Departamento de Enxeñeria Telemática
Universidade de Vigo

10 de enero de 2012

PROGRAMACIÓN II **(Duración del examen: 2 horas y 30 minutos)**

NOMBRE:

Cuestión 1. **(0.5 puntos)**

Se tiene el código correcto de la siguiente clase:

```
public class Punto2D {
  public int coordX;
  public int coordY;

  public Punto2D (int coordX, int coordY) {
    this.coordX = coordX;
    this.coordY = coordY;
  }
}
```

Se construye, a partir de la clase anterior, una que gestione ubicaciones en 3 dimensiones, y se hace de la forma siguiente:

```
public class Punto3D extends Punto2D{
  public int coordZ;
}
```

La compilación de esta última clase provoca un error. ¿Qué error es?

Respuesta:

Utilice el espacio disponible en el cuadro de respuesta para añadir el código que evitaría dicho error, indicando a qué clase se le agregaría:

Respuesta:

Cuestión 2. (0.5 puntos)

Se tiene la siguiente estructura del código de la clase `Viaje`:

```
public class Viaje{
  private String origen;
  private String destino;
  private int distancia;

  public Viaje (String origen, String destino, int distancia){



  }
  public String getOrigen () {

  }
  public String getDestino () {

  }
  public int getDistancia () {

  }
  public void setOrigen (String origen) {

  }
  public void setDestino (String destino) {

  }
  public void setDistancia (int distancia) {

  }
}
```

Termine de implementar los métodos de la clase utilizando los huecos que se han dejado libres para ello.

Cuestión 3. (0.5 puntos)

Asumiendo que el código de la clase `Viaje` de la cuestión anterior tiene los métodos implementados correctamente, escriba el código de un método denominado une`Viaje` que reciba como parámetros dos objetos de la clase `Viaje` y devuelva un objeto de la clase `Viaje`. Este objeto retornado tendrá como origen el origen del primer parámetro, como destino el destino del segundo parámetro y como distancia la suma de las distancias de los dos objetos `Viaje` que recibe como parámetro, siempre y cuando el origen del segundo parámetro sea idéntico al destino del primero; si esta condición no se cumple el resultado del método será `null`.

Respuesta:

Respuesta (cont):

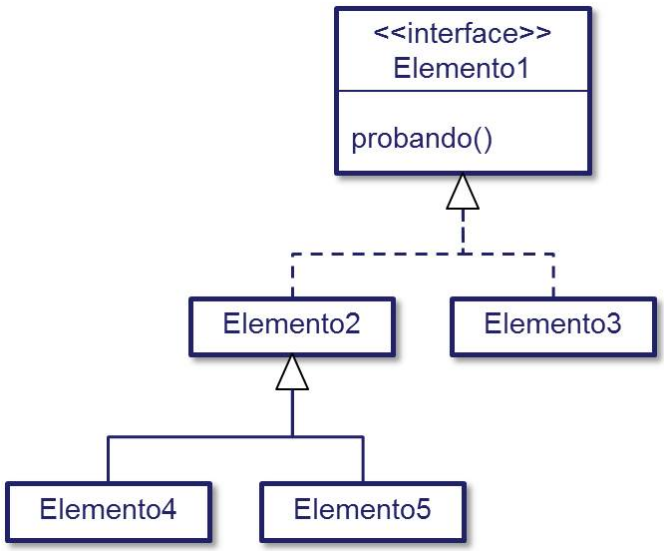
Cuestión 4. (0.5 puntos)

Modifique la implementación anterior para utilizar un objeto anónimo:

Respuesta :

Cuestión 5. (0.5 puntos)

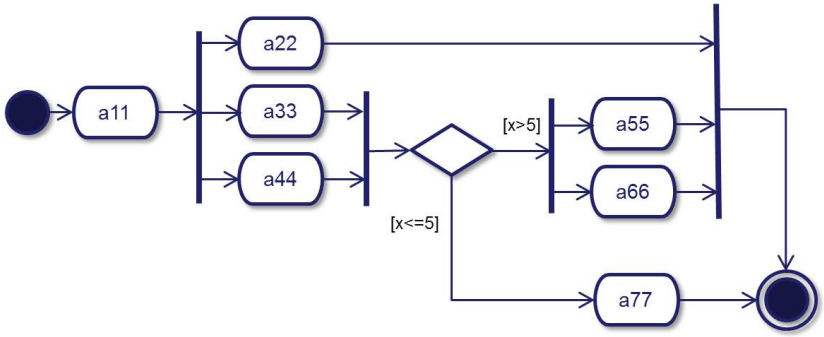
¿Cuáles de las siguientes afirmaciones son correctas sobre el método `probando()`?



