14 de diciembre de 2015

PROGRAMACIÓN II **(Duración del examen: 2,5 horas)**

NOMBRE:

Cuestión 1. **(0.5 puntos)**

Considere el código mostrado a continuación.

```
public class Prueba{
	public static int actualiza (Contador c, int j){
		c = new Contador (25);
		c.cuenta++;
		j++;
		return j;
	}

	public static void main (String[] args){
		Contador micontador = new Contador();
		int num = actualiza (micontador, 100);

		System.out.println("Contador: "+micontador.cuenta);
		System.out.println("Número: "+num);
	}
}


public class Contador{
	public int cuenta = 5;

	public Contador (int j){
		cuenta = j;
	}

	public Contador (){

	}
}
```

¿Cuál es el resultado de ejecutar dicho código?

Respuesta:

Cuestión 2. (1 punto)

Localice los errores presentes en el código mostrado a continuación y proponga una solución para cada uno de ellos.

NOTA: La calificación obtenida en esta cuestión será proporcional al número de errores identificados por el alumno siempre que todos ellos sean correctos. En el supuesto de incluir alguna respuesta incorrecta, la cuestión entera quedará invalidada (i.e., 0 puntos).

```
package p2;

public class Padre {
	int atributo1 = 5;
	protected double atributo2 = 2.3;

	public Padre (int i, double d){
		atributo1 = i;
		atributo2 = this.d;
	}

	protected String metodo3 (){
		return “Método del Padre”;
	}

	public static void incrementa (){
		atributo1++;
	}
} //Fin de la clase Padre

// ************************************************************************
package p1;
import p2.Padre;

public class Hijo extends Padre{
	private int atributo3 = 1;

	public Hijo (){
		Padre (2, 4.5);
	}

	public Hijo (int i){
		atributo3 = i;
	}

	public void metodo1 (){
		atributo1++;
	}

	String metodo3 (){
		return “Método del Hijo “ + super.metodo3();
	}

	public String metodo3 (String cad){
		return “Mi nueva cadena: “+ super.metodo3() + cad;
	}
} //Fin de la clase Hijo
```

```
// ********************************************************************
package p2;
import p1.Hijo;

public class Prueba{

	public void metodo2 (){
		Padre p = new Padre (1, 2.2);
		p.atributo2++;
	}

	public abstract Padre metodo5 (Hijo h);

	public void metodo6 () {}

	public static void main (String[] args){
		Hijo.metodo1();
		new Padre (1, 2.3).metodo3();
	}

} //Fin de la clase Prueba
```

Respuesta (Escribir de nuevo el código de las tres clases corrigiendo los errores identificados y añadiendo en forma de comentarios la causa de cada error)

Cuestión 3. (1 punto)

Considere el código Java indicado a continuación:

```
public class Test{
	public static void main (String[] args){
		try{
			metodo ();
			System.out.println("Después de llamar a método");
		}catch(ArithmeticException e){
			System.out.println("Excepción aritmética");
		}
		catch(Exception e){
			System.out.println("Ha ocurrido una excepción");
		}
		finally {
			System.out.println("Fin de la ejecución");
		}

	}

	static void metodo () throws Exception{
		System.out.println ("Resultado de la operación: "+ 1/0);
	}
}
```

(0.5 puntos) Indique si las siguientes afirmaciones son verdaderas o falsas. **NOTA: Fallar alguna de las afirmaciones supone invalidar la cuestión entera.**

1. La excepción ArithmeticException es de tipo comprobada (*checked*). _____
2. Al ejecutar el código se muestran por pantalla los mensajes "Excepción aritmética", "Ha ocurrido una excepción" y "Fin de la ejecución". _____
3. El resultado de ejecutar este código es independiente del orden de los dos bloques catch. _____
4. La sentencia throws Exception es opcional en este código. _____
5. Si en el bloque try no se lanzase ninguna excepción, el mensaje "Fin de la ejecución" no se mostraría por pantalla. _____

(0.5 puntos) Incorpore los cambios necesarios en el código anterior de forma que el resultado de ejecución sea:

```
Después de llamar a método.
Fin de la ejecución.
```

NOTA: El código que aparece sombreado en gris NO se puede modificar (aunque sí es posible añadir nuevas sentencias tanto en main como en metodo(), en caso de ser necesario).

Respuesta:

Problema 1. (2.5 puntos)

Un alumno de GETT está realizando prácticas externas en un herbolario donde le han encomendado la tarea de desarrollar un programa de gestión para la tienda. Para ello, el estudiante ha definido las clases Java indicadas a continuación: **Plantas**, **Arboles**, **Arbustos**, **Matas** y **Hierbas**. **Supóngase la existencia y correcta implementación de la clase Cliente.**

```
public abstract class Plantas{
	private double altura;
	private String origen;
	public ArrayList<Cliente> compradores = new ArrayList<Cliente>();

}

public class Arboles extends Plantas{

}

public class Arbustos extends Plantas{

}

public class Matas extends Plantas{

}

public class Hierbas extends Plantas{

}
```

(0.5 puntos) El encargado del herbolario comunica al estudiante la necesidad de modelar en su programa información sobre la estación idónea para podar ciertos tipos de **Plantas** (de forma que estos datos se puedan aprovechar para enviar avisos automáticos a los clientes en función de las compras que hayan hecho en el herbolario). Implementar una interfaz llamada **Podable**, añadiendo y/o modificando el código Java indicado arriba para contemplar el escenario propuesto, considerando que sólo los **Árboles** y **Arbustos** se pueden podar (los primeros en "*invierno*" y los segundos en "*primavera*").

Respuesta:

(0.5 puntos) El programador decide incorporar tantos métodos toString() como considere necesarios para poder acceder al valor de los atributos **altura** y **origen** de cada tipo de **Planta** (i.e., **Árboles**, **Arbustos**, **Hierbas** y **Matas**). Añade el código Java que permita implementar dicha funcionalidad **con la menor cantidad de código posible**, indicando la/s clase/s en la/s que debe ser alojado cada método toString().

Respuesta:

(0.5 puntos) Añadir el código Java necesario para implementar el constructor de copia de la clase **Arbustos**.

Respuesta:

(1 punto) Con la intención de avisar automáticamente a cada cliente sobre la mejor estación para podar las **Plantas** adquiridas en el herbolario, nuestro desarrollador ha implementado el código Java indicado a continuación. ¿Es válido dicho código suponiendo que el método **void enviarCorreo (ArrayList<Cliente> clientes, String estacion)** existe y está correctamente implementado? Justificar la respuesta de forma razonada. En caso de error, proponer una implementación correcta para el método **AvisarClientes(…)**

```
public class Gestor {
        public static void AvisarClientes (ArrayList<Plantas> lista){
            for (int i=0; i<lista.size(); i++)
              enviarCorreo (lista.get(i).compradores, ((Podable)lista[i]).getEstacionPodado());
        }
…
}
```

Respuesta: