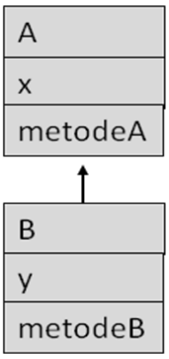
Exercicis d'Examen Parcial

Exercici
Donat el següent codi de la classe A i la classe B (que hereta de la classe A) i el diagrama il·lustrant la relació entre les classes

```
public class A{
        protected int x;
        public void metodeA(){
                ....
        }
}
public class B extends A{
        private int y;
        public void metodeB(){
                ....
        }
}
```



Respon a les següents preguntes:

1) Indica al costat de cadascuna de les línies del següent codi si hi haurà errors de compilació o no i explica breument perquè:

```
0       public static void main(String[] args) {
1               A a = new B();   OK
2               B b = new B();   OK
3               A c;             OK
4               c = b;           OK
5               int j = b.x;     OK
6               int i = a.x;     OK
7               int k = a.y;     Error de compilació, l'atribut y no està defint per a A.
8               a.metodeA();     OK
9               a.metodeB();     Error de compilació, metodeB no està definit per a A
10              b.metodeA();     OK
11              b.metodeB();     OK
12      }
```

1) Especifica si hi ha alguna **conversió** de tipus **implícita** en el codi anterior i en cas afirmatiu en quines línies.
Solució:
Sí, a la línia 1 i 4

2) Si afegim un nou mètode a la classe A anomenat imprimir que imprimeix el missatge "Missatge d'A", però no el sobreescriu a la classe B, que passa quan fem una crida d'aquesta forma:
```
b.imprimir();
```
Solució:
Apareixerà el missatge: "Missatge d'A"

3) Indica com has de sobreescriure el mètode imprimir a la classe B de manera que quan fas la crida
```
b.imprimir();
```
La sortida sigui: "Missatge de B"

Solució:
public void imprimir(){

```
        System.out.println("Missatge de B");
}
```

4) Ara, indica com has de sobreescriure el mètode imprimir a la classe B de manera que quan fas la crida

        b.imprimir();

La sortida sigui: "Missatge d'A"
"Missatge de B"

Solució:

```
public void  imprimir(){
        super.imprimir();
        System.out.println("Missatge de B");
}
```

5) Donades les classes A i B definides a dalt, indica quins seran els atributs visibles en una instància de la classe B.

Solució:

Al declarar la variable **x** com a ***protected*** indica que només aquesta classe, les clases que deriven d'ella i les classes del propi paquet tenen permis per utilizar-la.
La variable **y** és *privada* (***private***), la qual cosa vol dir que només les funcions membre d'aquesta classe tenen permis per utilizar-la. Es a dir, podem resumir la visibilitat com:

- Atributs visibles internament: **x i y**
- Atributs visibles des de classes del mateix paquet: **x**
- Atributs visibles des de classes d'un altre paquet: **cap**

Exemple:

1. Nova classe del mateix paquet

```
package paquetB;

public class NovaB {
    public static void main(String[] args) {
        B b = new B();
         System.out.println("access a x" + b.x);
         // System.out.println("access a x" + b.y);
    }
}
```

2. Nova classe d'un altre paquet

```
package nouPaquet;
import paquetB.B;

public class NovaB {
    public static void main(String[] args) {
        B b = new B();
         // System.out.println("access a x" + b.x);
         // System.out.println("access a x" + b.y);
    }
}
```

Exercici

Donat el següent programa

```
public class Aliases {
    public static void main(String[]args) {
            Persona x = new Persona();
            Persona y = new Persona();
            Persona z;
            x.edat = 20;
            y.edat = 30;
            z = y;
            y = x;
            x = z;
            x.edat = 26;
            System.out.println("L'edat en l'objecte referenciat per x és:" + x.edar);
            System.out.println("L'edat en l'objecte referenciat per y és:" + y.edat);
            System.out.println("L'edat en l'objecte referenciat per z és:" + z.edat);
    }
}
class Persona {
   public String nom;
   public int edat;
}
```

Indica quina és la sortida.

