Programación 3

Actividades de clase

Universitat d'Alacant
Universidad de Alicante

Unidad 1: introducción a la programación orientada a objetos

Problema 1:

Considera esta clase de Java que representa puntos en un espacio bidimensional:

```
public class Point {
  private double x, y;
  public Point() {
    x= y= 1.0;
  }
  public void setX(double newX) {
    x= newX;
  }
  public void setY(double newY) {
    y= newY;
  }
  public double getX() {
    return x;
  }
  public double getY() {
    return y;
  }
}
```

Y considera el código cliente que crea dos objetos de la clase e invoca al setter correspondiente:

```
class Main {
  public static void main(String[] args) {
    Point p1= new Point();
    Point p2= new Point();
    p1.setX(2);
    p2.setX(4);
    System.out.println(p1.getX());
    System.out.println(p2.getX());
  }
}
```

¿Cuántas veces está el código del método *set*X en la memoria de la máquina virtual de Java cuando se ejecuta el programa? Si el código no está duplicado en memoria, ¿cómo es posible que unas veces modifique el atributo de *p1* y otras el de *p2*?

Problema 2:

¿Qué ocurre si una clase de Java no define ningún constructor? Inspírate en este código para elaborar tu respuesta:

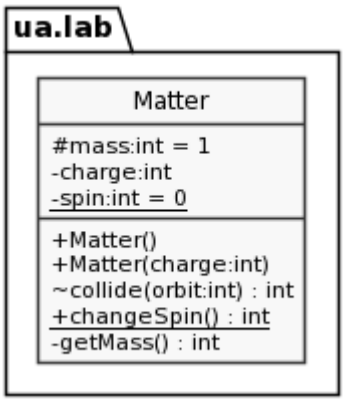
```
public class Point {
  private double x, y;

  public void setX(double newX) {
    x= newX;
  }
  public void setY(double newY) {
    y= newY;
  }
  public double getX() {
    return x;
  }
  public double getY() {
    return y;
  }
}
//código cliente
Point p = new Point();
System.out.println(p.getX()); // 0.0
```

Problema 3:

Crea en Java una clase A con un constructor sin parámetros que lleve la cuenta del número de instancias de dicha clase que existen en memoria en cada momento. Añade a la clase un método que devuelva el valor de esta cuenta. ¿Cuándo se destruyen los objetos? ¿Cómo nos podemos asegurar de que la cuenta está siempre actualizada?

Problema 4:

Escribe el código de Java correspondiente a la clase representada mediante el siguiente diagrama de clases de UML. Observa que hay un atributo y un método de clase (aparecen subrayados en el diagrama UML) y que la clase pertenece al paquete *ua.lab*.



Problema 5:

Un objeto inmutable en Java es aquel cuyo estado no puede cambiar una vez que el objeto ha sido creado. Si el programador desea un objeto con un estado diferente ha de crear uno nuevo *clonando* el existente. ¿Qué características de una clase de las que has estudiado hasta ahora permiten que sus objetos sean inmutables? Céntrate en cómo definirías una clase Circle que permita crear instancias inmutables de círculos. Después, contesta a esta otra pregunta: ¿qué parte de tu respuesta cambiaría si la clase tuviera únicamente atributos estáticos?

Problema 6:

Considera la siguiente clase de Java (basada en este código) que permite gestionar una cola.

```
class Queue {
  private int front, rear, capacity;
  private int queue[];

  Queue(int c) {
    front = rear = 0;
    capacity = c;
    queue = new int[capacity];
  }

  // insert an element at the rear of the queue
  void queueEnqueue(int data) {
    if (capacity == rear) {
      // TODO: throw an exception
    } else {
      queue[rear] = data;
      rear++;
    }
    return;
```

```
    }

    // delete an element from the front of the queue
    void queueDequeue() {
      if (front == rear) {
        // TODO: throw an exception
      } else {
        for (int i = 0; i < rear - 1; i++) {
          queue[i] = queue[i + 1];
        }
        rear--;
      }
      return;
    }

    // return front of the queue
    int queueFront() {
      if (front == rear) {
        // TODO: throw an exception
        return -1;
      }
      else
        return queue[front];
    }
  }
```

Esta estructura de datos puede verse como una lista a la que se añaden valores al final y de la que solo se puede consultar y extraer el primer elemento. En esta figura de la web GeeksforGeeks puedes ver esquemáticamente el proceso de encolar dos veces y desencolar una.

Implementa, por un lado, un constructor de copia superficial y, por otro lado, uno de copia profunda para la clase. ¿Cuántos bytes se reservan en memoria directa o indirectamente con uno u otro constructor de copia por la ejecución de la línea marcada con el comentario “here”? Asume que los enteros ocupan 4 bytes en la memoria de la máquina virtual de Java (JVM) y que las referencias ocupan 8 bytes.

```
public class Main {
  public static void main(String[] args) {
    Queue q = new Queue(4);
    q.queueEnqueue(20);
    q.queueEnqueue(30);
    q.queueEnqueue(40);
    q.queueEnqueue(50);
    q.queueDequeue();
    q.queueDequeue();
```

```
        System.out.println(q.queueFront());
        Queue q2= new Queue(q);  // here
        System.out.println(q2.queueFront());
    }
}
```