Examen final Programación Orientada a Objetos 1ª convocatoria

Grado en Ingeniería Informática – Curso 2021/22

(No están todas las preguntas del examen, además también había una parte de programación en Java pero "a papel")

- Dado el siguiente código, ¿generará algún tipo de error? ¿cuál es la salida por pantalla? Justifica la respuesta

```
Integer x = 100;
Integer y = x * (new Integer(8));
x=y;
System.out.println("Valor de x = "+ x);
x = 900;
System.out.println("Valor de y = "+ y);
```

- ¿Cuál o cuáles de las siguientes sentencias relativas a HashMap son correctas?

- El coste en tiempo de acceso a los datos de un *HashMap* no depende del número de datos almacenados
- Los *HashMap* son colecciones porque los objetos no se almacenan en el orden en el que se van introduciendo
- Los *HashMap* tienen asociado un *iterator* con el que se pueden recorrer todos los objetos almacenados en el *HashMap.*
- Cuando se invoca el método *get(cla1)* , se invoca al método *equals(cla1)* de las claves, de modo que, si el resultado es true para una de las claves, se devuelve el objeto que está asociado a dicha clave.
- Cuando se invoca el método *get(cla1)* , si el hash code de *cla1* coincide con el hash code de una de las claves del *HashMap* , se devuelve el objeto que está asociado a dicha clave
- En un *HashMap* , se devuelve el objeto que está asociado a dicha clave.
- En un *HashMap* , se pueden almacenar datos repetidos y nulos.

- ¿Cuál o cuáles de las siguientes sentencias relativas a HashMap son correctas?

- Los constructores se usan para reservar memoria para cualquier tipo de dato
- La inicialización de los atributos de una clase debería hacerse siempre en el constructor, salvo en el caso de las constantes (*final static*) que puede hacerse en el momento en el que se declaran.
- Si una clase tiene implementado un constructor, no se puede invocar el constructor por defecto.
- Los constructores pueden tener cualquier tipo y número de argumentos, pero siempre deben devolver *void.*
- En el siguiente trozo de código [*String x = new String("x1"); x = new String("x1")*] se realiza una reserva de memoria para la variable x.
- Los atributos de una clase que son objetos tienen *null* como su valor por defecto, salvo en el caso de wrappers a los tipos primitivos.

- Si *Periferico* es una clase declarada como *public abstract class Periferico implements Componente* que implementa el interface *Componente*, y *Pantalla* e *Impresora* son subclases de *Periferico*, dado el siguiente trozo de código:

```
Impresora impresora =  new Impresora();
Pantalla pantalla = new Pantalla();
Componente componente = (Componente) pantalla;
HashMap<String, Componente> componentes = new HashMap<>();
```

componentes.put(“A”, impresora);
componentes.put(“B”, componente);

¿Se generará algún tipo de error? Justifica la respuesta.

- ¿Cuál o cuáles de las siguientes afirmaciones sobre jerarquías de clases son ciertas?

- Desde un método de una clase derivada con super se puede acceder a los atributos y métodos privados (*private*) de la clase base.
- Desde un constructor sin argumentos de una clase derivada, mediante super se puede invocar a un constructor con argumentos de la clase base.
- La clase derivada hereda los atributos privados (*private*) de una clase base, pero no puede acceder a ellos.
- La clase derivada hereda todos los constructores de la clase base, pero no puede acceder a ellos.
- Para favorecer la encapsulación en herencia, los atributos de las clases base de una jerarquía deberían de ser protegidos (*protected*).
- Un constructor de la clase derivada siempre debería de invocar a un constructor de la clase base para inicializar los atributos privados de esa clase base (*private*).

- ¿Qué es el polimorfismo en herencia? ¿Qué utilidad tiene? Pon un ejemplo de esta utilidad. ¿Qué es el puntero *super*? ¿Qué visibilidad deberían tener los atributos de una clase?

- ¿Cuáles de las siguientes afirmaciones sobre almacenamiento de datos en Java son ciertas?

- Los objetos se almacenan en el montón, salvo cuando son instancias de wrappers a los tipos primitivos.
- Si en un método se declara un array de enteros (*int[]*), los valores del array se almacenarán en el montón y la referencia al array se almacenará en la pila .
- Dado el trozo de código [*for(int k=0; k<10000; k++) res+=k;*], si el tipo de dato de res es int se ejecutará más rápido que si res es una instancia de *Integer.*
- Cuando se crea un objeto de una clase, el código de los métodos y de los constructores de dicha clase se almacena en la pila.
- Las clases se almacenan en la pila.
- Todas las referencias a los objetos de un programa se almacenan en la pila.
- Los atributos de tipo primitivo se almacenan en la pila.

- ¿Qué características tiene un método abstracto *(abstract)*?

- Si tenemos dos excepciones *ExceptA* y *ExceptB*, de modo que *ExceptB* es una subclase de *ExceptA*, *metA*() puede lanzar una excepción *ExceptA* y *metB*() puede lanzar una excepción *ExceptB*, entonces dado el siguiente trozo de código:

```
try {
metB();
metA();
} catch (ExceptA except) { System.out.println(“Excepción A lanzada”);
} catch (ExceptB except) { System.out.println(“Excepción B lanzada”);
} finally {System.out.println(“Acabar”);}
```

¿Cuál es la salida de pantalla cuando ocurre una excepción metA()? ¿Y cuando ocurre en metB()? ¿Y si no se produce ninguna excepción? Justifica todas las respuestas.