valoarea funcţiei de mai jos. Programul citeste de la tastatura valoare variabilei x.

$$f(x)=\begin{cases} x^2+4x+4 & daca \quad x<0 \\ 0 & daca \quad x=0 \\ x^2+5x & daca \quad x>0 \end{cases}$$

5. Sa se scrie un program Java care calculeaza n! (n!=!*2*...*n), unde 0<n<13 este un numar natural.