

Apellidos: _____ Nombre: _____ DNI: _____

Centro Asociado en el que está MATRICULADO: _____

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja, y **ENTRÉGUELA OBLIGATORIAMENTE** con el resto de hojas de su examen. **Cualquier examen que no venga acompañado de esta hoja de enunciados no será corregido.** Complete **TODOS** los datos que se piden en la hoja de lectura óptica o **en caso contrario su examen no será corregido.** El examen consta de 20 preguntas. Cada respuesta acertada tiene un valor de 0.5 puntos y cada respuesta incorrecta de -0.25 puntos. Para superar el examen es necesario obtener una puntuación de al menos 5 puntos.

Test : Conteste exclusivamente en una HOJA DE LECTURA ÓPTICA, no olvidando marcar que su tipo de examen es E.

1. Indique cuál es el resultado de la ejecución del método *main* de la clase Ejemplo5:

```
public class Ejemplo5_Super {  
    static int a;  
    Ejemplo5_Super () {a=1; System.out.print(a+" ");}  
}  
  
public class Ejemplo5 extends Ejemplo5_Super {  
    static int a=10;  
    static void incrementa() { a++; System.out.print(a+" "); }  
    static { a+=3; System.out.print(a+" ");}  
    public static void main(String[] args) { incrementa(); }  
}
```

- A) 1 11 14 B) 11 14 C) 13 14 D) Ninguno de los anteriores.

2. Indique cuál es el resultado de la ejecución del método *main* de la clase Ejemplo6:

```
class Ejemplo6_A {  
    int i=1;  
    void escribe_i() { System.out.print(" "+i);}  
    void escribe_2i() { System.out.print(" "+2*i);}  
}  
  
class Ejemplo6_B extends Ejemplo6_A {  
    final int i=2;  
    void escribe_2i() { System.out.print(" "+2*i);}  
}  
  
public class Ejemplo6 {  
    public static void main(String[] args) {  
        Ejemplo6_B e_B = new Ejemplo6_B();  
        Ejemplo6_A e_AB = new Ejemplo6_B();  
        e_B.escribe_i(); e_B.escribe_2i();  
        e_AB.escribe_i(); e_AB.escribe_2i();  
    }  
}
```

- A) 1 4 1 4 B) 2 4 1 4 C) 2 4 1 2 D) Ninguno de los anteriores.

3. Indique cuál es el resultado de la ejecución del método *main* de la clase Ejemplo7:

```
public class Ejemplo7_A {
    public Ejemplo7_A() { System.out.print(" A"); }
    public Ejemplo7_A(int i) { System.out.print(" A"+i); }
}

public class Ejemplo7_B extends Ejemplo7_A {
    public Ejemplo7_B(int i) { System.out.print(" B"+i); }
}

public class Ejemplo7 {
    public static void main(String[] args) {
        Ejemplo7_B e = new Ejemplo7_B(3);
    }
}
```

- A) A3 B3 B) A B3 C) B3 A3 D) Ninguno de los anteriores.

4. Señale cuál de las siguientes afirmaciones, relativas a las clases mostradas a continuación, es correcta. Las líneas se han numerado con el propósito de facilitar la referencia a las mismas:

```
1  public abstract class Ejemplo8_A {
2      abstract void escribe_Uno();
3      void escribe_Dos() { System.out.println("Este es un método concreto"); }
4  }
5
6  public class Ejemplo8_B extends Ejemplo8_A {
7      void escribe_Uno() { System.out.println("Implementación del método abstracto"); }
8      void escribe_Tres() { System.out.println("Otro método concreto"); }
9  }
10
11 public class Ejemplo8 {
12     public static void main(String[] args) {
13         Ejemplo8_A e_2 = new Ejemplo8_B();
14         e_2.escribe_Uno();
15         e_2.escribe_Tres();
16     }
17 }
```

- A) Error en la línea 3: una clase abstracta no puede implementar métodos “concretos”, es decir, “no abstractos”.
B) Error en la línea 13: no se puede crear una referencia a una clase abstracta de forma que se utilice como referencia a un objeto de una subclase (Aclaración: “subclase” es lo mismo que “clase derivada”).
C) Error en la línea 15: el método *escribe_Tres()* es desconocido.
D) Las clases no contienen ningún error.