Solució:

Persona x = new Persona();
Persona y = new Persona();
Persona z;
x.edat = 20;
y.edat = 30;
z = y;

x →Objecte1 (edat=20)
y →Objecte2 (edat=30)
z →Objecte2 (edat=30)

y=x
x →Objecte1 (edat=20)
y →Objecte1 (edat=20)
z →Objecte2 (edat=30)

x=z
x → Objecte2 (edat=30)
y →Objecte1 (edat=20)
z →Objecte2 (edat=30)

x.edat = 26;
x → Objecte2 (edat=26)
y →Objecte1 (edat=20)
z →Objecte2 (edat=26)

Sortida per pantalla:

L'edat en lobjecte referenciat per x es:26
L'edat en lobjecte referenciat per y es:20
L'edat en lobjecte referenciat per z es:26

Exercici

Donada la interfície **InstrumentMusical** amb els mètodes tocar i afinar definida a continuació:

```
interface InstrumentMusical {
        public void tocar();
        public void afinar();
}
```

Defineix amb llenguatge Java la classe **Guitarra** que implementa aquesta interfície i imprimeix els següents missatges dins dels mètodes tocar: "La guitarra sona" i afinar: "La guitarra s'afina". Implementa també en la classe Guitarra un mètode sumaGuitarres que actualitza el compte de la quantitat d'objectes creats dins d'aquesta classe. Fes en el mètode main un bucle for de 10 iteracions que creï una guitarra cada vegada i al sortir del bucle imprimeix per pantalla el número de guitarres.

Solució:

```
//*********************************************************************
// Guitarra.java
//*********************************************************************
public class Guitarra implements InstrumentMusical{
   private static int numGuitarres=0;
   // constructor
   public Guitarra(){
         sumarGuitarres();
   }
   // Estem obligats a implementar els mètodes de la interfície InstrumentMusical
   public void tocar(){
      System.out.println("La guitarra sona");
   }
   public void afinar(){
      System.out.println("La guitarra s'afina");
   }

   private static void sumarGuitarres(){
      numGuitarres++;
   }
// definim el mètode sumarGuitarres public o private depenent de la utilització que anem a fer
}


//*********************************************************************
// ProvaGuitarres.java
//*********************************************************************

public class ProvaGuitarres {

  public static void main(String [] args){

      for (int i=0;i<10;i++){
         Guitarra g = new Guitarra();
      }
      System.out.println("Número de Guitarres = " + Guitarra.numGuitarres);
   }
}

//*********************************************************************
Sortida:
//*********************************************************************

Número de Guitarres = 10
```

Programació 2.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques. UB
Laura Igual

Exercici

a) "A la Facultat de Matemàtiques de la UB es desitja desenvolupar un programa que permeti emmagatzemar en una llista els estudiants que s'inscriuen per participar en una activitat de la Facultat. La informació d'interès d'un estudiant és el nom, edat i número de carnet d'estudiant. En aquesta activitat hi poden participar estudiants de grau i estudiants de màster. Els primers tindran associada la següent informació: el nom del grau que estan cursant, el número d'assignatures que han matriculat aquest any i el preu de cada assignatura (preu únic per a totes les assignatures segons el grau). Els estudiants de màster tindran associada la següent informació: el nom del màster que estan cursant, el preu del màster i si disposen d'alguna beca o no."

