Moștenirea și Polimorfismul

1. Clase abstracte

- O clasă abstractă este o clasă care nu poate fi instanțiată direct si se folosește ca model pentru alte clase.
- Poate conține metode abstracte (fără corp) și metode normale.
- O clasă care moștenește o clasă abstractă trebuie să implementeze metodele abstracte.

```
abstract class Animal {
    abstract void scoateSunet(); // metodă abstractă
}
```

2. Polimorfismul (Polymorphism)

- În Java, permite folosirea aceleiași referințe pentru a apela metode din obiecte de tipuri diferite.
- Comportamentul real este determinat la rulare (runtime).

```
Animal a = new Caine();
a.scoateSunet(); // Afișează "Ham!"
```

1. Se definește o clasă abstractă Animal cu:
    - metoda abstractă void scoateSunet();
    - metoda String getNume().
- Se definesc clasele Caine, Pisica și Vaca care extind Animal și suprascriu metoda scoateSunet().
- În clasa TestAnimale:
    - se creează un vector de obiecte de tip Animal;
    - se apelează metoda scoateSunet() pentru fiecare obiect.

```java
package lab6;
abstract class Animal {
    abstract void scoateSunet();
    abstract String getNume();
    @Override
    public String toString() {
        return getNume();
    }
}
class Caine extends Animal {
    @Override
    void scoateSunet() {
        System.out.println("Ham!");
    }
    @Override
    String getNume() {
        return "Caine";
    }
}
class Pisica extends Animal {
    @Override
    void scoateSunet() {
        System.out.println("Miau!");
    }

    @Override
    String getNume() {
        return "Pisica";
    }
}
class Vaca extends Animal {
    @Override
    void scoateSunet() {
        System.out.println("Muuu!");
    }

    @Override
    String getNume() {
        return "Vaca";
    }
}

public class TestAnimale {
    public static void main(String[] args) {
        Animal[] animale = {
            new Caine(),
            new Pisica(),
            new Vaca(),
            new Caine()
        };

        for (Animal a : animale) {
            System.out.print(a + " face: ");
            a.scoateSunet(); // polimorfism
        }
    }
}
```

2. Se definește clasa Vehicul cu atribute: marca, an și metoda afiseazaDetalii().

- Se definesc clasele Masina și Motocicleta care extind Vehicul și suprascriu metoda afiseazaDetalii().
- În clasa TestVehicule:
    - se creează un vector de obiecte de tip Vehicul (mașini și motociclete);
    - se afișează informațiile pentru fiecare vehicul folosind polimorfismul.

```java
package lab6;
import java.util.Scanner;
class Vehicul {
    protected String marca;
    protected int an;

    public Vehicul(String marca, int an) {
        this.marca = marca;
        this.an = an;
    }
    void afiseazaDetalii() {
        System.out.println("Vehicul " + marca + ", an " + an);
    }
}

class Masina extends Vehicul {
    private int numarUsi;

    public Masina(String marca, int an, int numarUsi) {
        super(marca, an);
        this.numarUsi = numarUsi;
    }
    @Override
    void afiseazaDetalii() {
        System.out.println("Mașină " + marca + ", an " + an + ", uși: " + numarUsi);
    }
}
class Motocicleta extends Vehicul {
    private boolean cauciucSpateLat;

    public Motocicleta(String marca, int an, boolean cauciucSpateLat) {
        super(marca, an);
        this.cauciucSpateLat = cauciucSpateLat;
    }
```

```java
class Motocicleta extends Vehicul {
    private boolean cauciucSpateLat;

    public Motocicleta(String marca, int an, boolean cauciucSpateLat) {
        super(marca, an);
        this.cauciucSpateLat = cauciucSpateLat;
    }

    @Override
    void afiseazaDetalii() {
        System.out.println("Motocicletă " + marca + ", an " + an +
                ", cauciuc spate lat: " + (cauciucSpateLat ? "da" : "nu"));
    }
}

public class TestVehicule {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Vehicul[] vehicule = new Vehicul[3];

        vehicule[0] = new Masina("Dacia", 2018, 4);
        vehicule[1] = new Motocicleta("Yamaha", 2020, true);
        vehicule[2] = new Masina("VW", 2015, 5);

        for (Vehicul v : vehicule) {
            v.afiseazaDetalii();
        }

        sc.close();
    }
}
```

3. Se definește clasa ContBancar cu atribute: titular, sold și metode:

    - depunere(), retragere(), afiseaza().

