# Exercicis Resolts del Bloc 2
# (Programació orientada a objectes)

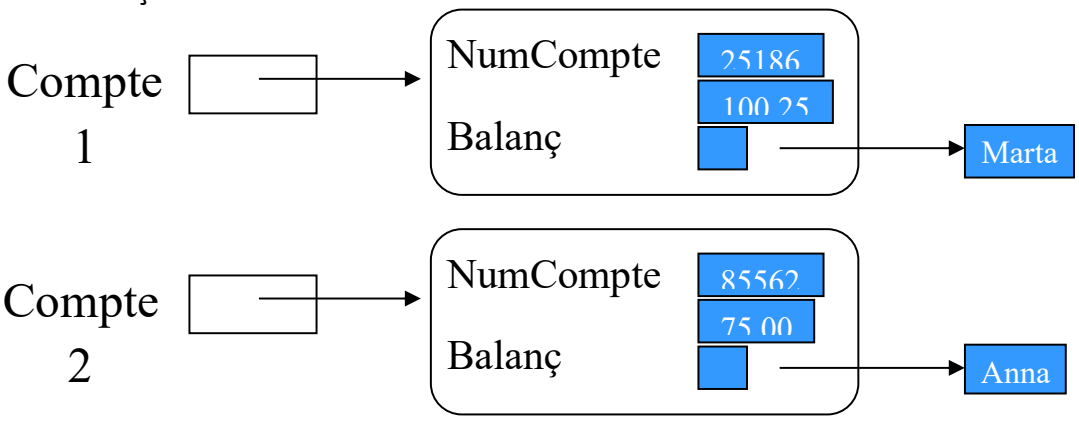
1. Enumereu el tipus de mètodes que coneixes i feu-ne una petita descripció.
2. Creieu que és viable tenir un mètode accessor de lectura (o consultors) amb visibilitat privada?
3. Creieu que té sentit que tots els mètodes accessors d'escriptura (o modificadors) tinguin visibilitat pública?
4. Quina diferència hi ha entre les responsabilitats de classe i les d'instància?
5. Poseu un exemple de responsabilitats de classe, diferent dels constructors i destructors, per a una classe Vehicle.
6. Per quin motiu el constructor ha de ser un mètode amb responsabilitat de classe i no ho pot ser amb responsabilitat d'instància?

7. Implementa en Java la classe abstracta Figura, i les classes Quadrat i Cercle que hereten de Figura.
8. Amplia la implementació de la classe Cercle que hereta de la classe abstracta Figura amb
    - Un contador de cercles,
    - Dos mètodes propis,
        1. Un mètode d'objecte per comparar cercles i
        2. Un mètode de classe per comparar cercles.

9. Genereu dues classes, A i B, amb els constructors per defecte (és a dir, sense cap argument), que tornin per pantalla algun missatge per saber quin és el constructor de cada una de les classes. Després, genereu una nova classe C que hereti de A, definiu un objecte de B dins de C i un constructor que no faci res. En acabar, definiu un objecte de la classe C i mostreu els resultats.
10. Modifiqueu l'exercici anterior per tal que les classes A i B tinguin un constructor amb arguments. Escriviu un constructor per a la classe C i executeu totes les inicialitzacions necessàries dins el constructor de C.
11. Genereu una classe anomenada Root que contingui una instància de cada una de les tres classes Comp1, Comp2 i Comp3. Deriveu una classe nova, Node a partir de la classe Root que contingui una instància de les tres classes "component". En el constructor de totes les classes, col·loqueu un missatge de manera que retorni per pantalla un identificador de la classe on és i mireu el resultat.

