Seminar 4. Introducere in programare Java - Clase si Obiecte

Introducere

Timp de lucru estimat: 5 minute

Acest document reprezinta Seminarul 4 pentru cursul de introducere in programare Java. El contine aspecte teoretice despre clasele in Java, probleme rezolvate si 10 probleme propuse pentru exersarea conceptelor.

1 Partea Teoretica

Timp de lucru estimat: 60 minute

1.1 Clase si Obiecte in Java

Java este un limbaj de programare orientat pe obiecte, ceea ce inseamna ca se bazeaza pe conceptul de **clase** si **obiecte**. O clasa este un sablon sau un tip definit de utilizator din care se pot crea obiecte. Obiectele sunt instante ale claselor si reprezinta entitati din lumea reala cu caracteristici si comportamente specifice.

1.1.1 Definirea unei Clase

O clasa in Java este definita folosind cuvantul cheie `class`, urmat de numele clasei.

Sintaxa generala:

```
public class NumeClasa {
        // Atribute (variabile de instanta)
        // Constructori
        // Metode
}
```

1.1.2 Atribute (Variabile de Instanta)

Atributele sunt variabile care apartin unei clase si definesc proprietatile acesteia.

Exemplu:

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

1.1.3 Constructori

Constructorii sunt metode speciale folosite pentru a crea obiecte ale unei clase. Un constructor are acelasi nume ca si clasa si nu are tip de returnare.

Exemplu:

```
public class Persoana {
        String nume;
        int varsta;

        // Constructor implicit
        public Persoana() {
                this.nume = "";
                this.varsta = 0;
        }

        // Constructor cu parametri
        public Persoana(String nume, int varsta) {
                this.nume = nume;
                this.varsta = varsta;
        }
}
```

1.1.4 Metode

Metodele definesc comportamentul unei clase si pot manipula atributele acesteia.

Exemplu:

```
public class Persoana {
        String nume;
        int varsta;

        public void afiseazaDetalii() {
                System.out.println("Nume: " + nume);
                System.out.println("Varsta: " +
                   varsta);
        }
}
```

1.1.5 Crearea Obiectelor

Pentru a crea un obiect al unei clase, se foloseste cuvantul cheie `new`.

Exemplu:

```
Persoana p = new Persoana("Ion", 30);
p.afiseazaDetalii();
```

1.2 Incapsularea

Incapsularea este un principiu al programarii orientate pe obiecte care presupune ascunderea datelor unei clase si furnizarea de metode pentru accesarea si modificarea acestora.

1.2.1 Accesori si Mutatori (Getteri si Setteri)

Pentru a proteja atributele unei clase, acestea sunt declarate `private`, iar accesul la ele se face prin metode publice numite getteri si setteri.

Exemplu:

```
public class Persoana {
        private String nume;
        private int varsta;

        public String getNume() {
                return nume;
        }

        public void setNume(String nume) {
                this.nume = nume;
        }

        public int getVarsta() {
                return varsta;
        }

        public void setVarsta(int varsta) {
                this.varsta = varsta;
        }
}
```

1.3 Mostenirea

Mostenirea permite unei clase (clasa derivata sau subclasa) sa mosteneasca atributele si metodele unei alte clase (clasa de baza sau superclasa).

Exemplu:

```
public class Animal {
        public void mananca() {
                System.out.println("Animalul
                   mananca.");
        }
}
```

```
public class Caine extends Animal {
        public void latra() {
                System.out.println("Cainele
                   latra.");
        }
}
```

1.4 Polimorfismul

Polimorfismul permite obiectelor sa fie tratate ca instante ale superclasei lor, mai degraba decat ale clasei lor concrete.

Exemplu:

```
Animal animal = new Caine();
animal.mananca(); // Va apela metoda din clasa
   Animal
```

1.5 Exemple Practice

1.5.1 Exemplul 1: Clasa `Carte`

Descriere: Definiti o clasa `Carte` cu atributele `titlu`, `autor` si `numarPagini`. Implementati constructori, getteri si setteri, si o metoda pentru afisarea detaliilor despre carte.

