Nombre: _____

1. Indica cuál de las siguientes afirmaciones es cierta:

- ☐ El diseño de un programa procedural está guiado por la división en tareas a realizar, mientras que el diseño orientado a objetos está dirigido a identificar quién tiene la responsabilidad de hacer las tareas.
  Esta es la correcta. En el programa procedural hay que definir las subrutinas (división de tareas) y en el programa orientado a objetos hay que definir las clases (quién tiene la responsabilidad).
- ☐ El diseño de un programa procedural está guiado por la identificación de quién tiene la responsabilidad de hacer las tareas, mientras que el diseño orientado a objetos está dirigido a la división del problema en tareas.
- ☐ El diseño de un programa procedural está guiado por la especificación de hechos y reglas lógicas que se cumplen en el problema, sin especificar el proceso de cómputo paso a paso, mientras que el diseño orientado a objetos está dirigido a identificar quién tiene la responsabilidad de hacer las tareas.
- ☐ El diseño de un programa procedural está guiado por la especificación de hechos y reglas lógicas que se cumplen en el problema, sin especificar el proceso de cómputo paso a paso, mientras que el diseño orientado a objetos está dirigido a la división del problema en tareas.

2. Indica cuál de las siguientes afirmaciones es cierta:

- ☐ El comportamiento ante un mensaje es idéntico para todos los objetos, con independencia de la clase a la que pertenezcan.
- ☐ El comportamiento ante un mensaje es idéntico para todos los objetos de una misma clase.
  Esta es la correcta. El comportamiento de un objeto de un determinado tipo (clase) a un mensaje está especificado en un método de la clase, por lo tanto diferentes objetos (instancias) de una clase se comportan del mismo modo ante el mismo mensaje.
- ☐ El comportamiento ante un método es idéntico para todos los objetos de una misma clase.
- ☐ El comportamiento ante un método es idéntico para todos los objetos, con independencia de la clase a la que pertenezcan.

3. El modelo DVD-20 de Filips es idéntico al modelo DVD-10, excepto que también permite visualizar películas en formato DivX y utiliza un algoritmo más eficiente para la reproducción de SVCD. Indica cuál sería la declaración correcta, aunque incompleta, en Java para especificar la relación entre estos dos tipos de DVD:

- ☐
```
class DVD20 extends DVD10 {
   public void playDivX () {...}
   public void playSVCD () {...}
}
```
- ☐
```
class DVD10 extends DVD20 {
   public void playDivX () {...}
   public void playSVCD () {...}
}
```

Esta es la correcta. El DVD-20 añade la funcionalidad (sobre el DVD-10) de mostrar DivX y además realiza la tarea de mostrar SVCD de forma diferente.

- ☐
```
class DVD20 extends DVD10 {
   public void playDivX () {...}
}
```
- ☐
```
class DVD20 extends DVD10 {
   public void playDivX () {...}
   public void playSVCD () { super.
       playSVCD();}
}
```

4. Sean las siguientes declaraciones clases:

```
interface I {...}
class A implements I {...}
class B extends A {...}
```

Indica cuál de las siguientes sentencias es correcta:

- ☐ `I x = new I ();`
- ☐ `B x = new I ();`
- ☐ `I x = new B ();`
- ☐ `B x = new A ();`

Esta es la correcta. A debe implementar todos los métodos de I, puesto que B extiende a A tiene todos los métodos de A y por lo tanto todos los de I.

5. Sean las siguientes declaraciones de clases

```
abstract class A{ public abstract void metodo(); }
class B extends A{public void metodo(int x){...}}
class C extends B {public void metodo(){...}}
```

y el siguiente segmento de código:

```
A[] a = new A[10];   //1
a[0] = new B();      //2
A c = new C();       //3
c.metodo();          //4
```

Indica cuál de las siguientes afirmaciones indica correctamente la aparición de un error en el código anterior.

- ☐ La linea 1, ya que la clase A es abstracta.
- ☐ La linea 2, ya que la clase B es abstracta.

  Esta es la correcta. La clase B no codifica el método abstracto de la clase A y por lo tanto sigue siendo abstracta y no es posible crear objetos de este tipo.
- ☐ La línea 4 da error, ya que metodo() es abstracto para la clase A.
- ☐ Todas las sentencias son correctas.

6. Considera estas dos posibilidades a la hora de definir las clases A y B

```
final class A{
   public void metodo1(){...}
   final public void metodo2(){}
}
class B extends A{}
```

```
class A{
   public void metodo1(){...}
   final public void metodo2(){}
}
class B extends A{
    public void metodo2(){}
}
```

¿Cuál de las siguientes afirmaciones se cumple para las clases anteriores?

- ☐ Solo la primera es incorrecta.
- ☐ Solo la segunda es incorrecta.
- ☐ Las dos versiones son incorrectas.
- ☐ Las dos son correctas.