12. Genereu una classe que tingui un mètode sobrecarregat. Genereu una classe A que tingui un mètode per a tornar per pantalla un paràmetre que, si és enter, n'hi sumi 1; si és real, n'hi sumi 3,5, i si és una cadena de caràcters, hi afegeixi un guió davant i un altre al darrere.
13. Creeu tres classes, A, B i C, de manera que formin una jerarquia de classes. La classe A és la superclasse i les classes B i C hereten de la classe A. Implementeu un mètode en la classe A anomenat toString que no rebi paràmetres i que retorni una cadena que indiqui que és un mètode de la classe A i, posteriorment, sobreescriviu aquest mètode en les classes B i C. Creeu una instància de cada classe i executeu aquest mètode.
14. Creeu una interfície iA que tingui un mètode anomenat getValue que retorni un enter; implementeu també una classe A que tingui un mètode constructor que rebi un enter i l'emmagatzemi en un atribut. Aquesta classe A ha d'implementar la interfície iA. Posteriorment, creeu una instància de la classe A i invoqueu el mètode getValue.

15. Implementeu en Java una classe que
    - Representi un compte bancari mitjançant una classe anomenada Account
    - El seu estat inclou el numero de compte, el saldo actual i el nom del propietari
    - Els serveis són: afegir o extraure diners, afegir interessos i obtenir un balanç dels diners




16. Implementeu en Java una classe *Transactions* que:
    - Utilitzi la classe *Account* i que declari diferents comptes
    - Afegeixi diners a cada compte
    - Permeti retirar diners d'un compte
    - Comprovar el balanç d'algun dels comptes
17. Implementeu en Java una classe *Vacances* on:

- El programa sol·licita el dia, mes i any de sortida del viatge i la durada del mateix. Si la data és superior al dia d'avui, mostra la data de final del viatge, sinó la data d'inici.
- Consulteu el javadoc de les classes *java.util.Date* i *java.util.GregorianCalendar* i utilitzeu el que creieu oportú.

18. Definiu a la classe "**FormaGeometrica**" un nou mètode abstracte imprimir() que mostri per consola la informació del tipus d'objecte ("rectangle", "triangle", ...), l'àrea i el perímetre.
    - Crear un vector de 4 posicions. Omplir aquest vector amb diferents instàncies de formes geomètriques: un cercle, dos rectangles i un triangle.
    - Recorreu el vector invocant al mètode imprimir() per a cada element.
19. Feu el mateix que a l'exercici anterior, però ara fent servir una llista genèrica (homogènia): ArrayList (que implementa la interficie List) per guardar les formes geomètriques i un iterador (classe Iterator) per fer el recorregut.
20. Feu el mateix que a l'exercici anterior però ara fent un recorregut usant el **for ... each**.
21. Feu una llista genèrica LinkedList de cercles. Afegir 3 cercles i recórrer la llista mostrant el radi de cada cercle.
22. Repetiu els exercicis 2 i 4 amb llistes heterogènies, fent servir iteradors i castings.
23.
    a. Definir la classe abstracta **Persona** amb un atribut nom, el mètode get i el set. Crear la classe **Estudiant** que a més de tenir un nom té un NIUB amb el get i el set. Crea un programa que mostri per pantalla el nom i NIUB de tres estudiants.
    b. Definir la interfície **Persona** amb els mètodes setName i getName. Implementa aquesta interfície en una nova classe **Estudiant** amb la mateixa funcionalitat de l'exercici anterior. Executa el programa de l'exercici anterior utilitzant la classe **Estudiant** d'aquest exercici.
    c. Quina diferència hi ha entre la classe **Persona** de l'exercici a i la interfície Persona de l'exercici b?

