Examen Parcial (Solució)

20 d'Abril de 2016

Nom i Cognoms: ..... NIUB: .....

Contesta els següents exercicis creant les classes i el codi Java que es demana en un projecte de Netbeans. Quan acabis, fes el lliurament a la tasca oberta del Campus Virtual amb un sol fitxer comprimit anomenat: Cognom1Cognom2Nom.

Enunciat:
A un Zoo us demanen desenvolupar un programa que permeti gestionar els animals i la seva cura per part dels empleats cuidadors, que consisteix en donar el menjar i netejar-los segons la necessitat de cada animal. Cada animal tindrà un identificador propi i a més, guardarà la data de la seva pròxima cura, així com els anys que te. En el cas dels tigres, haureu de guardar també el número de litres de llet que se'ls ha de donar per menjar. Cada cuidador, del que disposem el seu nom i NIF, tindrà una llista dels animals dels que s'ha de fer càrrec. A més, tindrem l'opció de consultar l'assignació d'un cuidador per un dia particular.

Seguint aquest enunciat, implementa en llenguatge Java el que es demana en els següents exercicis:

Exercici 1 (5 punts)
Implementa les següents classes **Animal**, **Tigre**, **Serp**, **Cuidador, AssignacioDiaria** i **GestorZoo1** seguint les indicacions de l'enunciat de dalt i les que segueixen a continuació:

A la classe abstracta **Animal** es defineixen els següents mètodes:

- *proximaCura:* Retornarà la data de la pròxima cura a realitzar en format Data (veure **Nota1**).
- *realitzarCura*: Serà un mètode abstracte que serà implementat a les subclasses.

A més, en aquesta classe has de:

- Guardar una variable pel número d'identificador i l'edat de l'animal.
- Guardar la data de la pròxima cura.
- Implementar un sol constructor on s'inicialitzen tots els atributs de la classe amb els valors passats per paràmetre.

A les classes **Tigre** i **Serp** has d'incloure els atributs (que s'han de deduir de l'enunciat) i implementar els següents mètodes:

- Un sol constructor on s'inicialitzen tots els atributs de la classe.
- El mètode *realitzarCura*:
    1. A la classe **Tigre**, aquest mètode mostrarà per pantalla la frase "La cura del tigre" seguida pel identificador del tigre, "s'ha de realitzar la data" seguida per la data de la seva pròxima cura "i donant" seguit dels litres de llet.
    2. A la classe **Serp**, aquest mètode mostrarà per pantalla la frase "La cura de la serp" seguida pel identificador de la serp, "s'ha de realitzar la data" seguida per la data de la seva pròxima cura.
- El mètode *toString*, que retornarà un String amb la frase:
    1. A la classe **Tigre**, "Tigre amb identificador: " seguit del número del identificador.
    2. A la classe **Serp**, "Serp amb identificador: " seguit del número del identificador.

La classe **Tigre** necessita consultors i modificadors mentre que en la classe **Serp** no caldrà.

En la classe **Cuidador** s'ha de poder afegir un animal a la llista d'animals assignats al cuidador. Implementa aquesta llista fent servir la classe ArrayList de Java. A més, s'han d'implementar els mètodes consultors i modificadors necessaris pels atributs de les classes, inclosa la llista de tots els animals assignats.

La classe **AssignacioDiaria** ha d'incloure com atributs:

- Un cuidador
- Una data
- Una llista d'animals assignats per aquesta data (ArrayList).

I definir els següents mètodes:

- *emplenarAssignacioDia*: que itera pels animals assignats a un cuidador i comprova quin està assignat per aquest dia per incloure'l a la llista de l'assignació diària. Aquesta comprovació es fa cridant al mètode *isSameDate* de **Data**.
- Un constructor que rep un cuidador i una data i que invoca al mètode *emplenarAssignacioDia* per emplenar la instància creada de la llista d'animals del dia.
- *toString*, que retornarà un string amb la frase: "El cuidador " seguit del nom " te una assignació pel dia " seguit de la data " dels següents animals" seguit de la llista d'animals de l'assignació diària. Per concatenar aquesta última informació has d'iterar pels animals de la llista de l'assignació diària.