Esta es la correcta. En el código de la izquierda la clase A es final y por lo tanto no se puede extender. En el código de la derecha metodo2() es final y no se puede sobreescribir en una clase derivada.

7. Considera la siguiente clase:

```
1  class A{
2     private static int metodo(int valor){
3        try{
4           valor = valor +1;
5           valor = valor + Integer.parseInt("aa");
6           valor = valor + 2;
7           System.out.println("Valor al final del try: " + valor);
8        }catch(NumberFormatException e){
9           valor = valor + Integer.parseInt("zz");
10          System.out.println("Valor al final del catch: " + valor);
11       }finally{
12          valor = valor + 3;
13          System.out.println("Valor al final de finally: " + valor);
14       }
15       valor = valor + 4;
16       System.out.println("Valor antes del return: " + valor);
17       return valor;
18    }
19    public static void main(String[] args){
20       try{
21          System.out.println(metodo(5));
22       }catch(Exception e){
23          System.err.println("Excepcion en metodo()");
24       }
25    }
26 }
```

Indica qué se muestra por pantalla al ejecutar este programa (las respuestas incorrectas a esta pregunta no restan puntos).

Valor al entrar a metodo(): 5
Línea 4: valor = 6
Línea 5: se lanza una excepción que se captura en la línea 8
Línea 9: Se lanza una excepción
Línea 12: valor = 9
Línea 13: Se muestra información y finaliza el método.
Línea 23: Se muestra información sobre el error.
Por lo tanto la salida por pantalla será:
Valor al final de finally: 9
Excepcion en metodo()

8. Indica cuál de las siguientes afirmaciones es correcta.

- ☐ Mediante el uso de semáforos el programador no tiene que responsabilizarse de la gestión del acceso exclusivo a un recurso.
- ☐ Tanto con monitores como con semáforos el programador no tiene que responsabilizarse de la gestión del acceso exclusivo a un recurso.
- ☐ Tanto con monitores como con semáforos el programador tiene que responsabilizarse de la gestión del acceso exclusivo a un recurso.
- ☐ Mediante el uso de monitores el programador no tiene que responsabilizarse de la gestión del acceso exclusivo a un recurso.

  Esta es la correcta. La exclusión mutua en el acceso al recurso está garantizada mediante el uso de monitores.

9. Dadas las siguientes clases:

```
class Hilo extends Thread{
   private int id;
   public Hilo(int i){
      id = i;
   }
   public void run(){
      int valor = id;
      for (int i=0; i<3;i++){
         valor = valor * valor;
         try{
            if (id <2)
               Thread.sleep(10000);
            else
               Thread.sleep(100);
         }catch(InterruptedException e){}
      }
      System.out.println(valor);
   }
}
```

```
class TestHilo{
   public static void main(String[] args){
      Hilo h1 = new Hilo(1);
      Hilo h2 = new Hilo(2);

      h1.run();
      h2.start();
      try{
         h1.join();
      }catch(InterruptedException e){}
      System.out.println("Fin");
   }
}
```

Indica cuál de las siguientes opciones muestra la salida por pantalla al ejecutar el programa:

☐ 1 Fin 256	☐ 256 1 Fin	☐ Fin 256 1	☐ 256 Fin 1

La correcta es la primera. A h1 se le envía el mensaje run() que se ejecuta como cualquier otro método (no concurrente). Al finalizar este método muestra un 1. h2 se lanza concurrentemente con el main(). h1.join() no tiene ningún efecto ya que el h1 no se ha lanzado. Se muestra Fin y cuando acaba h2 se muestra 256.

10. Las tareas realizadas por el método wait() de la clase Object son:

☐ Retirar el uso exclusivo del recurso al hilo en el que se ha ejecutado el método, almacenar dicho hilo en una cola y despertar a todos los hilos que estuviesen en espera para acceder al recurso.

☐ Retirar el uso exclusivo del recurso al hilo en el que se ha ejecutado el método y almacenar dicho hilo en una cola.

Esta es la correcta.

☐ Conceder el uso exclusivo del recurso al hilo en el que se ha ejecutado el método.

☐ Almacenar el hilo en el que se ha ejecutado el método en una cola y despertar a todos los hilos que estuviesen en espera para acceder al recurso.