24. Quina diferència hi ha entre la classe **Persona** de l'exercici 23 i la interfície Persona de l'exercici 24?

# Solucions:

1. En aquest bloc, hem parlat dels tipus de mètodes següents:
   • Mètodes constructors: permeten crear instàncies de la classe.
   • Mètodes accessors d'escriptura: permeten modificar el valor dels atributs de la classe.
   • Mètodes accessors de lectura i atributs derivats: retornen el valor d'un atribut i permeten resoldre algun càlcul o procés basant-se en els valors dels atributs i en els arguments del mètode.
   • Mètodes de classe: permeten fer algun procés o càlcul relacionat amb la classe, però sense haver de tenir una instància de la classe.
2. Els mètodes accessors de lectura (els que poden consultar el valor dels atributs de la classe) normalment s'utilitzen per a donar accés als altres objectes, per això solen ser mètodes públics que formen part de la interfície de la classe. De tota manera, pot tenir sentit restringir la visibilitat d'un accessor de lectura quan no es vol que un atribut sigui visible per les altres classes. Recordeu que, per qüestions de qualitat, sempre s'hauria d'accedir als atributs per mitjà de mètodes accessors.
3. La resposta ha de ser rotundament no. Fixeu-vos que seria un contrasentit definir com a privats o protegits tots els atributs (costum més que recomanable) i després definir com a públics tots els accessors d'escriptura corresponents. Només s'haurà de donar visibilitat pública a aquells accessors d'escriptura que hagin de ser necessàriament visibles per altres classes, sempre que es pugui assegurar que el seu possible ús extern no comprometrà el bon funcionament de l'aplicació.
4. Les responsabilitats d'instància (tant si són atributs com si són mètodes) són aquelles que es demanen als objectes d'una classe, tant sí és de recordar coses com de fer-ne. En el cas de les responsabilitats de classe, és la classe –i no les seves instàncies– la que emmagatzema dades o realitza els processos o càlculs.
5. Uns comptadors del nombre de classes creades, destruïdes, etc.
6. Com que el constructor ens serveix per a crear una instància d'un objecte, no el podem definir com a instància ja que, inicialment, no en tenim cap i, per tant, no en podem crear cap. A banda, els mètodes d'instància serveixen per a modificar l'estat de la instància, i el propòsit del mètode constructor no és aquest.

7. Implementació de les classes Figura, Quadrat i Cercle:

```
//*********************************************************************
// Figura.java
//*********************************************************************
public abstract class Figura {
   protected String color;
   protected double x, y;
   protected double area;
   protected double perimetre;
   // Mètodes abstractes:
   public abstract double calculaArea();
   public abstract double calculaPerimetre();
   //Retorna el Color
   public String getColor(){
      return color;
   }
   //Assigna el Color
    public void setaColor(String color){
      this.color=color;
   }
   //Retorna la posició de la Figura
    public double [] getPosicion(){
      double [] posicioxy = {x, y};
      return posicioxy;
   }
   //Assigna la posició de la Figura
   public void setPosicio(double[] posicioxy){
      x=posicioxy[1];
      y=posicioxy[2];
   }
}

//*********************************************************************
// Quadrat.java
//*********************************************************************
public class Quadrat extends Figura {
   private double costat; // longitud dels costats
   // constructors
   public Quadrat() {
         costat=0.0; }
   public Quadrat(double costat) {
         this.costat=costat; }
   // Calcula l'àrea del quadrat
   public double calculaArea() {
      area = (float) (costat * costat );
      return area;
```

```
   }
   // Calcula el valor del perímetre
   public double calculaPerimetre(){
      perimetre = (double) (4 * costat);
      return perimetre;
   }
//*********************************************************************
// Cercle.java
//*********************************************************************
public class Cercle extends Figura {
        public static final double PI=3.14159265358979323846;
        public double x, y, radi;
        // constructors
        public Cercle(double x, double y, double radi) {
                this.x=x; this.y=y; this.radi =radi
                numCercles++;}
        public Cercle(double radi) { this(0.0, 0.0, radi); }
        public Cercle(Cercle c) { this(c.x, c.y, c. radi); }
        public Cercle() { this(0.0, 0.0, 1.0); }
         // calcula l'area del cercle
           public double calculaArea() {
              area = (double) (PI * radi * radi);
              return area;
           }
           // calcula el valor del perímetre
           public double calculaPerimetre(){
              perimetre = (double) (2 * PI * radi);
              return perimetre;
           }
} // fi de la classe Cercle
```