En la classe **GestorZoo1** implementa un mètode main on es creï un cuidador anomenat "Pere Andreu" amb NIF "12345678E" i se li assignin la cura dels següents animals:

1. Un tigre amb identificador 1 amb la data de pròxima cura el "15/05/2016", amb 5 anys d'edat i amb 1.5 litres de llet per menjar.
2. Un tigre amb identificador 2 amb la data de pròxima cura el "28/04/2016", amb 2 anys d'edat i amb 2 litres de llet per menjar.
3. Una serp amb identificador 3 amb la data de pròxima cura el "06/05/2016" i amb 4 anys d'edat.
4. Una serp amb identificador 4 amb la data de pròxima cura el "20/04/2016" i amb 1 anys d'edat.

Un cop introduïdes les instàncies, afegeix al mètode main les següents accions:

- Crea una assignació diària del cuidador pel dia d'avui
- Mostra aquesta assignació diària d'avui per pantalla.

Nota1: Utilitza la classe **Data** implementada en Data.java i disponible al Campus Virtual.
Nota2: Per iterar sobre les llistes de tipus ArrayList fes servir un iterador.

Solució:

Animal.java
****

```
package ub.edu.prog2;

public abstract class Animal {
   private int id;
   private Data proximaCura;
   private int edat;

   public Animal(int id, Data proximaCura, int edat)
   {
      this.id = id;
      this.proximaCura = proximaCura;
      this.edat = edat;
   }

   public int getID()
   {
      return this.id;
```

```
    }

    public Data proximaCura()
    {
        return this.proximaCura;
    }

    protected abstract void realitzarCura();
}
```

Serp.java
****

```
package ub.edu.prog2;

import java.util.Date;

public class Serp extends Animal {

    public Serp(int id, Data proximaCura, int edat)
    {
        super(id,proximaCura,edat);
    }

    @Override
    public void realitzarCura()
    {
        System.out.println("La cura de la serp " + this.getID() + " s'ha de realitzar la data " +
this.proximaCura() + ".");
    }

    @Override
    public String toString()
    {
        return "Serp amb identificador: " + this.getID();
    }
}
```

Tigre.java
****

```
package ub.edu.prog2;

public class Tigre extends Animal {
    private float litresLlet;

    public Tigre(int id, Data proximaCura, int edat, float litresLlet)
    {
        super(id,proximaCura,edat);
        this.litresLlet = litresLlet;
    }

    public float getLitresLlet()
    {
        return this.litresLlet;
    }
```

```
    public void setLitresLlet(float litresLlet)
    {
        this.litresLlet = litresLlet;
    }

    @Override
    public void realitzarCura()
    {
        System.out.println("La cura del tigre " + this.getID() + " s'ha de realitzar la data " + this.proximaCura() + " i donant-li " + this.litresLlet + " litres de llet.");
    }

    @Override
    public String toString()
    {
        return "Tigre amb identificador: " + this.getID();
    }
}
```

Cuidador.java
****

```
package ub.edu.prog2;

import java.util.ArrayList;
import java.util.Iterator;

public class Cuidador {
    private String nom;
    private String nif;
    private ArrayList<Animal> animalsAssignats;

    public Cuidador(String nom, String nif)
    {
        this.nom = nom;
        this.nif = nif;
        this.animalsAssignats = new ArrayList<>();
    }

    public String getNom()
    {
        return this.nom;
    }

    public String getNIF()
    {
        return this.nif;
    }

    public ArrayList<Animal> getAnimals()
    {
        return this.animalsAssignats;
    }

    public void getNom(String nom)
    {
        this.nom = nom;
    }
```

```
    public void getNIF(String nif)
    {
        this.nif = nif;
    }

    public void setAnimals(ArrayList<Animal> animals)
    {
        this.animalsAssignats = animals;
    }

    public void afegirAnimal(Animal animal) {
        this.animalsAssignats.add(animal);
    }

}
```

