Programación II - 2010/2011 - Universitat Jaume I
Evaluación continua - Módulo 2 de teoría - Grupo A

19 de abril de 2011

La duración máxima de esta prueba es de 75 minutos. No puedes consultar libros ni apuntes.

Ejercicio 1 (3.5 puntos, 0.5 puntos por apartado)

Imagina que hubiésemos implementado así las clases `Punto` y `Circulo`. Ten en cuenta que hay importantes diferencias respecto de la implementación que conoces y que, en este ejercicio, se te pide pensar las consecuencias de una implementación que puede ser antinatural.

```
public class Punto {
    double x, y;
    public Punto(double laX, double laY) {
        x = laX;
        y = laY;
    }
    public Punto(Punto otroPunto) {
        x = otroPunto.x;
        y = otroPunto.y;
    }
    public void intercambiarXY() {
        double aux = x;
        x = y;
        y = aux;
    }
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}


public class Circulo {
    Punto centro;
    double radio;
    public Circulo(Punto elCentro, double elRadio) {
        centro = new Punto(elCentro);
        radio  = elRadio;
    }
    public Circulo(Circulo otroCirculo) {
        centro = otroCirculo.centro;
        radio  = otroCirculo.radio;
    }
    public void transformar() {
        centro.intercambiarXY();
        radio = radio * 3;
    }
    public void ver(String nombre) {
        System.out.println(nombre + " = " + centro + " -- " + radio);
    }
}
```

Indica lo que se escribiría en la salida estándar al ejecutar cada uno de los siguientes programas utilizando esas versiones. No es necesario que expliques por qué.

(a)
```
public class EjercicioA {
    public static void main (String[] args) {
        Circulo c1 = new Circulo(new Punto(2,3), 4);
        Circulo c2 = c1;
        c2.transformar();
        c1.ver("c1");
        c2.ver("c2");
    }
}
```

(b)
```
public class EjercicioB {
    public static void main (String[] args) {
        Circulo c1 = new Circulo(new Punto(2,3), 4);
        Circulo c2 = new Circulo(c1);
        c2.transformar();
        c1.ver("c1");
        c2.ver("c2");
    }
}
```

(c)
```
public class EjercicioC {
    public static void main (String[] args) {
        Punto p1 = new Punto(2,3);
        Circulo c1 = new Circulo(p1, 4);
        Circulo c2 = new Circulo(p1, 5);
        c2.transformar();
        c1.ver("c1");
        c2.ver("c2");
    }
}
```

(d)
```
public class EjercicioD {
    public static void transformarCirculo(Circulo c) {
        c.transformar();
        c.ver("c");
    }
    public static void main (String[] args) {
        Circulo c1 = new Circulo(new Punto(2,3), 4);
        transformarCirculo(c1);
        c1.ver("c1");
    }
}
```

(e)
```
public class EjercicioE {
    public static void intercambiarCirculos(Circulo c1, Circulo c2) {
        Circulo aux = c1;
        c1 = c2;
        c2 = aux;
    }
    public static void main (String[] args) {
        Circulo [] v = new Circulo[2];
        v[0] = new Circulo(new Punto(2,3), 4);
        v[1] = new Circulo(new Punto(5,6), 7);
        intercambiarCirculos(v[0], v[1]);
        v[0].ver("v[0]");
        v[1].ver("v[1]");
    }
}
```

(f)
```
public class EjercicioF {
    public static void intercambiarCirculos(Circulo [] dosCirculos) {
        Circulo aux = dosCirculos[0];
        dosCirculos[0] = dosCirculos[1];
        dosCirculos[1] = aux;
    }
    public static void main (String[] args) {
        Circulo [] v = new Circulo[2];
        v[0] = new Circulo(new Punto(2,3),4);
        v[1] = new Circulo(new Punto(5,6),7);
        intercambiarCirculos(v);
        v[0].ver("v[0]");
        v[1].ver("v[1]");
        intercambiarCirculos( v.clone() );
        v[0].ver("v[0]");
        v[1].ver("v[1]");
    }
}
```

(g)
```
public class EjercicioG {
    public static Circulo primerCirculo(Circulo [] circulos) {
        return circulos[0];
    }
    public static void main (String[] args) {
        Circulo [] v = new Circulo[2];
        v[0] = new Circulo(new Punto(2,3),4);
        v[1] = new Circulo(new Punto(5,6),7);
        Circulo c1 = primerCirculo(v);
        c1.transformar();
        v[0].ver("v[0]");
        v[1].ver("v[1]");
        Circulo c2 = primerCirculo( v.clone() );
        c2.transformar();
        v[0].ver("v[0]");
        v[1].ver("v[1]");
    }
}
```

Ejercicio 2 (6.5 puntos)

Implementa una clase `RecorridoTren` para guardar la información de las ciudades en las que tiene parada un tren y las horas de parada previstas. Incluye en la clase métodos públicos que permitan:

1. Crear un nuevo recorrido, inicialmente vacío.
2. Añadir una nueva parada al recorrido, dando el nombre de la ciudad y la hora.
3. Buscar la ciudad en la que se realiza una parada dando la hora de la parada. Si no se encuentra, debe devolver `null`. Puedes suponer que ninguna hora aparece más de una vez.
4. Comparar dos recorridos para comprobar si son iguales empleando el método `equals`. Considera que dos recorridos son iguales si tienen las mismas paradas, aunque no se hayan añadido en el mismo orden. Puedes suponer que ningún recorrido contiene ciudades ni horas repetidas. No olvides que cualquiera de los dos recorridos puede estar vacío.
5. Crear un nuevo recorrido, copiando los datos de un recorrido existente de modo tal que cualquier posible transformación que se haga posteriormente con uno de los recorridos no afecte a los datos del otro recorrido, incluyendo las transformaciones mediante métodos que se añadan en el futuro a la clase (por ejemplo, para modificar la hora de una parada o para eliminar una parada).

Puedes elegir la representación interna de los datos que prefieras. Por ejemplo, puedes tener un vector de cadenas para guardar los nombres de las ciudades y un vector de horas en el que la componente i-ésima guarde la hora de parada en la ciudad i-ésima. Alternativamente, puedes tener un vector de paradas, en el que cada parada contenga el nombre de la ciudad y la hora. También puedes elegir si quieres tener las paradas ordenadas de menor a mayor hora o no. Indica cuál es el coste computacional de cada uno de los métodos con la solución que escojas, y comenta brevemente si conoces una solución más eficiente (aunque no es necesario que implementes la solución más eficiente ahora).

Puedes utilizar la clase `String` de Java. Puedes suponer que dispones de una implementación de la clase `Hora` con los métodos que necesites, pero debes indicar de qué métodos se trata. No puedes suponer que dispones de la clase `Parada`, si quieres usarla debes implementarla. Puedes hacer uso de otros métodos auxiliares de la clase `RecorridoTren` si proporcionas su implementación.