

**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:** Dada la declaración de las siguientes variables, indicar cuáles de ellas son correctas.

1. `float foo = -1;` **no da error**
2. `float foo1 = 1.0;` **le falta la F, para ser float**
3. `float foo2 = 42e1;` **le falta la f**
4. `float foo3 = 2.02f;` **float**
5. `float foo4 = 3.03d;` **double**
6. `float foo5 = 0x0123;` **literal hexadecimal**

- a. 1 y 2  
b. 1 y 3  
**c. 4 y 6 (Respuesta correcta)**  
d. 3 y 4

**Pregunta 2:** Dado el siguiente fragmento de código, indica cuál de las siguientes afirmaciones es correcta en relación al valor de la variable `foo`.

**Número de Línea**    **Código**

- ```
4      int index = 1;
5      boolean[] test = new boolean[3];
6      boolean foo = test [index];
```

- a. `foo` tiene el valor 0  
b. `foo` tiene el valor `null`  
**c. `foo` tiene el valor `false` (Respuesta correcta)** `boolean` se inicializan a `false`  
d. Se produce una excepción y `foo` no posee ningún valor

**Pregunta 3:** Dadas las siguientes expresiones, indica cuál de las opciones es la correcta.

1. `(1 > 1) && (1 > 1) == (1 > 1) == false`
2. `(1 == 1) | (10 > 1) == true | true == true`

- a. La expresión 1 es evaluada como falsa y la expresión 2 como falsa.  
**b. La expresión 1 es evaluada como falsa y la expresión 2 como verdadera. (Respuesta correcta)**  
c. La expresión 1 es evaluada como verdadera y la expresión 2 como falsa.  
d. La expresión 1 es evaluada como verdadera y la expresión 2 como verdadera.

**Pregunta 4:** Dado el siguiente código, ¿cuál es su resultado?

**Número de Línea**    **Código**

- ```
4      class Top {
5          public Top(String s) { System.out.print("B"); }
6      }
7      public class Bottom2 extends Top {
8          public Bottom2(String s) { System.out.print("D"); }
9          public static void main(String [] args) {
10             Bottom2 obj=new Bottom2("C");
11             System.out.println(" ");
12         }
13     }
```

- a. BD  
b. DB  
c. BDC

**d. Error de compilación (Respuesta correcta)** no invoca al constructor de la superclase super(s)

**Pregunta 5:** Dado el siguiente código, ¿cuál de las afirmaciones es cierta?

Número de Línea	Código
4	class Hotel {
5	public int reservas;
6	public void reservar() {
7	reservas++;
8	}
9	}
10	public class SuperHotel extends Hotel {
11	public void reservar() {
12	reservas--;
13	}
14	public void reservar(int size) {
15	reservar();
16	super.reservar();
17	reservas += size;
18	}
19	public static void main(String[] args) {
20	SuperHotel hotel = new SuperHotel();
21	hotel.reservar(2);
22	System.out.print(hotel.reservas);
23	}
24	}

- a. Error de compilación.
- b. Lanza una excepción en tiempo de ejecución.
- c. 0.

**d. 2. (Respuesta correcta)**

**Pregunta 6:** 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. (Respuesta correcta)** Pag. 170

**Pregunta 7:** Según el texto de la bibliografía básica de la asignatura, indique cuál de las siguientes afirmaciones es correcta:

**a. Un encapsulamiento apropiado en las clases reduce el acoplamiento. (Respuesta correcta)**

- b. El término acoplamiento describe cuánto se ajusta una unidad de código a una tarea lógica o a una entidad.
- c. El acoplamiento describe la conectividad de los propios objetos de una clase.
- d. Un sistema débilmente acoplado se caracteriza por la imposibilidad de modificar una de sus clases sin tener que realizar cambios en ninguna otra.

**Pregunta 8:** Según el texto de la bibliografía básica de la asignatura, indique cuál de las siguientes afirmaciones es **FALSA** en relación a los métodos polimórficos:

- a. Una variable polimórfica es aquella que puede almacenar objetos de diversos tipos.