- Se definesc clasele ContEconomii și ContCurent care extind ContBancar:
    - ContEconomii: are atribut rataDobanda și metodă aplicaDobanda().
    - ContCurent: are atribut limitaDescoperit și permite retrageri până la limită.
- În clasa TestConturi:
    - se creează un vector de conturi diferite;
    - se efectuează operații de depunere/retragere;
    - se afișează rezultatele și se demonstrează polimorfismul.

```
class ContBancar {
    protected String titular;
    protected double sold;
    public ContBancar(String titular, double soldInitial) {
        this.titular = titular;
        this.sold = soldInitial;
    }
    public void depunere(double suma) {
        sold += suma;
    }
    public boolean retragere(double suma) {
        if (suma <= sold) {
            sold -= suma;
            return true;
        }
        return false;
    }
    public void afiseaza() {
        System.out.println(titular + " - sold: " + sold);
    }
}
class ContEconomii extends ContBancar {
    private double rataDobanda; // ex: 0.05 = 5%
    public ContEconomii(String titular, double soldInitial, double rataDobanda) {
        super(titular, soldInitial);
        this.rataDobanda = rataDobanda;
    }
    public void aplicaDobanda() {
        sold += sold * rataDobanda;
    }
    @Override
    public void afiseaza() {
        System.out.println(titular + " (Economii) - sold: " + sold +
                ", rata dobândă: " + (rataDobanda * 100) + "%");
    }
}

class ContCurent extends ContBancar {
    private double limitaDescoperit;

    public ContCurent(String titular, double soldInitial, double limitaDescoperit) {
        super(titular, soldInitial);
        this.limitaDescoperit = limitaDescoperit;
    }

    @Override
    public boolean retragere(double suma) {
        if (suma <= sold + limitaDescoperit) {
            sold -= suma;
            return true;
        }
        return false;
    }

    @Override
    public void afiseaza() {
        System.out.println(titular + " (Curent) - sold: " + sold +
                ", limită descoperit: " + limitaDescoperit);
    }
}

public class TestConturi {
    public static void main(String[] args) {
        ContBancar[] conturi = new ContBancar[3];

```

```
62
63 public class TestConturi {
64     public static void main(String[] args) {
65         ContBancar[] conturi = new ContBancar[3];
66
67         conturi[0] = new ContEconomii("Ana", 1000, 0.05);
68         conturi[1] = new ContCurent("Bogdan", 500, 300);
69         conturi[2] = new ContBancar("Mihai", 800);
70
71         ((ContEconomii) conturi[0]).aplicaDobanda();
72         conturi[1].retragere(700); // foloseste descoperitul
73         conturi[2].retragere(900); // esuează
74
75         for (ContBancar c : conturi) {
76             c.afiseaza();
77         }
78     }
79 }
80
```

4.Se definește clasa Persoana cu atribute: nume, cnp și metoda afiseazaInfo().

- Se definesc clasele Student și Profesor care extind Persoana.
    - Student: atribut suplimentar medie.
    - Profesor: atribute suplimentare specializare și salariu.
- În clasa TestUniversitate:
    - se citesc datele persoanelor de la tastatură (tip, nume, CNP, medie/specializare/salariu);
    - se salvează într-un vector de tip Persoana[];
    - se afișează informațiile pentru fiecare obiect folosind metoda afiseazaInfo().

```java
package lab6;
import java.util.Scanner;
class Persoana {
    protected String nume;
    protected String cnp;
    public Persoana(String nume, String cnp) {
        this.nume = nume;
        this.cnp = cnp;
    }
    public void afiseazaInfo() {
        System.out.println("Persoană: " + nume + ", CNP: " + cnp);
    }
}
class Student extends Persoana {
    private double medie;
    public Student(String nume, String cnp, double medie) {
        super(nume, cnp);
        this.medie = medie;
    }
    @Override
    public void afiseazaInfo() {
        System.out.println("Student: " + nume + ", CNP: " + cnp + ", Medie: " + medie);
    }
}
class Profesor extends Persoana {
    private String specializare;
    private double salariu;
    public Profesor(String nume, String cnp, String specializare, double salariu) {
        super(nume, cnp);
        this.specializare = specializare;
        this.salariu = salariu;
    }
    @Override
    @Override
    public void afiseazaInfo() {
        System.out.println("Profesor: " + nume + ", CNP: " + cnp +
                ", Specializare: " + specializare + ", Salariu: " + salariu);
    }
}

public class TestUniversitate {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Introduceți numărul de persoane: ");
        int n = sc.nextInt();
        sc.nextLine(); // curățare buffer
        Persoana[] persoane = new Persoana[n];
        for (int i = 0; i < n; i++) {
            System.out.println("\nPersoană #" + (i + 1));
            System.out.print("Tipul persoanei (1 - Student, 2 - Profesor): ");
            int tip = sc.nextInt();
            sc.nextLine(); // curățare buffer
            System.out.print("Nume: ");
            String nume = sc.nextLine();
            System.out.print("CNP: ");
            String cnp = sc.nextLine();
            if (tip == 1) {
                System.out.print("Medie: ");
                double medie = sc.nextDouble();
                persoane[i] = new Student(nume, cnp, medie);
            } else if (tip == 2) {
                System.out.print("Specializare: ");
                String specializare = sc.nextLine();
                if (specializare.isEmpty()) { // dacă a rămas gol din cauza enterului
                    specializare = sc.nextLine();
                }
```

```
                    specializare = sc.nextLine();
                }
                System.out.print("Salariu: ");
                double salariu = sc.nextDouble();
                persoane[i] = new Profesor(nume, cnp, specializare, salariu);
            } else {
                System.out.println("Tip invalid! Se adaugă o persoană generică.");
                persoane[i] = new Persoana(nume, cnp);
            }
            sc.nextLine(); // curățare pentru următorul ciclu
        }

        System.out.println("\n--- Lista Persoanelor ---");
        for (int i = 0; i < persoane.length; i++) {
            persoane[i].afiseazaInfo();
        }

        sc.close();
    }
}
```

5.Se definește clasa abstractă Produs cu atribute: denumire, pret și metoda calculeazaPretFinal().