Implementare in Java:

```
public class Carte {
        private String titlu;
        private String autor;
        private int numarPagini;

        // Constructor implicit
        public Carte() {
                this.titlu = "";
                this.autor = "";
                this.numarPagini = 0;
        }

        // Constructor cu parametri
        public Carte(String titlu, String autor,
           int numarPagini) {
                this.titlu = titlu;
                this.autor = autor;
                this.numarPagini = numarPagini;
        }

        // Getteri si Setteri
        public String getTitlu() {
```

```
                return titlu;
        }

        public void setTitlu(String titlu) {
                this.titlu = titlu;
        }

        public String getAutor() {
                return autor;
        }

        public void setAutor(String autor) {
                this.autor = autor;
        }

        public int getNumarPagini() {
                return numarPagini;
        }

        public void setNumarPagini(int numarPagini)
           {
                this.numarPagini = numarPagini;
        }

        // Metoda pentru afisarea detaliilor
        public void afiseazaDetalii() {
                System.out.println("Titlu: " +
                   titlu);
                System.out.println("Autor: " +
                   autor);
                System.out.println("Numar pagini: "
                   + numarPagini);
        }
}
```

Utilizare:

```
public class Main {
        public static void main(String[] args) {
                Carte carte = new Carte("Povestea
                   lui Harap Alb", "Ion Creanga",
                   120);
                carte.afiseazaDetalii();
        }
}
```

1.5.2 Exemplul 2: Clasa `ContBancar`

Descriere: Definiti o clasa `ContBancar` cu atributele `numarCont` si `sold`. Implementati metode pentru depunere, retragere si afisarea soldului.

Implementare in Java:

```
public class ContBancar {
        private String numarCont;
        private double sold;

        public ContBancar(String numarCont) {
                this.numarCont = numarCont;
                this.sold = 0.0;
        }

        public void depunere(double suma) {
                if (suma > 0) {
                        sold += suma;
                        System.out.println("Ati
                           depus: " + suma);
                } else {
                        System.out.println("Suma
                           trebuie sa fie
                           pozitiva.");
                }
        }

        public void retragere(double suma) {
                if (suma > 0 && suma <= sold) {
                        sold -= suma;
                        System.out.println("Ati
                           retras: " + suma);
                } else {
                        System.out.println("Fonduri
                           insuficiente sau suma
                           invalida.");
                }
        }

        public void afiseazaSold() {
                System.out.println("Soldul curent
                   este: " + sold);
        }
}
```

Utilizare:

```
public class Main {
        public static void main(String[] args) {
```

```
                ContBancar cont = new
                    ContBancar("RO123456789");
                cont.depunere(500);
                cont.retragere(200);
                cont.afiseazaSold();
        }
}
```

1.5.3 Exemplul 3: Mostenire - Clasele `Vehicul` si `Masina`

Descriere: Definiti o clasa de baza `Vehicul` cu atributele `viteza` si `capacitate`. Derivati clasa `Masina` care mosteneste `Vehicul` si adauga atributul `numarUsi`.

Implementare in Java:

```
public class Vehicul {
        protected int viteza;
        protected int capacitate;

        public Vehicul(int viteza, int capacitate) {
                this.viteza = viteza;
                this.capacitate = capacitate;
        }

        public void afiseazaDetalii() {
                System.out.println("Viteza: " +
                    viteza);
                System.out.println("Capacitate: " +
                    capacitate);
        }
}

public class Masina extends Vehicul {
        private int numarUsi;

        public Masina(int viteza, int capacitate,
            int numarUsi) {
                super(viteza, capacitate);
                this.numarUsi = numarUsi;
        }

        public void afiseazaDetalii() {
                super.afiseazaDetalii();
                System.out.println("Numar usi: " +
                    numarUsi);
        }
}
```

Utilizare:

```
public class Main {
        public static void main(String[] args) {
                Masina masina = new Masina(180, 5,
                   4);
                masina.afiseazaDetalii();
        }
}
```

2 Partea Practica

Timp de lucru estimat: 60 minute

2.1 Probleme Rezolvate

Problema 1: Clasa `Student`

Enunt: Definiti o clasa `Student` care are atributele `nume`, `varsta` si `nota`. Implementati metode pentru setarea si obtinerea valorilor atributelor, precum si o metoda care determina daca studentul a promovat (nota ¿= 5).