**b. Las llamadas a métodos en Java no son polimórficas. (Respuesta correcta)**

- c. El mismo método puede invocar en diferentes momentos diferentes métodos dependiendo del tipo dinámico de la variable usada para hacer la invocación.
- d. Cada objeto en Java tiene un método `toString` que puede usarse para devolver un `String` de su representación.

**Pregunta 9:** Según el texto de la bibliografía básica de la asignatura, indique cuál de las siguientes opciones declarará un método en una clase que fuerza a una subclase a implementarlo:

- a. `static void methoda (double d1) {}`
- b. `public native double methoda();`
- c. `abstract public void methoda () ;` (Respuesta correcta)**
- d. `protected void methoda (double d1){}`

**Pregunta 10:** Dado el siguiente fragmento de código que pretende mostrar un ejemplo de sobrescritura, indique cuál de las siguientes opciones completaría el código para dar lugar a un ejemplo correcto de sobrescritura:

Número de Línea	Código
4	<code>class BaseClass {</code>
5	<code>    private float x = 1.0f ;</code>
6	<code>    protected float getVar () {return x;}</code>
7	<code>}</code>
8	<code>class Subclass extends BaseClass {</code>
9	<code>    private float x = 2.0f;</code>
10	<code>    //Insertar código aquí</code>
11	<code>}</code>

- a. `float getVar ( ) { return x;}`
- b. `public float getVar ( ) { return x;}` (Respuesta correcta)**
- c. `float double getVar ( ) { return x;}`
- d. `public float getVar (float f ) { return f;}`

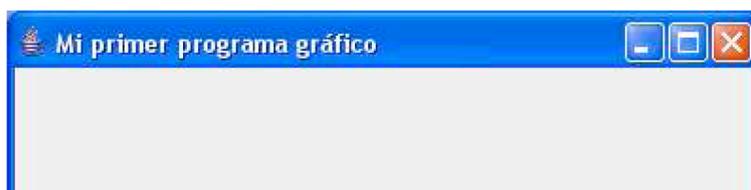
**Pregunta 11:** Según el texto de la bibliografía básica de la asignatura, indique cuál de las siguientes afirmaciones es correcta en relación a la programación por parejas:

- a. Consiste en programar una clase por duplicado con el objetivo de depurar los errores más fácilmente.
- b. Es una manera de producir código, opuesta a la programación extrema en la que un solo programador desarrolla las clases asignadas.
- c. Era una técnica de programación tradicional que las empresas eliminaron para reducir costes.
- d. Es uno de los elementos de una técnica que se conoce como programación extrema. (Respuesta correcta)**

**Pregunta 12:** La ejecución del siguiente fragmento de código ...

Número de Línea	Código
4	<code>import javax.swing.*;</code>
5	<code>class PrimerFrame extends JFrame</code>
6	<code>{</code>
7	<code>    public PrimerFrame()</code>
8	<code>    {</code>
9	<code>        setTitle("Mi primer programa gráfico");</code>
10	<code>        setSize(400,100);</code>
11	<code>    }</code>
12	<code>}</code>
13	<code>public class FrameTest</code>
14	<code>{</code>
15	<code>    public static void main (String[] args)</code>
16	<code>    {</code>
17	<code>        JFrame frame = new PrimerFrame();</code>
18	<code>        frame.setVisible (true);</code>
19	<code>    }</code>
20	<code>}</code>

Da lugar al siguiente programa:



Pero este último programa tiene el problema de que cuando se cierra la ventana, a pesar de que dejamos de verla, el programa no finaliza su ejecución. De esta forma, para que el programa funcione correctamente, hemos de interceptar el evento que se produce cuando cerramos la ventana y hacer que el programa termine su ejecución en ese momento. Indique qué clase hemos de definir en este caso y asociárselo al `JFrame` del ejemplo:

- a. `ActionListener`
- b. `ComponentListener`
- c. `WindowListener` (Respuesta correcta)**
- d. `ItemListener`

**Pregunta 13:** En el siguiente fragmento de código hemos definido la ejecución de cinco bloques. Estos bloques se ejecutarán dependiendo de las excepciones que se produzcan en cada caso. Indique cuál de las siguientes afirmaciones es correcta:

Número de Línea	Código
4	<code>// Bloque1</code>
5	<code>try{</code>
6	<code>    // Bloque2</code>
7	<code>}catch (ArithmeticException e) {</code>
8	<code>    // Bloque3</code>
9	<code>}finally{</code>
10	<code>    // Bloque4</code>
11	<code>}</code>
12	<code>// Bloque5</code>

- a. El Bloque4 no se ejecutará si se produce una excepción de tipo aritmético en el Bloque2
- b. El Bloque4 no se ejecutará si se produce un acceso a un objeto nulo (null) en el Bloque2
- c. El Bloque4 se ejecutará antes que el Bloque3 si se produce una excepción de tipo aritmético en el Bloque2
- d. El Bloque4 se ejecutará antes de que la excepción producida por un acceso a un objeto nulo (null) en el Bloque2 se propague hacia arriba (Respuesta correcta)**

**Pregunta 14:** Indique el resultado de ejecutar el siguiente código que se muestra a continuación:

Número de Línea	Código
4	<code>public class test {</code>
5	<code>    public static void add3 (Integer i) {</code>
6	<code>        int val = i.intValue();</code>
7	<code>        val += 3;</code>
8	<code>        i = new Integer (val);</code>
9	<code>    }</code>
10	<code>    public static void main (String args[]) {</code>
11	<code>        Integer i = new Integer (0);</code>
12	<code>        add3 (i);</code>
13	<code>        System.out.println (i.intValue ( ) );</code>
14	<code>    }</code>
15	<code>}</code>

- a. El programa indicará un fallo en tiempo de compilación.
- b. El programa imprime por pantalla el valor "0". (Respuesta correcta)**
- c. El programa imprime por pantalla el valor "3".
- d. El programa lanzará una excepción en la línea 6 (`int val = i.intValue();`).

**Pregunta 15:** Dado el siguiente código ...

<b>Número de Línea</b>	<b>Código</b>
4	public class testJunio {
5	public void setVar (int a, int b, float c) {
6	}
7	// INSERTAR CÓDIGO AQUÍ
8	}

Y los siguientes métodos:

1	private void setVar (int a, float c, int b) { }	<b>diferentes tipos de los parámetros</b>
2	protected void setVar (int a, int b, float c) { }	
3	public int setVar (float a, int b, int c) {return b;}	
4	public int setVar (int a, int b, float c) {return a;}	
5	protected float setVar (int a, int b, float c) {return c;}	

Indique qué métodos permiten una sobrecarga del método setVar de manera correcta:

a. 1 y 2

**b. 1 y 3 (Respuesta correcta)**

c. 3 y 5

d. 3 y 4

## PARTE PRÁCTICA [6,5 PUNTOS]:

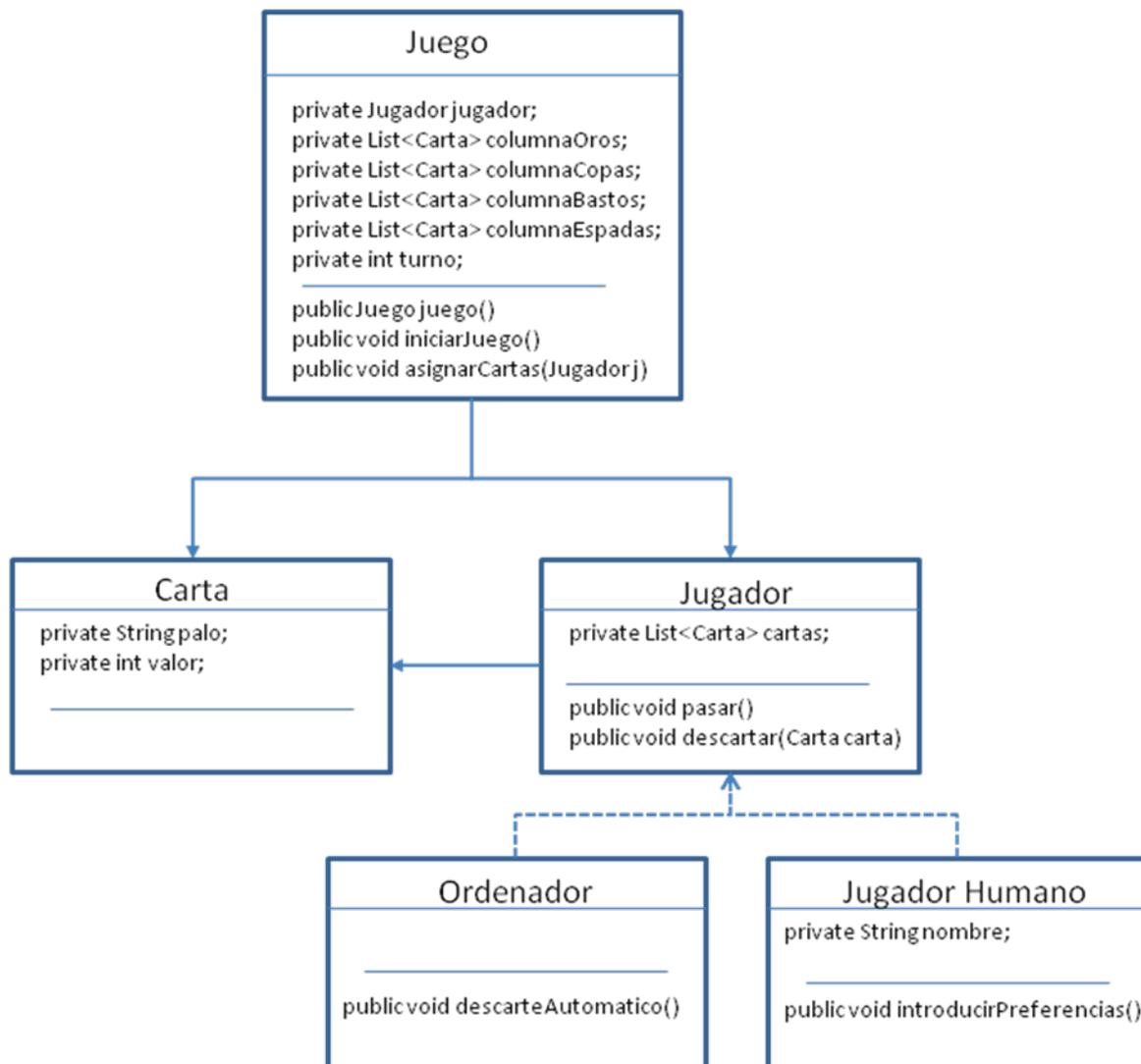
El juego del Cinquillo Solitario es una variedad del popular Cinquillo en el cual un jugador puede jugar de manera online contra el ordenador. El juego se inicia con el reparto de todas las cartas de una baraja española que consta de 48 naipes o cartas, clasificados en cuatro palos (oros, bastos, copas y espadas) y numerados del 1 al 12. El objetivo del juego consiste en descartarse (quedarse sin cartas) antes que el oponente.

El jugador que posee el cinco de oros lo coloca boca arriba encima de la mesa y de esta forma empieza el turno de descartes. En turnos alternativos, cada jugador puede descartarse de máximo un naipe. Solo se pueden colocar cincos o todas aquellas cartas que siguen en progresión ascendente o descendente a las que hay en la mesa y sean del mismo palo. Es decir, si por ejemplo solamente está colocado el cinco de oros en la mesa, los jugadores solo podrán colocar el seis o el cuatro de oros o un cinco de otro palo.

Si un jugador no puede colocar ninguna carta pasa, y le toca el turno al siguiente jugador. Nunca se puede pasar si se puede colocar alguna carta. El primer jugador que consigue colocar todas sus cartas sobre la mesa es el ganador.

En cuanto a la dinámica del juego, uno de los contrincantes será un jugador humano (introducimos sus datos y sus preferencias por el teclado) y el otro contrincante será el propio ordenador.