5. Indique cuál es el resultado de la ejecución del método *main* de la clase Ejemplo9:

```
public class Ejemplo9 {
    public static void main(String[] args) {
        int d, a;
        try {
            d=0; a=42/d; System.out.print(" A");
        } catch (ArithmeticException e1) {
            System.out.print(" B"); d=1;
        } catch (Exception e2) {
            System.out.print(" C");
        }
        System.out.println(" D");
    }
}
```

- A) A B C B) B D C) B C A D D) Ninguno de los anteriores.

6. El método `init()` de un applet se ejecuta:

- A) Sólo una única vez y siempre después de cargar el applet en el navegador e invocar al método `main()`.
- B) Siempre que se invoque al método `start()` por las razones que sean: petición del usuario, cambio de contexto, maximización de la ventana del navegador, etc.
- C) Una única vez durante toda la vida del applet, es decir, desde que se carga en el navegador hasta que se descarga.
- D) Ninguna de las anteriores.

7. El método `start()` :

- A) Es de obligada implementación en todos los applets, por lo que siempre hay que incluirlo.
- B) Es de uso opcional en los applets, es decir, no es necesario incluirlo.
- C) Es de uso opcional en los applets pero, si se utiliza, siempre debe ir acompañado de su complementario, el método `stop()`.
- D) Ninguna de las anteriores.

8. En líneas generales, para poder ejecutar un applet como si se tratara de una aplicación:

- A) Se añade el método `main()` al código fuente applet, se instancia el applet, se crea un frame, se incluye el applet en el frame mediante el método `add()` y se realizan las correspondientes llamadas a los métodos `init()` y `start()`.
- B) Se añade el método `main()` al código fuente applet, se instancia el applet y se realizan las correspondientes llamadas a los métodos `init()` y `start()`.
- C) Se ejecuta desde la línea de comandos utilizando el comando `java` como si se tratase de una aplicación. Previamente habrá sido necesario compilar el applet como aplicación utilizando la opción `javac -s` (`s=standalone`).
- D) Ninguna de las anteriores.

9. Si en tiempo de ejecución se quieren conocer las propiedades y métodos de un Java Bean es necesario:

- A) Utilizar el método estático `getBeanInfo()` de la clase `Introspector` y los métodos `getPropertyDescriptors()` y `getMethodDescriptors()` de la interfaz `BeanInfo` para obtener arrays con las propiedades editables y métodos del Java Bean.
- B) Utilizar el método estático `getBeanInfo()` de la clase `Beans` y los métodos `getPropertyDescriptors()` y `getMethodDescriptors()` de la interfaz `BeanInfo` para obtener arrays con las propiedades editables y métodos del Java Bean.
- C) Utilizar el método estático `getBeanName()` de las clases `Property` y `Method` para obtener el nombre del Bean y a continuación los métodos `getPropertyDescriptors()` y `getMethodDescriptors()` para obtener arrays con las propiedades y los métodos.
- D) Ninguna de las anteriores.

10. En Java, la disminución de la prioridad de un thread:

- A) Se realiza mediante el método `setMinimumPriority()` de la clase `Thread` y ocasiona que el thread se ejecute con menor frecuencia.
- B) Se efectúa mediante el método `setPriority(int newPriority)` y puede ocasionar que el thread nunca se ejecute si los restantes threads se mantuviesen con una prioridad superior.
- C) Se realiza mediante el método `setPriority(int newPriority)` y ocasiona que el thread se ejecute con menor frecuencia que los threads con una prioridad superior.
- D) Ninguna de las anteriores.