AssignacioDiaria.java
****

```
package ub.edu.prog2;

import java.util.ArrayList;
import java.util.Iterator;

public class AssignacioDiaria {
    private Cuidador cuidador;
    private Data data;
    private ArrayList<Animal> assignacio;

    public AssignacioDiaria(Cuidador c, Data d) {
        this.cuidador = c;
        this.data = d;
        this.assignacio = new ArrayList<>();
        this.emplenarAssignacioDiaria();
    }

    public void emplenarAssignacioDiaria() {
        Iterator<Animal> iter = this.cuidador.getAnimals().iterator();
        while (iter.hasNext()) {
            Animal a = iter.next();
            if (data.isSameDate(a.proximaCura())) {
                assignacio.add(a);
            }
        }
    }

    @Override
    public String toString()
    {
        String s = "El cuidador " + this.cuidador.getNom() + " te una assignacio pel dia " + this.data
+ " dels següents animals: \n";
        Iterator<Animal> iter = this.assignacio.iterator();
        while (iter.hasNext()) {
            s += iter.next().toString() + '\n';
        }

        return s;
    }
```

```
}
```

GestorZoo1.java
****

```
package ub.edu.prog2;

public class GestorZoo1 {

   public static void main(String[] args) {
      Cuidador cuidador = new Cuidador("Pere Andreu", "12345678E");

      Tigre tigre1 = new Tigre(1, new Data("15/05/2016"), 5, 1.5f);
      Tigre tigre2 = new Tigre(2, new Data("18/04/2016"), 2, 2f);
      Serp serp1 = new Serp(3, new Data("06/05/2016"), 4);
      Serp serp2 = new Serp(4, new Data("20/04/2016"), 1);

      cuidador.afegirAnimal(tigre1);
      cuidador.afegirAnimal(tigre2);
      cuidador.afegirAnimal(serp1);
      cuidador.afegirAnimal(serp2);

      AssignacioDiaria assignacioDiaria = new AssignacioDiaria(cuidador, Data.getData());
      System.out.println(assignacioDiaria.toString());
   }
}
```

Sortida per pantalla:

```
El cuidador Pere Andreu te una assignació pel dia 20/04/2016 dels següents animals:
Serp amb identificador: 4
```

Exercici 2 (1.5 punts)

a) Afegeix els següents canvis en el codi:

- Implementa una classe **GestorZoo2** amb un mètode *main* on es creï el mateix cuidador que en la classe **GestorZoo1** i se li assignin la cura dels mateixos animals.
- Implementa un mètode a la classe **Cuidador**, anomenat *realitzarTotesCures*, que recorri els animals que te assignat, fent servir un iterador, per realitzar-los la cura, invocant el mètode *realitzarCura* per a cada element de la llista.
- Invoca aquest mètode *realitzarTotesCures* des del *main* de la classe **GestorZoo2.**

b) En un full a part, contesta a la següent pregunta sobre el codi implementat: Es fa servir el polimorfisme en el mètode *realitzarTotesCures* de la classe **Cuidador**? En cas afirmatiu, explica on exactament i com s'observa.

Solució:

a) Nou mètode a la classe Cuidador:

Cuidador.java
****

```
public class Cuidador {
   public void realitzarTotesCures()
   {
      Iterator<Animal> iter = this.animalsAssignats.iterator();
      while (iter.hasNext())
      {
         iter.next().realitzarCura();
```

```
      }
   }
}
```

GestorZoo2.java
****

```
package ub.edu.prog2;

  public class GestorZoo2 {
     public static void main(String[] args) {

        Cuidador cuidador = new Cuidador("Pere Andreu", "12345678E");

        Tigre tigre1 = new Tigre(1, new Data("15/05/2016"), 5, 1.5f);
        Tigre tigre2 = new Tigre(2, new Data("11/04/2016"), 2, 2f);
        Serp serp1 = new Serp(3, new Data("06/05/2016"), 4);
        Serp serp2 = new Serp(4, new Data("20/04/2016"), 1);

        cuidador.afegirAnimal(tigre1);
        cuidador.afegirAnimal(tigre2);
        cuidador.afegirAnimal(serp1);
        cuidador.afegirAnimal(serp2);

        cuidador.realitzarTotesCures();
     }
  }
```