8. Reimplementació de la classe Cercle:

```
//*********************************************************************
// Cercle.java
//*********************************************************************
public class Cercle extends Figura {
        static int numCercles = 0;
        public static final double PI=3.14159265358979323846;
        public double x, y, radi;
        // constructors
        public Cercle(double x, double y, double radi) {
                this.x=x; this.y=y; this.radi =radi
                numCercles++;}
        public Cercle(double radi) { this(0.0, 0.0, radi); }
        public Cercle(Cercle c) { this(c.x, c.y, c. radi); }
        public Cercle() { this(0.0, 0.0, 1.0); }
         // calcula l'area del cercle
           public double calculaArea() {
              area = (double) (PI * radi * radi);
              return area;
           }
           // calcula el valor del perímetre
           public double calculaPerimetre(){
              perimetre = (double) (2 * PI * radi);
              return perimetre;
           }
        // método de objeto para comparar círculos
        public Circulo elMayor(Circulo c) {
                if (this.r>=c.r) return this; else return c;
        }
        // método de clase para comparar círculos
        public static Circulo elMayor(Circulo c, Circulo d) {
                if (c.r>=d.r) return c; else return d;
        }
} // fi de la classe Cercle
```

9. Implementació de les classes A, B, C i Main:

```
// A.java    *****************************************************
public class A {
        public A() {
                System.out.println("ClassA constructor method");
```

```
        }
}
// B.java    ********************************************************
public class B {
        public B() {
                System.out.println("ClassB constructor method");
        }
}
// C.java    ********************************************************
public class C extends A {
        private B b;
        public C() {
                super();
        }
}
// Main.java    ******************************************************
public class Main {
        public static void main(String[] args) {
                C c = new C();
        }
}
// Sortida per pantalla*************************************************
ClassA constructor method
```

10. Implementació de les classes A, B, C i Main:

```
// A.java    ********************************************************
public class A {
        public A(String pName) {
                System.out.println("ClassA constructor method - Hello " + pName);
        }
}
// B.java    ********************************************************
public class B {
        public B(String pName) {
                System.out.println("ClassB constructor method - Hello " + pName);
        }
}

// C.java    ********************************************************
public class C extends A {
        private B b;
        public C(String pName) {
                super(pName);
```

```
                b = new B(pName);
        }
}
// Main.java    ********************************************************
public class Main {
        public static void main(String[] args) {
                C c = new C("my friend");
        }
}

// Sortida per pantalla*************************************************
ClassA constructor method - Hello my friend
ClassB constructor method - Hello my friend
```

11. Implementació de les classes Comp1, Comp2, Comp3, Root, Node, Main:

```
// Comp1.java    ********************************************************
public class Comp1 {
        public Comp1() {
                System.out.println("Com1 constructor method");
        }
}

// Comp2.java    ********************************************************
public class Comp2 {
        public Comp2() {
                System.out.println("Com2 constructor method");
        }
}
// Comp3.java    ********************************************************
public class Comp3 {
        public Comp3() {
                System.out.println("Com3 constructor method");
        }
}
// Root.java    *********************************************************
public class Root {
        private Comp1 c1;
        private Comp2 c2;
        private Comp3 c3;
        public Root() {
                c1 = new Comp1();
                c2 = new Comp2();
                c3 = new Comp3();
                System.out.println("Root constructor method");
        }
```

```
}
// Node.java    ********************************************************
public class Node extends Root{
        private Comp1 c1;
        private Comp2 c2;
        private Comp3 c3;
        public Node() {
                super();
                c1 = new Comp1();
                c2 = new Comp2();
                c3 = new Comp3();
                System.out.println("Node constructor method");
        }
}

// Main.java    ********************************************************
public class Main {
        public static void main(String[] args) {
                Node n = new Node();
        }
}

// Sortida per pantalla****************************************
Com1 constructor method
Com2 constructor method
Com3 constructor method
Root constructor method
Com1 constructor method
Com2 constructor method
Com3 constructor method
Node constructor method
```

12. Implementació de les classes A i Main:

```
// A.java    ********************************************************
public class A {
        public static void method(int i) {
                System.out.println("Param type = int - Result : " + (i + 1));
        }
        public static void method(double d) {
                System.out.println("Param type = double – Result : " + (d + 3.5));
        }
        public static void method(String s) {
                String S = "-";
                S = S.concat(s).concat("-");
```

```
                System.out.println("Param type = String – Result : " + S);
        }
}
// Main.java    ********************************************************
public class Main {
        public static void main(String[] args) {
                A.method(2);
                A.method(2.7);
                A.method("Good morning");
        }
}
// Sortida per pantalla*****************************************
Param type = int - Result : 3
Param type = double - Result : 6.2
Param type = String - Result : -Good morning
```