Seguint aquest enunciat, implementa en Java les tres classes següents que es necessitaran per al programa:

- **Estudiant** (classe abstracta),
- **EstudiantGrau** (classe que hereta de la classe **Estudiant**)
- **EstudiantMaster** (classe que hereta de la classe **Estudiant**)

incloent els seus atributs (que s'han de deduir de l'enunciat) i els següents mètodes:

- Un sol constructor on s'inicialitzen tots els atributs de la classe
- Tots els mètodes consultors i modificadors per als atributs de la classe
- Un mètode per calcular el preu de la matrícula d'aquest any anomenat imprimirPreuMatricula.
  El mètode imprimirPreuMatricula serà un mètode abstracte de la classe **Estudiant**, ja que s'han de conèixer els estudis que està cursant l'estudiant per tal de poder fer el càlcul del preu de la matrícula. A la classe **EstudiantGrau**, aquest mètode calcularà el preu de la matrícula multiplicant el número d'assignatures matriculades pel seu preu únic i imprimirà per pantalla el nom de l'estudiant i el preu resultant de la seva matrícula. A la classe **EstudiantMaster**, aquest mètode imprimirà per pantalla el nom de l'estudiant i el preu de la matrícula del màster que serà igual a la quantitat del preu del màster si l'estudiant no té una beca i serà 0 euros si l'estudiant té una beca.

Organitza el codi en fitxers amb noms i indica els paquets i els imports necessaris.

b) Implementa una altra classe **Activitat** amb un mètode main on es crea una llista genèrica d'estudiants i s'ompli amb les següents instàncies:

- Maria de 18 anys, estudiant del grau de Matemàtiques, que té un carnet amb el número 0001 i que cursa 4 assignatures amb un preu de 100,00 euros cadascuna.
- Joan de 20 anys, estudiant del grau d'Enginyeria Informàtica, que té un carnet amb el número 0002 i que cursa 5 assignatures amb un preu de 100,00 euros cadascuna.
- Marc de 23 anys, estudiant del màster d'Intel·ligència Artificial amb un preu de matrícula de 300,00 euros, que té un carnet amb el número 0003 i que té beca.

Recorre aquesta llista fent servir un iterador i invoca el mètode imprimirPreuMatricula() per a cada element.

c) Indica quina és la sortida del mètode main.