- Se definesc clasele Electronice, Haine și Alimente care **extind** Produs.
    - Electronice: reducere 10%.
    - Haine: reducere 20%.
    - Alimente: atribut aproapeExpirare; dacă e true → reducere 30%.
- În clasa TestMagazin:
    - se citesc produsele de la tastatură (tip, denumire, preț, stare expirare);
    - se calculează prețul final pentru fiecare;
    - se afișează lista produselor și totalul de plată.

```java
package lab6;
import java.util.Scanner;
//Clasa abstractă de bază
abstract class Produs {
 protected String denumire;
 protected double pret;

 public Produs(String denumire, double pret) {
     this.denumire = denumire;
     this.pret = pret;
 }
 // Metodă comună pentru calculul prețului final
 public double calculeazaPretFinal() {
     return pret;
 }
 @Override
 public String toString() {
     return denumire + " (preț de bază: " + pret + ")";
 }
}
//Subclasa pentru electronice
class Electronice extends Produs {
 public Electronice(String denumire, double pret) {
     super(denumire, pret);
 }
 @Override
 public double calculeazaPretFinal() {
     return pret * 0.9; // reducere 10%
 }
}
//Subclasa pentru haine
class Haine extends Produs {
 public Haine(String denumire, double pret) {
     super(denumire, pret);
 }

 @Override
 public double calculeazaPretFinal() {
     return pret * 0.8; // reducere 20%
 }
}

//Subclasa pentru alimente
class Alimente extends Produs {
 private boolean aproapeExpirare;

 public Alimente(String denumire, double pret, boolean aproapeExpirare) {
     super(denumire, pret);
     this.aproapeExpirare = aproapeExpirare;
 }

 @Override
 public double calculeazaPretFinal() {
     if (aproapeExpirare == true) {
         return pret * 0.7; // reducere 30%
     } else {
         return pret;
     }
 }

 @Override
 public String toString() {
     return denumire + " (preț de bază: " + pret + ", aproape de expirare: " + aproapeExpirare + ")";
 }
}

```

```java
//Clasa principală
public class TestMagazin {
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Introduceți numărul de produse: ");
    int n = sc.nextInt();
    sc.nextLine(); // curățare buffer

    Produs[] cos = new Produs[n];

    // Citirea produselor de la tastatură
    for (int i = 0; i < n; i++) {
        System.out.println("\nProdus #" + (i + 1));
        System.out.println("1 - Electronice");
        System.out.println("2 - Haine");
        System.out.println("3 - Alimente");
        System.out.print("Alegeți tipul produsului: ");
        int tip = sc.nextInt();
        sc.nextLine(); // curățare

        System.out.print("Introduceți denumirea produsului: ");
        String denumire = sc.nextLine();

        System.out.print("Introduceți prețul de bază: ");
        double pret = sc.nextDouble();

        if (tip == 1) {
            cos[i] = new Electronice(denumire, pret);
        } else if (tip == 2) {
            cos[i] = new Haine(denumire, pret);
        } else if (tip == 3) {
            System.out.print("Produsul este aproape de expirare? (true / false): ");
            boolean aproapeExpirare = sc.nextBoolean();
```

MainForma.java | package-inf... | TestAnimale.... | TestVehicul... | TestConturi.... | *TestUniver...

```java
            String denumire = sc.nextLine();

            System.out.print("Introduceți prețul de bază: ");
            double pret = sc.nextDouble();

            if (tip == 1) {
                cos[i] = new Electronice(denumire, pret);
            } else if (tip == 2) {
                cos[i] = new Haine(denumire, pret);
            } else if (tip == 3) {
                System.out.print("Produsul este aproape de expirare? (true / false): ");
                boolean aproapeExpirare = sc.nextBoolean();
                cos[i] = new Alimente(denumire, pret, aproapeExpirare);
            } else {
                System.out.println("Tip invalid! Produs ignorat.");
                cos[i] = new Electronice("Necunoscut", 0);
            }
        }

        // Afișarea produselor și calculul totalului
        double total = 0;
        System.out.println("\n--- Coșul de cumpărături ---");

        for (int i = 0; i < cos.length; i++) {
            double pretFinal = cos[i].calculeazaPretFinal();
            System.out.println(cos[i].toString() + " -> Preț final: " + pretFinal + " RON");
            total = total + pretFinal;
        }

        System.out.println("\nTotal de plată: " + total + " RON");

        sc.close();
    }
}
```