13. Implementació de les classes A, B, C i Main:

```
// A.java    ********************************************************
public abstract class A {
        public A() {
        }
        public String toString() {
                return "This method is owned by class A";
        }
}
// B.java    ********************************************************
public class B extends A {
        public B() {
                super();
        }
        public String toString() {
                return "This method is owned by class B";
        }
}
// C.java    ********************************************************
public class C extends A {
        public C() {
                super();
        }
        public String toString() {
                return "This method is owned by class C";
        }
}
// Main.java    ********************************************************
public class Main {
```

```
        public static void main(String[] args) {
                B b = new B();
                C c = new C();
                System.out.println(b.toString());
                System.out.println(c.toString());
        }
}
// Sortida per pantalla****************************************
This method is owned by class B
This method is owned by class C
```

14. Implementació de la interfície iA i les classes A i Main:

```
// iA.java    ********************************************************
public interface iA {
        public int getValue();
}
// A.java    *********************************************************
public class A implements iA {
        private int value;
        public A(int i) {
                this.value = i;
        }
        public int getValue() {
                return value;
        }
}
// Main.java    ******************************************************
public class Main {
        public static void main(String[] args) {
                A a = new A(321);
                System.out.println("Valor de l'atribut = " + a.getValue());
        }
}

// Sortida per pantalla****************************************
Valor de l'atribut = 321
```

15. Implementació de la classe Account:

```
//*****************************************************************
// Account.java
//*****************************************************************
import java.text.NumberFormat;
```

```
public class Account
{
	private final double RATE = 0.035; // constant for interest rate of 3.5%
	private long acctNumber;
	private double balance;
	private String name;
	//-----------------------------------------------------------------
	// Sets up the account by defining its owner, account number,
	// and initial balance.
	//-----------------------------------------------------------------
	public Account (String owner, long account, double initial)
	{
		name = owner;
		acctNumber = account;
		balance = initial;
	}
	//-----------------------------------------------------------------
	// Deposits the specified amount into the account. Returns the
	// new balance.
	//-----------------------------------------------------------------
	public double deposit (double amount)
	{
		balance = balance + amount;
		return balance;
	}
	//-----------------------------------------------------------------
	// Withdraws the specified amount from the account and applies
	// the fee. Returns the new balance.
	//-----------------------------------------------------------------
	public double withdraw (double amount, double fee){
		balance = balance - amount - fee;
		return balance;
	}
	//-----------------------------------------------------------------
	// Adds interest to the account and returns the new balance.
	//-----------------------------------------------------------------
	public double addInterest (){
		balance += (balance * RATE);
		return balance;
	}
	//-----------------------------------------------------------------
	// Returns the current balance of the account.
	//-----------------------------------------------------------------
	public double getBalance (){
		return balance;
	}
```

```
        //-----------------------------------------------------------------
        // Returns a one-line description of the account as a string.
        //-----------------------------------------------------------------
        public String toString (){
                NumberFormat fmt = NumberFormat.getCurrencyInstance();
                return (acctNumber + "\t" + name + "\t" + fmt.format(balance));
        }
} // Fi classe Account
```

16. Implementació de la classe Transaccions:

```
//*****************************************************************
// Transactions.java
//*****************************************************************
public class Transactions
{
        //-----------------------------------------------------------------
        // Creates some bank accounts and requests various services.
        //-----------------------------------------------------------------
        public static void main (String[] args)
        {
                Account acct1 = new Account ("Ted Murphy", 72354, 102.56);
                Account acct2 = new Account ("Jane Smith", 69713, 40.00);
                Account acct3 = new Account ("Edward Demsey", 93757, 759.32);
                acct1.deposit (25.85);
                double smithBalance = acct2.deposit (500.00);
                System.out.println ("Smith balance after deposit: " + smithBalance);
                System.out.println ("Murphy balance after withdrawal: " +
                acct2.withdraw (430.75, 1.50));
                acct1.addInterest();
                acct2.addInterest();
                acct3.addInterest();
                System.out.println ();
                System.out.println (acct1);
                System.out.println (acct2);
                System.out.println (acct3);
        }
}
```