Implementare in Java:

```
public class Student {
        private String nume;
        private int varsta;
        private double nota;

        public Student(String nume, int varsta,
           double nota) {
                this.nume = nume;
                this.varsta = varsta;
                this.nota = nota;
        }

        // Getteri si Setteri
        public String getNume() {
                return nume;
        }

        public void setNume(String nume) {
                this.nume = nume;
        }

        public int getVarsta() {
                return varsta;
        }

        public void setVarsta(int varsta) {
```

```
                this.varsta = varsta;
        }

        public double getNota() {
                return nota;
        }

        public void setNota(double nota) {
                this.nota = nota;
        }

        // Metoda pentru a verifica daca studentul
           a promovat
        public boolean aPromovat() {
                return nota >= 5;
        }
}
```

Utilizare:

```
public class Main {
        public static void main(String[] args) {
                Student student = new
                   Student("Maria", 20, 7.5);
                System.out.println("Nume: " +
                   student.getNume());
                System.out.println("Varsta: " +
                   student.getVarsta());
                System.out.println("Nota: " +
                   student.getNota());

                if (student.aPromovat()) {
                        System.out.println("Studentul
                           a promovat.");
                } else {
                        System.out.println("Studentul
                           nu a promovat.");
                }
        }
}
```

Problema 2: Clasa `Dreptunghi`

Enunt: Creati o clasa `Dreptunghi` cu atributele `lungime` si `latime`. Implementati metode pentru calculul ariei si a perimetrului.

Implementare in Java:

```
public class Dreptunghi {
        private double lungime;
        private double latime;
```

```
        public Dreptunghi(double lungime, double
            latime) {
                this.lungime = lungime;
                this.latime = latime;
        }

        public double calculeazaAria() {
                return lungime * latime;
        }

        public double calculeazaPerimetru() {
                return 2 * (lungime + latime);
        }
}
```

Utilizare:

```
public class Main {
        public static void main(String[] args) {
                Dreptunghi dreptunghi = new
                    Dreptunghi(5.0, 3.0);
                System.out.println("Aria: " +
                    dreptunghi.calculeazaAria());
                System.out.println("Perimetrul: " +
                    dreptunghi.calculeazaPerimetru());
        }
}
```

2.2 Probleme Propuse

1. Clasa `Cerc`

Definiti o clasa `Cerc` cu atributul `raza`. Implementati metode pentru calculul ariei si a circumferintei cercului.

2. Clasa `Angajat`

Creati o clasa `Angajat` cu atributele `nume`, `departament` si `salariu`. Implementati metode pentru afisarea datelor angajatului si pentru calculul salariului anual.

3. Clasa `Factura`

Definiti o clasa `Factura` care contine atributele `numarFactura`, `descriere`, `cantitate` si `pretUnitate`. Implementati o metoda care calculeaza valoarea totala a facturii.

4. Clasa `Calculator`

Creati o clasa `Calculator` care ofera metode pentru operatii aritmetice de baza: adunare, scadere, inmultire si impartire.

5. Clasa `Temperatura`

Definiti o clasa `Temperatura` cu metode pentru conversia temperaturii din grade Celsius in Fahrenheit si invers.

6. Clasa `Contor`

Creati o clasa `Contor` care incrementeaza sau decrementeaza o valoare intreaga. Implementati metode pentru resetarea contorului si pentru obtinerea valorii curente.

7. Clasa `Patrat`

Definiti o clasa `Patrat` care mosteneste clasa `Dreptunghi` din problema rezolvata si adapteaza metodele pentru un patrat.

8. Clasa `StudentMaster`

Creati o clasa `StudentMaster` care mosteneste clasa `Student` si adauga atributul `specializare`. Implementati o metoda pentru afisarea tuturor detaliilor.

9. Clasa `Complex`

Definiti o clasa `Complex` pentru numere complexe cu atributele `parteReala` si `parteImaginara`. Implementati metode pentru adunarea si inmultirea a doua numere complexe.

10. Clasa `Biblioteca`

Creati o clasa `Biblioteca` care gestioneaza o colectie de obiecte de tip `Carte`. Implementati metode pentru adaugarea, eliminarea si afisarea cartilor din biblioteca.

2.3 Instructiuni pentru Rezolvare

Pentru fiecare problema:

- Analizati cerintele si identificati atributele si metodele necesare.
- Definiti clasa cu atributele corespunzatoare.
- Implementati constructori, getteri si setteri daca este necesar.
- Scrieti metodele cerute in enunt.
- Creati o clasa `Main` sau un program de test pentru a demonstra functionalitatea clasei.