- El método `probando()` debe ser implementado por la clase Elemento3 y, posiblemente, por la clase Elemento4
- El método `probando()` debe ser implementado sólo por la clase Elemento5.
- El método `probando()` debe ser implementado por las clases Elemento2, Elemento3, Elemento4 y Elemento5.
- Ninguna de las clases tiene obligación de implementar el método `probando()`, dado que ya está implementado por Elemento1.

Cuestión 6. (0.5 puntos)

Según el siguiente diagrama ¿cuáles de las actividades pueden ocurrir simultáneamente?



- a44 y a66
- a44, a33 y a22
- a22 y a77
- a77 y a66

Cuestión 7. (0.5 puntos)

Indicar qué es y qué hace el siguiente método:

```
import java.util.*;
public class Clase {
  public static int metodo(Iterator<Integer> i) {
    if (!i.hasNext()) return 0;
    else {
      return ((Integer)i.next()).intValue() + metodo(i);
    }
  }
}
```

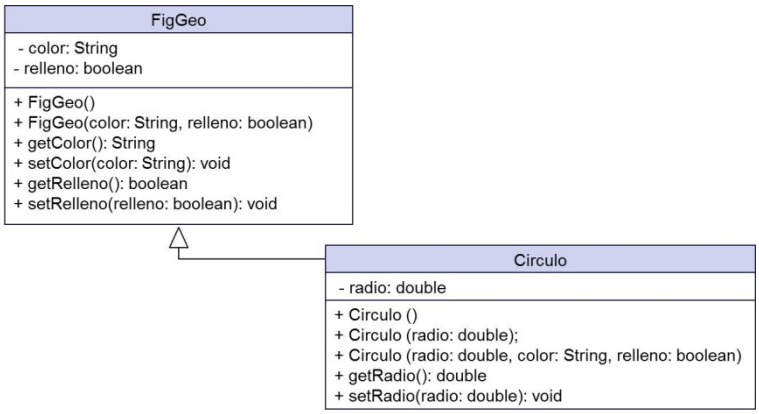
- Es un método recursivo que devuelve la suma de todos los elementos contenidos en un objeto que implementa la Interfaz Iterator y contiene enteros
- Es un método recursivo que iguala a cero todos los elementos contenidos en un objeto que implementa la interfaz Iterator y contine enteros
- Es un método recursivo que no hace nada porque hay un error en la definición del caso base
- Es un método iterativo que suma a cada número entero contenido en un objeto que implementa la interfaz Iterator todos los números anteriores

Cuestión 8. (0.5 puntos)

A qué tipo de proyectos están dirigidos los métodos ágiles como SCRUM:

- Proyectos grandes en los que están involucrados muchos grupos de trabajo y personas
- Proyectos pequeños en los que está involucrado un grupo pequeño de trabajo
- Proyectos grandes en los que está involucrado un grupo pequeño de trabajo
- Proyectos en los que se suelen cambiar los grupos de trabajo involucrados

Problema 1. (2.5 puntos)



Se tiene la relación entre las clases `FigGeo` y la clase `Circulo` que se indica en el diagrama. Asumiendo que la clase `FigGeo` está correctamente implementada:

1. Escriba el código de la clase `Circulo`. Los métodos constructores deberán hacer uso de los métodos constructores de su superclase. Por defecto, se crean objetos cuyo radio sea la unidad. La implementación de la clase `Circulo`, al igual que la de su superclase, deberá pertenecer al paquete `Problema1`. El método constructor que recibe un único parámetro, imprimirá también por pantalla el color del objeto. **(0.5 puntos)**