17. Implementació de la classe Vacances:

```
//*****************************************************************
// Vacances.java
//*****************************************************************
import java.util.Date;
```

```
import java.util.GregorianCalendar;
import java.util.Calendar;
public class Vacances {
	private Date dataInici;
	private Date dataFinal;
	private int durada;
	private GregorianCalendar calendar;
	public Vacances(int dia, int mes, int any){
		durada = 0;
		calendar = new GregorianCalendar();
		calendar.set(any, mes, dia);
		dataInici = calendar.getTime();
		dataFinal = calendar.getTime();
	}
	public void setDies(int dies){
		durada = dies;
	}
	public Date getInici(){
		return dataInici;
	}
	public Date getFinal(){
		Date avui = new Date();
		if (avui.before(dataInici)) {
			calendar.add(Calendar.DATE, durada);
			dataFinal = (Date) calendar.getTime();
		}
		return dataFinal;
	}
	public static void main(String[] args){
		if (args.length < 4) {
			System.out.println ("Falten params: vacances <dia> <mes> <any>
		<durada> ");
		}else {
			int dia = Integer.parseInt(args[0]);
			int mes = Integer.parseInt(args[1]); // 0 gener, 1 febrer, etc.
			int any = Integer.parseInt(args[2]);
			int durada = Integer.parseInt(args[3]);
			Vacances vac = new Vacances(dia, mes, any);
			vac.setDies(durada);
			System.out.println("Les vacances comencen: " + vac.getInici());
			System.out.println("I acaben "+ vac.getFinal());
		}
	}
}
```

18. Reimplementació de les classes Figura, Cercle, Quadrat afegint el mètode imprimir i implementació de la classe TestFiguraGeometrica:

```
//*********************************************************************
// Figura.java
//*********************************************************************
public abstract class Figura {
   protected String color;
   protected double x, y;
   protected double area;
   protected double perimetre;

   // Mètodes abstractes:
   public abstract double calculaArea();
   public abstract double calculaPerimetre();
   public abstract void imprimir();

   //Retorna el Color
   public String getColor(){
      return color;
   }
   //Assigna el Color
    public void setaColor(String color){
      this.color=color;
   }
   //Retorna la posició de la Figura
    public double [] getPosicion(){
      double [] posicioxy = {x, y};
      return posicioxy;
   }
   //Assigna la posició de la Figura
   public void setPosicio(double[] posicioxy){
      x=posicioxy[1];
      y=posicioxy[2];
   }
}

//*********************************************************************
// Quadrat.java
//*********************************************************************
public class Quadrat extends Figura {
   private double costat; // longitud dels costats
   // constructors
   public Quadrat() {
         costat=0.0; }
   public Quadrat(double costat) {
```

```
        this.costat=costat; }
   // Calcula l'àrea del quadrat
   public double calculaArea() {
      area = (float) (costat * costat );
      return area;
   }
   // Calcula el valor del perímetre
   public double calculaPerimetre(){
      perimetre = (double) (4 * costat);
      return perimetre;
   }
   public void imprimir(){
      System.out.println("Imprimeix Quadrat");
          System.out.println("Àrea Quadrat= " + calculaArea());
      System.out.println("Perímetre Quadrat= " + calculaPerimetre());
   }

}
//*********************************************************************
// Cercle.java
//*********************************************************************
public class Cercle extends Figura {
        static int numCercles = 0;
        public static final double PI=3.14159265358979323846;
        public double x, y, radi;
        // constructors
        public Cercle(double x, double y, double radi) {
                this.x=x; this.y=y; this.radi =radi
                numCercles++;}
        public Cercle(double radi) { this(0.0, 0.0, radi); }
        public Cercle(Cercle c) { this(c.x, c.y, c. radi); }
        public Cercle() { this(0.0, 0.0, 1.0); }
         // calcula l'area del cercle
           public double calculaArea() {
              area = (double) (PI * radi * radi);
              return area;
           }
           // calcula el valor del perímetre
           public double calculaPerimetre(){
              perimetre = (double) (2 * PI * radi);
              return perimetre;
           }
        // método de objeto para comparar círculos
```

```
        public Circulo elMayor(Circulo c) {
                if (this.r>=c.r) return this; else return c;
        }
        // método de clase para comparar círculos
        public static Circulo elMayor(Circulo c, Circulo d) {
                if (c.r>=d.r) return c; else return d;
        }
        public void imprimir(){
             System.out.println("Imprimeix Cercle");
             System.out.println("Àrea Cercle= " + calculaArea());
             System.out.println("Perímetre Cercle= " + calculaPerimetre());
          }

} // fi de la classe Cercle

//*********************************************************************
// TestFiguraGeometrica.java
//*********************************************************************
public class TestFiguraGeometrica {
   public static void main(String [] args){
      Cercle c = new Cercle(1);
      Quadrat q = new Quadrat(1);
           Figura [] v = {c, q};
      for(int idx=0; idx < v.length; idx++) {
         Figura f = v[idx];
         f.imprimir();
      }
   }
}
//*********************************************************************
Sortida per pantalla:
//*********************************************************************
Imprimeix Cercle
Àrea Cercle= 3.141592653589793
Perímetre Cercle= 6.283185307179586
Imprimeix Quadrat
Àrea Quadrat= 1.0
Perímetre Quadrat= 4.0
```