Solució:

```
//*********************************************************************
// Estudiant.java
//*********************************************************************
package edu.ub.prog2.activitat;
public abstract class Estudiant {
        protected String nom;
        protected int edat;
        protected int numCarnet;

     //Constructors
        public Estudiant(String pNom, int pEdat, int pNumCarnet) {
                nom = pNom;
                edat = pEdat;
                numCarnet = pNumCarnet;
```

```
        }
    // Mètodes consultors:
        public String getNom() {
                return nom;
    }
        public int getEdat() {
                return edat;
    }
    public int getNumCarnet() {
                return numCarnet;
    }

    // Mètodes modificadors:
        public void setNom(String pNom) {
                 nom = pNom;
    }
        public void setEdat(int pEdat) {
                edat = pEdat;
    }
        public void setNumCarnet(int pNumCarnet) {
                 numCarnet = pNumCarnet;
    }

    //Mètode abstracte, es declara sense implementació
    public abstract void imprimirPreuMatricula();
}


//*********************************************************************
// EstudiantGrau.java
//*********************************************************************
package edu.ub.prog2.activitat;
public class EstudiantGrau extends Estudiant{
   private String nomGrau;
   private int numAssig;
   private double preuAssig;

   //Constructor
   public EstudiantGrau(String pNom, int pEdat, int pNumCarnet, String pNomGrau, int
pNumAssig, double pPreuAssig) {
      super(pNom, pEdat, pNumCarnet); // aquesta crida sempre al principi del mètode.
      nomGrau = pNomGrau;
      numAssig = pNumAssig;
      preuAssig = pPreuAssig;
   }
//Comentaris:
// No es pot definir un constructor sense paràmetres si no he definit a la classe pare un
// constructor sense paràmetres.
// S'ha d'utilitzar super, ja que sinó es crida al constructor per defecte de la classe pare
// Estudiant() i no està disponible perquè s'ha sobreescrit amb el constructor amb arguments

   // Mètodes consultors:
   public String getNomGrau() {
         return nomGrau;
   }
   public int  getNumAssig() {
         return numAssig;
   }
   public double getPreuAssig() {
         return preuAssig;
   }
```

```
    // Mètodes modificadors:
    public void setNomGrau(String pNomGrau) {
            nomGrau = pNomGrau;
    }
    public void setNumAssig(int pNumAssig) {
            numAssig = pNumAssig;
    }
    public void setPreussig(double pPreuAssig) {
            preuAssig = pPreuAssig;
    }

    //Implementació del mètode abstracte de la classe pare Estudiant
    public void imprimirPreuMatricula() {
        double preuMatricula;
        preuMatricula = numAssig * preuAssig;
        System.out.println("El preu de la matrícula de l'estudiant de grau " + nom + " és " +
preuMatricula);
    }
}

//*********************************************************************
// EstudiantMaster.java
//*********************************************************************

package edu.ub.prog2.activitat;
public class EstudiantMaster extends Estudiant{
    private String nomMaster;
    private double preuMaster;
    private boolean becat;

    //Constructor
    public EstudiantMaster(String pNom, int pEdat, int pNumCarnet, String pNomMaster, double
pPreuMaster, boolean pBecat) {
        super(pNom, pEdat, pNumCarnet);
        nomMaster = pNomMaster;
        preuMaster = pPreuMaster;
        becat = pBecat;
    }

    // Mètodes consultors:
    public String getNomMaster() {
            return nomMaster;
    }
    public double getPreuMaster() {
            return preuMaster;
    }
    public boolean getBecat() {
            return becat;
    }

    // Mètodes modificadors:
    public void setNomMaster(String pNomMaster) {
            nomMaster = pNomMaster;
    }
    public void setPreuMaster(double pPreuMaster) {
            preuMaster = pPreuMaster;
    }
    public void setBecat(boolean pBecat) {
            becat = pBecat;
    }
```

```
   //Implementació del mètode abstracte de la classe pare Estudiant
   public void imprimirPreuMatricula() {
      double preuMatricula;
      if(becat){
         preuMatricula= 0;
      }else{
         preuMatricula= preuMaster;
      }
      System.out.println("El preu de la matrícula de l'estudiant de màster " + nom + " és "+
preuMatricula);
          }
}
//*********************************************************************
// Activitat.java
//*********************************************************************
package edu.ub.prog2.activitat;
import java.util.ArrayList;
import java.util.Iterator;

public class Activitat {
   static public void main(String[] args){
       // Creem tres instàncies d'estudiants:
       Estudiant est1 = new EstudiantGrau("Maria", 18, 0001, "Matemàtiques", 4, 100.00);
       Estudiant est2 = new EstudiantGrau("Joan", 20, 0002, "Informàtica", 5, 100.00);
       Estudiant est3 = new EstudiantMaster("Marc", 23, 0003, "Intel·ligència Artificial", 300.00,
true);

        // Creem una llista genèrica d'estudiants:
       ArrayList<Estudiant> llistaInscrits = new ArrayList<Estudiant>();

       // Omplim la llista amb les tres instàncies:
       llistaInscrits.add(est1);
       llistaInscrits.add(est2);
       llistaInscrits.add(est3);

       //Recorrem aquesta llista invocant al mètode imprimirPreuMatricula() per a cada element:
       for(Iterator<Estudiant> i = llistaInscrits.iterator();i.hasNext();){
         i.next().imprimirPreuMatricula();
       }
   }
}
```

Sortida per pantalla:

```
El preu de la matrícula de l'estudiant de grau Maria és 400.0
El preu de la matrícula de l'estudiant de grau Joan és 500.0
El preu de la matrícula de l'estudiant de màster Marc és 0.0
```

Programació 2.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques. UB
Laura Igual

