

### **PARTE TEÓRICA - TEST [2,5 PUNTOS]:**

Sólo una de las respuestas es válida. Las respuestas correctas se puntuarán con +1.0, mientras que las respondidas de manera incorrecta se puntuarán con -0.25. Las no contestadas no tendrán influencia ni positiva ni negativa en la nota.

**Pregunta 1:** ¿Cuál es el resultado de ejecutar el siguiente fragmento de código?

```
if ("String".toString() == "String")
    System.out.println("Igual");
else
    System.out.println("No Igual");
```

- a. El código compilará e imprime "Igual".
- b. El código compilará e imprime "No Igual".
- c. El código compilará pero producirá un error de ejecución.
- d. El código no compilará.

**Pregunta 2:** ¿Cuál es el resultado de ejecutar el siguiente código?

```
public class Ejemplo {
    private int i=j;
    private int j=10;
    public static void main (String []args) {
        System.out.println ((new Ejemplo()).i);
    }
}
```

- a. Da un error de compilación debido a las restricciones de acceso a las variables privadas de Ejemplo.
- b. Da un error de compilación debido a la referencia que se hace a variables declaradas con posterioridad.
- c. No da ningún error de compilación y produce como salida el valor 0.
- d. No da ningún error de compilación y produce como salida el valor 10.

**Pregunta 3:** ¿Cuál de las siguientes es una característica de la clase java.lang.Exception?

- a. private.
- b. extends Throwable.
- c. implements Throwable.
- d. final.

**Pregunta 4:** Sean "Mamifero" y "Gato" dos clases que mantienen una relación de herencia padre-hijo. ¿Qué habría que modificar en el siguiente código para que sea correcto y por qué?

```
Animal a;           /* Línea 1 */
Gato b;             /* Línea 2 */
a = new Animal (); /* Línea 3 */
b = a;              /* Línea 4 */
```

- a. Nada. Es correcto.
- b. No se puede asignar un objeto a otro de otra clase, luego cambiamos la línea 2: Animal b; .
- c. Es necesario explicitar el tipo cuando asignamos un objeto a otro objeto perteneciente a una clase hija, luego cambiamos la línea 4: b = (Gato) a; .
- d. Es necesario explicitar el tipo y crear una nueva instancia cuando asignamos un objeto a otro objeto perteneciente a una clase hija, luego cambiamos la línea 4: b = new (Gato) a; .

**Pregunta 5:** Una variable de clase, definida como static ...

- a. No puede ser accedida desde otra clase.
- b. Si se modifica, lo hace para todas las instancias de la clase.
- c. Es de valor constante.
- d. Solo puede ser accedida desde clases del mismo paquete.

**Pregunta 6:** En una estructura `switch`, ¿en qué lugar tiene que ser colocado el bloque de sentencias “default”?

- a. Antes de las diferentes sentencias `case`.
- b. Después de todas las sentencias `case`.
- c. Después de las sentencias `case` pero antes de la sentencia `finally`.
- d. Puede colocarse en el lugar que se quiera.

**Pregunta 7:** Dada la siguiente instrucción:

```
x = y--;
```

¿Cuál de las siguientes afirmaciones es verdadera DESPUÉS de ejecutarse la instrucción?

- a. La instrucción da un error de compilación.
- b.  $x > y$ .
- c.  $x == y$ .
- d.  $x < y$ .

**Pregunta 8:** ¿Qué ocurre si se compila y ejecuta el siguiente código?

Número de Línea	Código
-----------------	--------

4	<code>public class Clase {</code>
5	<code>    public static void main (String []arguments) {</code>
6	<code>        met (arguments); }</code>
7	<code>    public void met (String []arguments) {</code>
8	<code>        System.out.println(arguments);</code>
9	<code>        System.out.println(arguments[1]); }</code>

- a. Da un error de compilación porque no se puede hacer referencia al método no-estático `met`.
- b. Da un error de compilación porque el método `main` no puede ser estático.
- c. Da un error de compilación porque el array `arguments` no puede pasarse como parámetro al método `met`.
- d. Da un error de ejecución porque en el acceso al array `arguments` nos salimos del rango de dicho array.

**Pregunta 9:** ¿Cuál de los siguientes no es un identificador válido en Java?

- a. `#variable`.
- b. `$variable`.
- c. `_variable`.
- d. `vari_able`.