19. Implementació de la classe TestFiguraGeometrica, utilitzant les classes Figura, Cercle i Quadrat de l'exercici anterior

```
//*********************************************************************
// TestFiguraGeometrica.java
//*********************************************************************
public class TestFiguraGeometrica {
   public static void main(String [] args){
Cercle c = new Cercle(1);
     Quadrat q = new Quadrat(1);
     ArrayList<Figura> vec = new ArrayList<Figura>();
         vec.add(c);
      vec.add(q);
      for(Iterator<Figura> i = vec.iterator(); i.hasNext(); ){
        i.next().imprimir();
      }
   }
}
//*********************************************************************
Sortida per pantalla:
//*********************************************************************
Imprimeix Cercle
Àrea Cercle= 3.141592653589793
Perímetre Cercle= 6.283185307179586
Imprimeix Quadrat
Àrea Quadrat= 1.0
Perímetre Quadrat= 4.0
```

20. Implementació de la classe TestFiguraGeometrica, utilitzant les classes Figura, Cercle i Quadrat de l'exercici anterior

```
//*********************************************************************
// TestFiguraGeometrica.java
//*********************************************************************
public class TestFiguraGeometrica {
   public static void main(String [] args){
Cercle c = new Cercle(1);
     Quadrat q = new Quadrat(1);
     ArrayList<Figura> vec = new ArrayList<Figura>();
         vec.add(c);
      vec.add(q);
```

```
        for(Figura item: vec){
       item.imprimir();
     }
   }
}
//*********************************************************************
Sortida per pantalla:
//*********************************************************************
Imprimeix Cercle
Àrea Cercle= 3.141592653589793
Perímetre Cercle= 6.283185307179586
Imprimeix Quadrat
Àrea Quadrat= 1.0
Perímetre Quadrat= 4.0
```

21. Implementació de la classe TestFiguraGeometrica, utilitzant les classes Figura, Cercle i Quadrat de l'exercici anterior

```
//*********************************************************************
// TestFiguraGeometrica.java
//*********************************************************************
public class TestFiguraGeometrica {
public static void main(String [] args){
   Cercle c1 = new Cercle(1);
   Cercle c2 = new Cercle(2);
   LinkedList<Cercle> vec = new LinkedList<Cercle>();
   vec.add(c1);
   vec.add(c2);
   for(Iterator<Cercle> i = vec.iterator(); i.hasNext();) {
      System.out.println("El radi del cercle és = " + i.next().getRadi());
   }
   }
}
//*********************************************************************
Sortida per pantalla:
//*********************************************************************
El radi del cercle és = 1.0
El radi del cercle és = 2.0
```