Exercici

A l'Hotel Plaza es desitja desenvolupar un programa que permeti emmagatzemar una llista de clients que s'inscriuen com clients habituals i que com a tals paguen una quota anual. La informació d'interès dels clients és el nom, DNI, telèfon i quota anual. En aquesta llista hi poden haver dos tipus de clients, els clients estàndard i els clients vip. Els primers paguen la totalitat de la quota anual i els segons tenen un descompte del 50% sobre aquest preu.

a) Seguint aquest enunciat, implementa en llenguatge Java les tres classes següents que es necessitaran per al programa:
    a. **Client**,
    b. **ClientEstandard** (classe que hereta de la classe **Client**)
    c. **ClientVip** (classe que hereta de la classe **Client**)

   Inclou a cada classe els atributs (que s'han de deduir de l'enunciat) i els següents mètodes:
    d. Un sol constructor on s'inicialitzen tots els atributs de la classe
    e. Tots els mètodes consultors i modificadors per als atributs de la classe.
    f. Un mètode per imprimir la quota anual, anomenat imprimirQuotaAnual, que serà diferent per les tres classes:
        i. Serà un mètode abstracte de la classe **Client**.
        ii. A la classe **ClientEstandard**, aquest mètode mostrarà per pantalla el nom del client i la seva quota anual.
        iii. A la classe **ClientVip**, aquest mètode imprimirà per pantalla el nom del client i la quota anual aplicant-li el descompte corresponent a aquest tipus de clients.

   Organitza el codi en fitxers amb noms i indica els paquets i els imports necessaris.

b) Implementa una altra classe **GestorClients** amb un mètode main on es creï una llista de clients i s'ompli amb les següents instàncies:
    a. John Smith, amb el DNI 10922456 i telèfon 93 123 45 67 és un client vip.
    b. Peter Adams, amb el DNI 21987410 i telèfon 93 321 78 96 és un client estàndard.

   Considera que la quota anual és de 110.50€.
   Recorre aquesta llista fent servir un iterador i invoca el mètode imprimirQuotaAnual per a cada element.

c) Indica quina és la sortida del mètode main.

Solució:

```
//*********************************************************************
// Client.java
//*********************************************************************
package examen;

public abstract class Client {
	private String nom;
	private int dni;
	private String telefon;
	private double quotaAnual;

	// Constructor
	public Client(String nom, int dni, String telefon, double quotaAnual){
		this.nom = nom;
		this.dni = dni;
		this.telefon = telefon;
		this.quotaAnual = quotaAnual;
	}
```

```
	//Mètode abstracte, es declara sense implementació
	public abstract void imprimirQuotaAnual();

	// Mètodes modificadors:
	public void setNom(String nom){
		this.nom=nom;
	}
	public void setDNI(int dni){
		this.dni=dni;
	}
	public void setTelefon(String telefon){
		this.telefon=telefon;
	}
	public double setQuotaAnual(){
		return quotaAnual;
	}
	// Mètodes consultors:
	public String getNom(){
		return this.nom;
	}
	public String getTelefon(){
		return this.telefon;
	}
	public String getDNI(){
		return this.telefon;
	}
	public double getQuotaAnual(){
		return quotaAnual;
	}
}

//*********************************************************************
// ClientVip.java
//*********************************************************************
package examen;

public class ClientVip extends Client{
	private double descompte;

	// Constructor
	public ClientVip(String nom, int dni, String telefon, double quotaAnual){
		super(nom, dni, telefon, quotaAnual);
		this.descompte = 0.5;
	}
	// Mètode propi d'aquesta classe
	public double calcularQuota() {
		double quota;
		quota = super.getQuotaAnual();
		return quota - (quota*descompte);
	}
	//Implementació del mètode abstracte de la superclasse Client
	public void imprimirQuotaAnual() {
		System.out.println("La quota anual a pagar pel client " + super.getNom() + " és: " +
		this.calcularQuota());
	}
}

//*********************************************************************
// ClientEstandard.java
//*********************************************************************
package examen;
```

```
public class ClientEstandard extends Client{
	// Constructor
	public ClientEstandard(String nom, int dni, String telefon, double quotaAnual){
		super(nom, dni, telefon, quotaAnual);
	}
	//Implementació del mètode abstracte de la superclasse Client
	public void imprimirQuotaAnual() {
		System.out.println("La quota anual a pagar pel client " + super.getNom() + " és: " +
		super.getQuotaAnual());
	}
}

//*********************************************************************
// GestorClients.java
//*********************************************************************
package examen;
import java.util.ArrayList;
import java.util.Iterator;

public class GestorClients {

	public static void main(String[] args){
		String nom;
		int dni;
		String telefon;
		double quotaAnual;
		quotaAnual = 110.50;
		// Creem les dues instàncies que necessitem:
		nom = "John Smith";
		dni = 10922456;
		telefon = "93 123 45 67";
		ClientVip client1 = new ClientVip(nom, dni, telefon, quotaAnual);
		nom = "Peter Adams";
		dni = 21987410;
		telefon = "93 321 78 96";
		ClientEstandard client2 = new ClientEstandard(nom, dni, telefon, quotaAnual);
		// Creem una llista de Clients
		ArrayList<Client> llistaClients = new ArrayList<Client>();
		// Omplim la llista amb les tres instàncies:
		llistaClients.add(client1);
		llistaClients.add(client2);
		//Recorrem aquesta llista invocant al mètode imprimirQuotaAnual() per a cada
		element:
		for(Iterator<Client> i = llistaClients.iterator();i.hasNext();){
		i.next().imprimirQuotaAnual();
		}
	}
}
```

Sortida:

```
La quota anual a pagar pel client John Smith és: 55.25
La quota anual a pagar pel client Peter Adams és: 110.5
```

Exercici

L'aplicació presentada a l'exercici anterior es vol ampliar afegint el control de reserves de l'hotel que fa cada client, així com el control del número de reserves que s'han pagat en total a l'hotel.
a) Per fer això, implementa en llenguatge Java una classe **ReservaHotel** que implementa **una** reserva d'**un** client. Aquesta classe ha de tenir com a mínim un mètode constructor al que li passem un objecte de tipus Client i un mètode anomenat *pagametRealitzat* que incrementa en un la quantitat de reserves pagades.

b) Indica a més el codi que s'hauria d'afegir al mètode main de la classe **GestorClients,** implementat a l'exercici anterior, per tal de que el client Jonh Smith faci dues reserves d'hotel, les pagui i es mostri per pantalla el número de reserves pagades.

Per fer els dos apartats (a) i (b) d'aquest exercici no feu servir ni taules ni arrays.

Solució:

```
//*********************************************************************
// ReservaHotel.java
//*********************************************************************
package examen;

public class ReservaHotel {
	private Client client;
	private static int numReservesPagades;

	public ReservaHotel(Client client){
		this.client = client;
	}
	public void pagametRealitzat (){
		numReservesPagades++;
	}
	// Mètodes consultors:
	public static int getNumReserves(){
		return numReservesPagades;
	}
	public Client getClient(){
		return client;
	}
	// Mètode modificador:
	public void setClient(Client client){
		this.client = client;
	}
}



//*********************************************************************
// GestorClients.java
//*********************************************************************
package examen;

public class GestorClients{

	public static void main(String [] args){
		// Afegir al codi de l'exercici anterior
		ReservaHotel reserva1 = new ReservaHotel(client1);
		reserva1.pagametRealitzat();
		ReservaHotel reserva2 = new ReservaHotel(client1);
		reserva2.pagametRealitzat();
```

```
                System.out.println("Número de Reserves pagades = " +
                ReservaHotel.getNumReserves());
        }
}


//**********************************************************************
Sortida:
//**********************************************************************
Número de Reserves pagades = 2
```