- a) [1,5 puntos] Diseñe las clases necesarias que permita desarrollar el juego del Cinquillo Online utilizando un paradigma orientado a objetos. Debe hacerse uso de los mecanismos de la programación orientada a objetos siempre que sea posible y un diseño que permita la reutilización del código y facilite su mantenimiento.



- b) **[1,5 puntos]** Implemente un método que defina el funcionamiento del ordenador, teniendo en cuenta que todos sus procesos tienen que hacerse automáticamente sin la intervención del usuario.

```
public void descartar(){

    ArrayList<ArrayList<Carta>> columnas = new ArrayList<ArrayList<Carta>>();
    columnas = juego.getCartas();
    Carta cartaMenor = null;
    Carta cartaMayor = null;
    boolean descarte = false;

    if(esPosibleDescarte(columnas)){

        for (ArrayList<Carta> columna : columnas) {

            if(columna.size(>0){

                String palo = columna.get(0).getPalo();
                cartaMayor = columna.get(0);
                cartaMayor = columna.get(columna.size());

                for (Carta carta : super.getCartas()) {

                    if(carta.getPalo().equals(palo)){
                        if(carta.getValor()+1==cartaMenor.getValor()){
                            descartar(carta);
                            descarte = true;
                            break;
                        }else if(carta.getValor()-1==cartaMayor.getValor()){
                            descartar(carta);
                            descarte = true;
                            break;
                        }
                    }
                }
            }
            if(descarte){
                break;
            }
        }
    }

    if(!descarte){
        for (Carta carta : super.getCartas()) {
            if(carta.getValor()==5){
                descartar(carta);
            }
        }
    }

    }else{
        super.pasar();
    }
}
```

- c) **[1,5 puntos]** Proporcione un método que muestre la lógica del juego, definiendo la información necesaria para establecer el uso de clases, interacciones entre elementos, declaración y uso de variables y métodos necesarios, etc.

```
public void iniciarJuego() {  
  
    boolean cartasPendientes = true;  
  
    Jugador jugadorHumano = new Jugador();  
    Jugador jugadorOrdenador = new Jugador();  
  
    asignarCartas(jugadorHumano);  
    asignarCartas(jugadorOrdenador);  
  
    while(cartasPendientes) {  
  
        if(jugadorHumano.iniciarTurno()) {  
            System.out.println("Jugador Humano Ganador");  
            break;  
        } else if(jugadorOrdenador.iniciarTurno()) {  
            System.out.println("Ordenador Ganador");  
            break;  
        }  
    }  
}
```

- d) **[2,0 puntos]** Indiqué qué modificaciones son necesarias introducir en la aplicación para permitir la participación de varios jugadores humanos (hasta 4). Para ello el juego en lugar de constar de partidas individuales en las cuales gana el jugador que antes se descarta, para a ser una partida formada por un conjunto de rondas. El ordenador deberá llevar un registro de los puntos que cada jugador ha conseguido en cada ronda. El jugador que consigue descartarse primero logrará 3 puntos, el jugador o jugadores que se quede con un mayor número de cartas al finalizar la ronda obtendrá 0 puntos. El resto obtendrá 1 punto. La partida finaliza cuando un jugador consiga llegar al menos a los 10 puntos, ganando el que más puntos tenga en caso de superar esta puntuación varios jugadores. En caso de empate se jugará una ronda extra para decidir el ganador.

- Para la participación de varios jugadores, tan solo es necesario tener una instancia de cada jugador y asignar los turnos en un orden lógico. En lugar de tener una variable con la instancia del jugador, sería necesaria una lista de jugadores.
- Para que el juego pueda anotar el tanteo de las partidas, se debería crear una clase "Tanteo" que almacene por cada una de las rondas de la partida, los puntos conseguidos por cada jugador. También sería necesario que almacenara los puntos que cada jugador ha conseguido de manera global. Además, para implementar la asignación de puntos, habría que crear un método que al final de cada partida analice las cartas que los jugadores no han conseguido descartar. Este método actualizaría la información de los puntos de cada jugador y debe dar por finalizada la partida en caso de que un jugador llegase a la puntuación máxima. La ronda extra también sería responsabilidad de este método y de la información almacenada en la clase Tanteo.