22. Implementació de la classe TestFiguraGeometrica, utilitzant les classes Figura, Cercle i Quadrat de l'exercici anterior

```
//*********************************************************************
// TestFiguraGeometrica.java
//*********************************************************************
public class TestFiguraGeometrica {
   public static void main(String [] args){
      Cercle c = new Cercle(1);
      Quadrat q = new Quadrat(1);
      ArrayList vec = new ArrayList();
       vec.add(c);
       vec.add(q);
       for(Iterator i = vec.iterator(); i.hasNext();){
          Object obj = i.next();
          if (obj instanceof Cercle){
             Cercle nf = (Cercle) obj; // downcasting
             nf.imprimir();
          }else{
             Quadrat nf = (Quadrat) obj; // downcasting
             nf.imprimir();
          }
       }
   }
}
//*********************************************************************
Sortida per pantalla:
//*********************************************************************
Imprimeix Cercle
Àrea Cercle= 3.141592653589793
Perímetre Cercle= 6.283185307179586
Imprimeix Quadrat
Àrea Quadrat= 1.0
Perímetre Quadrat= 4.0
```

23. Implementació de la classe Persona,
    a. Implementació de la classe Persona, Estudiant i Main.

```
//*********************************************************************
// Persona.java
//*********************************************************************
public abstract class Persona{
   private String nom;
   public void setNom(String nom){
```

```
    this.nom = nom;
  }
  public String getNom(){
    return nom;
  }
}
//**********************************************************************
// Estudiant.java
//**********************************************************************
public class Estudiant extends Persona{
  private String niub;
  public String getNiub(){
    return niub;
  }
  public void setNiub(String niub){
    this.niub = niub;
  }
}
//**********************************************************************
// Main.java
//**********************************************************************
public class Main {
public static void main(String[] args) {
    Estudiant e1 = new Estudiant(),
          e2 = new Estudiant(),
          e3 = new Estudiant();
    e1.setNom("Joan");
    e1.setNiub("11111111");
    e2.setNom("Marc");
    e2.setNiub("22222222");
    e3.setNom("Anna");
    e3.setNiub("33333333");

    System.out.println("Estudiant 1:");
    System.out.println(e1.getNom());
    System.out.println(e1.getNiub());
    System.out.println("Estudiant 2:");
    System.out.println(e2.getNom());
    System.out.println(e2.getNiub());
    System.out.println("Estudiant 3:");
    System.out.println(e3.getNom());
```

```
      System.out.println(e3.getNiub());
   }
}
```

b. Implementació de la interfície Persona, i la classe Estudiant

```
//*********************************************************************
// Persona.java
//*********************************************************************
public interface Persona {
   public void setNom(String nom);
   public String getNom();
}

//*********************************************************************
// Estudiant.java
//*********************************************************************
public class Estudiant implements Persona {
   String nom, niub;
   public void setNom(String nom) {
      this.nom = nom;
   }
   public String getNom() {
      return nom;
   }
   public void setNiub(String niub){
      this.niub = niub;
   }
   public String getNiub(){
      return niub;
   }
}
```

c. Les interfícies són com classes abstractes, perquè no es poden instanciar i només s'utilitzen per heretar. La diferència és que les interfícies no poden tenir variables (només constants) i els mètodes no poden tenir cos.

## Referències:

[1] **Construcción de software orientado a objetos**, Bertrand Meyer. Prentice Hall, 1998.

[2] **Programació orientada a objectes**. Joan Arnedo Moreno i Daniel Riera i Terrén. Manuals de la UOC. Col·lecció 105. ISBN:978-84-9788-582-9.05/2007.

[3] **Head First Java**. Bert Bates, Kathy Sierra. O'Reilly Media, 2005.