**Pregunta 10:** En la definición de una interface en Java :

- a. Es necesaria emplear la palabra clave `abstract`.
- b. La signatura de los métodos de una interfaz tienen visibilidad `public` o `private`, pero no `protected`.
- c. No se permiten campos constantes.
- d. Aunque no se indique usando la palabra clave `final`, todos los campos son tratados como si así fuesen.

**Pregunta 11:** Una clase interna:

- a. Puede acceder a los campos y métodos públicos y protegidos de la clase envolvente, pero no privados.
- b. Puede acceder a los campos y métodos públicos de la clase envolvente, pero no a privados ni a protegidos.
- c. Puede acceder a los campos y métodos públicos y privados de la clase envolvente.
- d. No puede acceder a los campos y métodos privados de la clase envolvente.

**Pregunta 12:** Según el texto de la bibliografía básica de la asignatura, indique cuál de las siguientes afirmaciones es correcta:

- a. La depuración es la actividad cuyo objetivo es determinar si una pieza de código produce el comportamiento pretendido.
- b. La prueba viene a continuación de la depuración.
- c. La depuración es una actividad dedicada a determinar si un segmento de código contiene errores.
- d. La depuración es el intento de apuntar con precisión y corregir un error en el código.

**Pregunta 13:** ¿De qué clase deriva la clase ArrayList?

- a. ArrayList.
- b. AbstractCollection.
- c. ArrayCollection.
- d. ListCollection.

**Pregunta 14:** Cuando queremos que un objeto oiga eventos de acción disparados por el usuario, el objeto tiene que implementar la interfaz ...

- a. ActionEvent.
- b. ActionListener.
- c. ListenerAction.
- d. ListenerEvent.

**Pregunta 15:** Un conjunto es una:

- a. Que almacena cada elemento individual una sola vez como mínimo. No mantiene un orden específico.
- b. Que almacena cada elemento individual una sola vez como mínimo. Mantiene un orden específico.
- c. Que almacena cada elemento individual una sola vez como máximo. No mantiene un orden específico.
- d. Que almacena cada elemento individual una sola vez como máximo. Mantiene un orden específico.

### **PARTE PRÁCTICA [6,5 PUNTOS]:**

Considere para su estudio una versión del conocido juego Space Invaders. Este juego consiste en que varias filas de naves alienígenas o UFOs avanzan hacia la base defensora, con movimientos oscilatorios de izquierda a derecha, bajando poco a poco. Así, una nave guardián defiende la base y trata de evitar los misiles lanzados esporádicamente por las naves invasoras. La nave guardián lanza disparos de uno en uno. El juego finaliza cuando todos los invasores han sido alcanzados o cuando los invasores llegan a la base.

- a) **[2,5 puntos]** Suponga que la implementación del juego se hace en base a la existencia de una clase `Nave`, de tipo abstracto, que sirve de clase de referencia para otras posibles subclases (tanto las naves alienígenas como la nave guardiana). A partir de ésta, se genera una nueva clase denominada `NaveUFO` que sirve para modelar UNA nave UFO. El juego dispondrá de un total cuatro filas de siete naves UFO cada una de ellas, que se encuentran en la parte superior de la pantalla del juego. Proporcione la estructura de ambas clases (y del bloque completo de naves UFO), así como los diferentes atributos que considere imprescindibles para cada una de ellas y los principales métodos accesores y modificadores. Si se necesita del uso de alguna otra clase auxiliar, debe definirse también en este apartado.
- b) **[2,5 puntos]** Proporcione el método `desplazarNavesUFO` que simula el movimiento de las naves UFO a derecha e izquierda. El movimiento se realizará cuando el salta un determinado timer (que no hay que implementar, sólo el método que se llama cuando este timer salta). Las naves parten de la zona superior izquierda y se desplazan hasta la parte derecha de la pantalla. Cuando llegan al final, bajan todas las naves una posición y comienzan a desplazarse ahora hacia la izquierda. Se deja a su elección el prototipo que tienen que tener estos métodos, pero han de ser coherentes con lo expuesto en el apartado anterior. Si se necesita del uso de alguna otra clase auxiliar, debe definirse también en este apartado.
- c) **[1,5 puntos]** Proporcione el método `disparaMisil`, que simula el disparo de un proyectil ascendente por parte de una nave UFO. Se recuerda la restricción de que en un momento determinado sólo puede haber activo un único misil. Se deja a su elección el prototipo que tienen que tener estos métodos, pero han de ser coherentes con lo expuesto en el primer apartado. Si se necesita del uso de alguna otra clase auxiliar, debe definirse también en este apartado.