Sortida per pantalla:

La cura del tigre 1 s'ha de realitzar la data 15/05/2016 i donant-li 1.5 litres de llet.
La cura del tigre 2 s'ha de realitzar la data 11/04/2016 i donant-li 2.0 litres de llet.
La cura de la serp 3 s'ha de realitzar la data 06/05/2016.
La cura de la serp 4 s'ha de realitzar la data 20/04/2016.

b)

Sí, a la línia: itr.next().realitzarCura();
S'observa en que la sortida per pantalla depèn de la instància que conte la llista (tigre o serp).

Exercici 3 (1.5 punts)

a) Afegeix els següents canvis en el codi:

- Implementa una classe **GestorZoo3** amb un mètode *main* on es creï el mateix cuidador que en la classe **GestorZoo1** i se li assignin la cura dels mateixos animals.
- Implementa un mètode a la classe **Cuidador**, anomenat *alletarTigres*, que recorri els animals que té assignats, fent servir un iterador, i mostri per pantalla la quantitat de litres de llet que necessita cada tigre de la llista.
- Invoca aquest mètode *alletarTigres* des del *main* de la classe **GestorZoo3.**

b) En un full a part, contesta a la següent pregunta sobre el codi implementat: Es fa servir el polimorfisme en el mètode *alletarTigres* de la classe **Cuidador**? En cas afirmatiu, explica on exactament i com s'observa.

Solució:

a)
Nou mètode a la classe Cuidador:

Cuidador.java
****

```
public class Cuidador {
void alletarTigres() {
      Iterator<Animal> iter = this.animalsAssignats.iterator();
      while (iter.hasNext())
      {
         Animal a = iter.next();
         if (a instanceof Tigre)
            System.out.println("Tigre " + ((Tigre)a).getID() + " necessita " + ((Tigre)
a).getLitresLlet());
      }
   }
}
```

GestorZoo3.java
****

```
package ub.edu.prog2;

public class GestorZoo3 {
   public static void main(String[] args) {

      Cuidador cuidador = new Cuidador("Pere Andreu", "12345678E");

      Tigre tigre1 = new Tigre(1, new Data("15/05/2016"), 5, 1.5f);
      Tigre tigre2 = new Tigre(2, new Data("11/04/2016"), 2, 2f);
      Serp serp1 = new Serp(3, new Data("06/05/2016"), 4);
      Serp serp2 = new Serp(4, new Data("20/04/2016"), 1);

      cuidador.afegirAnimal(tigre1);
      cuidador.afegirAnimal(tigre2);
      cuidador.afegirAnimal(serp1);
      cuidador.afegirAnimal(serp2);

      cuidador.alletarTigres();
   }
}
```

Sortida per pantalla:

```
Tigre 1 necessita 1.5
Tigre 2 necessita 2.0
```

b) No. Fem servir instanceof.

Exercici 4 (2 punts)
Indica quin terme es defineix en les següents definicions:

a) Col·lecció de declaracions de mètodes abstractes que defineixen un comportament, però no l'implementa. Permet implementar herència múltiple en Java.
Interfície

b) Acció de demanar a un altre objecte la resolució, total o parcial, d'un mètode.
Delegació

c) Tècnica de conversió de dades que ens permet utilitzar una instància d'una classe o tipus bàsic de dades com si es tractés d'una instància d'un altre classe o tipus de dades. Permet conversions d'ampliació i reducció.
Casting

d) Ocultació de la implementació concreta d'una classe en què s'ofereixen només aquelles dades que es vol que les altres classes puguin utilitzar. És una característica dels mòduls que permet abstreure la gestió de la informació d'aquests de la representació interna de la informació que contenen.
Encapsulament