11. ¿ Qué realizaría el siguiente código en Java?

```
interface coche{void velocidad();}  
interface coche_carreras extends coche{void maxima_velocidad();}  
class ferrari implements coche_carreras{  
public void velocidad(){}  
public void maxima_velocidad(){}  
}
```

- A) Se define una interfaz coche. Se define una interfaz por herencia coche_carreras y se implementa en ferrari.
- B) Se define una interfaz coche y se implementa en ferrari.
- C) No es un código correcto ya que una interfaz no se puede definir haciendo uso de la herencia.
- D) Ninguna de las anteriores.

12. ¿ Es correcta es sentencia en JAVA?

```
double[] quinielas= new double[15];
```

- A) Es correcto,se está declarando y creando un array llamado quiniela de 15 elementos.
- B) Es incorrecto.Los arrays JAVA no se declaran y crean a la vez.
- C) Es correcto pero sólo está definiendo un vector quinielas. Posteriormente habra que crear el objeto
- D) Ninguna de las anteriores

13. Cuando se ejecute el siguiente código cuál será el contenido final de la variable cliente?

```
class prueba  
{ public static void main (string[] args)  
  costumer cliente;  
  cliente= new customer();  
  cliente= new custumer();  
}
```

- A) Se crean dos objetos cliente de la clase customer y son los dos accesibles usando su el único identificador cliente definido.
- B) Se crean dos objetos cliente de la clase customer sólo se puede acceder al último creado. El primero es inaccesible.
- C) Se crean dos objetos cliente de la clase customer. El último creado no se puede acceder usando el identificador cliente definido
- D) Ninguna de las anteriores

14. Un (thread) hilo tiene

- A) 4 estados: Nuevo, Ejecutable, “muerto” y bloqueado. Es incorrecto.Los arrays JAVA no se declaran y crean a la vez.
- B) Los hilos no tiene ningún estado asociado.
- C) Un hilo tien tres estado: Nuevo, Ejecutable y muerto ya que nunca se puede bloquear.
- D) Ninguna de las anteriores

15. ¿ Que tienen en común los métodos sleep(), suspend() y wait() respecto a los hilos (threads)

- A) No tienen nada en común.
- B) Ponen a un hilo en el estado bloqueado.
- C) Se utilizan para inicializar un hilo.
- D) Ninguna de las anteriores

16. ¿Cuál de las siguientes afirmaciones es cierta?

- A) Es posible crear objetos de clases abstractas.
- B) Es posible crear objetos de interfaces.
- C) Es posible crear manejadores de interfaces.
- D) Ninguna de ellas.

17. ¿Cuál de las siguientes afirmaciones NO es cierta?

- A) *BufferedInputStream* es un flujo de bytes.
- B) *FileReader* es un flujo de bytes.
- C) *PrintWriter* es un flujo de caracteres.
- D) *BufferedReader* es un flujo de caracteres.

18. En Java, se dice que un *thread* se encuentra en estado *muerto* si:

- A) Ha finalizado su método *run()*.
- B) Si ha ejecutado un *sleep()*.
- C) Si ha recibido el mensaje *suspend()*.
- D) Si ha ejecutado un *wait()*.

19. ¿Cuál de las siguientes afirmaciones es cierta?

- A) Un *ServerSocket* se usa para que el cliente devuelva un *Socket* al servidor.
- B) Un *ServerSocket* se usa para que el servidor devuelva un *Socket* al cliente.
- C) Un *ServerSocket* se usa para que el cliente devuelva un *Socket* al thread principal.
- D) Ninguna de ellas

20. ¿Cuál de las siguientes afirmaciones es cierta?

- A) Un objeto que implementa *Iterator*, permite recorrer una colección bidireccionalmente.
- B) Un objeto que implementa *ListIterator*, permite recorrer una colección bidireccionalmente.
- C) *Iterator* extiende a *ListIterator*.
- D) *Iterator* implementa *ListIterator*.