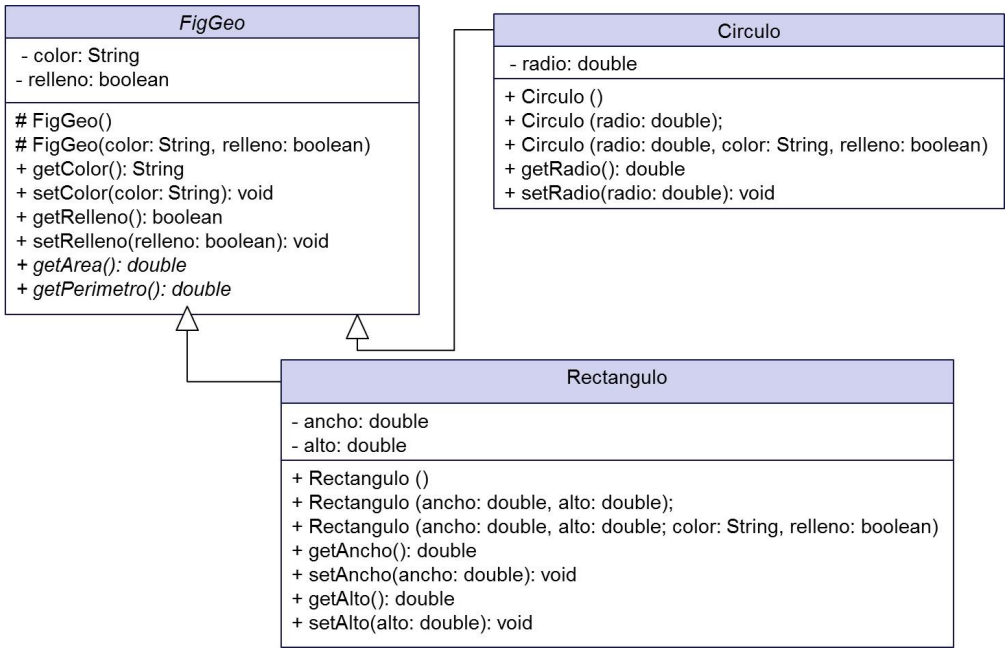
Respuesta:

2. Implemente el método `main`, único método de la clase `Prueba` que, perteneciendo al mismo paquete que las clases anteriores, tenga como funcionalidad la siguiente: **(0.5 puntos)**
    a. Crear el objeto `miCirculo` de la clase `Circulo`, de radio 70, con relleno y de color Rojo.

b. Imprimir por pantalla los tres parámetros definitorios del objeto (radio, color y relleno).

Respuesta:

3. La definición de la clase `FigGeo` se ha modificado, además se ha agregado una nueva clase `Rectangulo`, tal y como se indica en el siguiente diagrama de clases. Complete la implementación de la clase `Circulo` para que ésta sea una clase concreta: **(0.5 puntos)**



Respuesta:

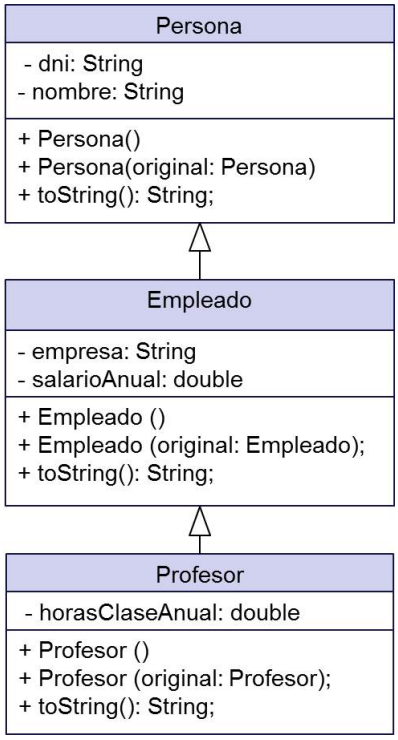
4. Amplíe la clase `Prueba` (apartado 2) para que implemente un método que reciba como parámetros dos figuras geométricas y que sea capaz de decidir si sus áreas son idénticas: devolverá un valor `true` si lo son y un valor `false` en caso contrario. Este método debería trabajar correctamente con cualquier figura geométrica, aunque se decida ampliar el diagrama de clases (p. ej. agregando la clase cuadrado, la clase hexágono, la clase pentágono, etc.) **(0.5 puntos)**

Respuesta:

5. Se desea tener un contador de todas las figuras geométricas que existen simultáneamente. Implemente una solución que utilice la variable `static int numFigGeo`. En el hueco reservado escriba el código de la declaración de dicha variable y la implementación de los métodos donde se utilice, indicando a qué clases pertenecen. Implemente también un método denominado `getNumFigGeo()` que devuelva el número de figuras geométricas. **(0.5 puntos)**

   Respuesta:

Problema 2. (1.5 puntos)



1. Implemente el código del método `toString` de las 3 clases. Se valorará la optimización de instrucciones. **(0.5 puntos)**
Respuesta:

2. Implemente el código del constructor de copia de la clase `Profesor` asumiendo que están implementados los constructores de copia de las otras dos clases, que deberá utilizar en su implementación. **(0.5 puntos)**
Respuesta:

3. Dado el siguiente fragmento de código, y asumiendo que todos los métodos han sido correctamente implementados, indique cuáles, de las afirmaciones siguientes, son correctas: **(0.5 puntos)**

```
. . .
Profesor miProfesor = new Profesor();
miProfesor.setNombre("Pepito Pérez");
miProfesor.dni(3546213465H);
Profesor tuProfesor = new Profesor(miProfesor);
Profesor suProfesor;
suProfesor = tuProfesor;
. . .
```

- Hay 3 objetos de la clase `Profesor` en el *heap*
- Todos los objetos de la clase `Profesor` en el *heap* tienen el mismo contenido
- El código no compila porque no se ha creado correctamente el objeto `suProfesor`
- El código no compila porque no se ha sobrecargado adecuadamente el operador "=" en la última instrucción.

Problema 3. (2 puntos)

Tenemos una empresa constructora que dispone de varios tipos de vehículos: motocicletas, automóviles, furgonetas, camiones, vehículos especiales, etc. Cada uno de estos vehículos tiene un coste distinto por kilómetro en circulación por autopista. Como excepción, los vehículos especiales no pueden circular por autopista.

Para modelar este escenario se plantea la utilización de la jerarquía de clases que se presenta a continuación. Merece una atención especial la interfaz `Autopistable`, que contiene un único método llamado `getCosteAPPKm()` que nos permite obtener el coste por kilómetro por autopista de un vehículo determinado.

Asumiendo que se dispone de la programación completa de todas las clases indicadas y sus métodos se pide:

1. Hacer el diagrama de clases correspondiente en UML
   Respuesta:

2. Completar el siguiente programa `main` para que calcule el consumo medio por kilómetro de autopista de todos los vehículos que pueden ir por una autopista de una empresa.

Como parámetro de entrada se pasa el nombre de la empresa como único argumento de línea de comando.
Respuesta:

```
public class FlotaEmpresa {
  public static void main(String[] args) {
    int numTotalVehiculos = 0;
    double costeMedioAPPKm = 0;

    Vehiculo[] flota = getTotalVehiculos(args[0]);
```

```
    System.out.printf("Coste medio: %4.2f\n", costeMedioAPPKm);
  }
  public static Vehiculo[] getTotalVehiculos(String nombreEmpresa) {
        // Devuelve un array con todos los tipos de vehículos de la empresa
  }
  public static int getNumVehiculos(String nombreEmpresa, Vehiculo v){
  // Devuelve el número de vehículos que la empresa tiene del tipo v
  }
}
  public interface Autopistable {
       public double getCosteAPPKm();
}
public class Vehiculo {
   // Se omiten la mayoría de campos de datos, constructores y métodos
       private String matricula;
       private String tipoCombustible;
       private double consumoCombustiblePKm;
       public double getConsumoCombustiblePKm(){};
}
public class Utilitario extends Vehiculo implements Autopistable {
       private int numAsientos;
       public double getCosteAPPKm() {
              return 0.20;
       }
}
public class Moto extends Utilitario {
}
public class Automobil extends Utilitario {
}
public class Comercial extends Vehiculo {
       private double cargaMaxima;
       private double pesoVacio;
}
public class Furgoneta extends Comercial implements Autopistable {
       public double getCosteAPPKm() {
              return 0.30;
```

```
        }
}
public class Camion extends Comercial implements Autopistable {
  public double getCosteAPPKm() {
    return 0.40;
  }
}
public class Especial extends Vehiculo {
        // No implementa la interfaz Autopistable
}
```