11. Escribe la implementación de las clases (y/o interfaces) que creas conveniente para representar las operaciones matemáticas como la Suma, Resta, Multiplicacion y Division, de tal forma que la siguiente clase pueda ser compilada y ejecutada sin errores y reutilizando la mayor cantidad de código como sea posible.

```
class Matematicas{
   public static void main(String[] args){
      OperacionBinaria op = new Suma(4,3);
      System.out.println(op.opera());
      op = new Resta(4,3);
      System.out.println(op.opera());
      op = new Multiplicacion(4,3);
      System.out.println(op.opera());
      op = new Division(4,3);
      System.out.println(op.opera());
   }
}
```

```
// Una solución utilizando una interface

interface OperacionBinaria{
   public double opera();
}

class Suma implements OperacionBinaria{
   private double op1, op2;
   public Suma(double o1, double o2){
      op1 = o1;
      op2 = o2;
   }
   public double opera(){
      return op1 + op2;
   }
}

class Resta extends Suma{
   public Resta(double o1, double o2){
      super(o1,-o2);
   }
}

class Multiplicacion implements OperacionBinaria{
   private double op1, op2;
      public Multiplicacion(double o1, double o2){
         op1 = o1;
         op2 = o2;
   }
   public double opera(){
      return op1 * op2;
   }
}

class Division extends Multiplicacion{
   public Division(double o1, double o2){
      super(o1,1/o2);
   }
}

class Matematicas{
public static void main(String[] args){
   OperacionBinaria op = new Suma(4,3);
   System.out.println(op.opera());
   op = new Resta(4,3);
   System.out.println(op.opera());
   op = new Multiplicacion(4,3);
   System.out.println(op.opera());
   op = new Division(4,3);
   System.out.println(op.opera());

   }
}
```

```
// Una solución utilizando una clase abstracta

abstract class OperacionBinaria{
   protected double op1, op2;
   public OperacionBinaria(double o1, double o2){
      op1 = o1;
      op2 = o2;
   }
   abstract public double opera();
}

class Suma extends OperacionBinaria{
   public Suma(double o1, double o2){
      super(o1,o2);
   }
   public double opera(){
      return op1 + op2;
   }
}

class Resta extends Suma{
   public Resta(double o1, double o2){
      super(o1,-o2);
   }
}

class Multiplicacion extends OperacionBinaria{
   public Multiplicacion(double o1, double o2){
      super(o1,o2);
   }
   public double opera(){
      return op1 * op2;
   }
}

class Division extends Multiplicacion{
   public Division(double o1, double o2){
      super(o1,1/o2);
   }
}

class Matematicas{
public static void main(String[] args){
   OperacionBinaria op = new Suma(4,3);
   System.out.println(op.opera());
   op = new Resta(4,3);
   System.out.println(op.opera());
   op = new Multiplicacion(4,3);
   System.out.println(op.opera());
   op = new Division(4,3);
   System.out.println(op.opera());

   }
}
```

12. Sea la siguiente clase:

```
public class Indicador {
   static int presion = 0;
   static final int limiteSeguridad = 20;
   static Object sincObj;

   public static void main ( String[] args ) {
      sincObj = new Object();
      Trabajador[] t = new Trabajador[10];

      for (int i=0; i<10; i++) {
         t[i] = new Trabajador();
         t[i].start();
      }

      try {
         for (int i=0; i<10; i++)
            t[i].join();
      }
      catch (InterruptedException e){}

      System.out.println ("Lectura de indicador: " + presion + "(
          limite 20)");
   }
   static class Trabajador extends Thread {
      public void run () {
         synchronized (Indicador.sincObj) {
            if ( Indicador.presion < (Indicador.limiteSeguridad - 15)
                 ) {
               try {
                  sleep (100); //simula retraso en activar el control
               }
               catch (InterruptedException e){}
               Indicador.presion += 15;
            }
            else ; //no hacer nada presión demasiado elevada
         }
      }
   }
}
```

Supongamos que, debido a unas modificaciones en el diseño de la aplicación, nos piden que la clase Trabajador extienda a una clase Empleado que nos viene dada y que no podemos modificar. Se pide, realizar las modificaciones que sean necesarias en las clases anteriores para introducir esta modificación y que todo siga funcionando exactamente igual.

```
public class IndicadorSolucion {
   static int presion = 0;
   static final int limiteSeguridad = 20;
   static Object sincObj;

   public static void main ( String[] args ) {
      sincObj = new Object();
      Thread[] t = new Thread[10];

      for (int i=0; i<10; i++) {
         t[i] = new Thread(new Trabajador());
         t[i].start();
      }

      try {
         for (int i=0; i<10; i++)
         t[i].join();
      }
      catch (InterruptedException e){}

      System.out.println ("Lectura de indicador: " + presion + "(limite 20)");
   }
   static class Trabajador extends Empleado implements Runnable {
   public void run () {
      synchronized (IndicadorSolucion.sincObj) {
      if ( IndicadorSolucion.presion < (IndicadorSolucion.limiteSeguridad - 15) ) {
         try {
         Thread.sleep (100); //simula retraso en activar el control
         }
         catch (InterruptedException e){}
         IndicadorSolucion.presion += 15;
      }
      else ; //no hacer nada presión demasiado elevada
      }
   }
   }
}
```

Puntuación:
Preguntas 1 a 10: Respuesta correcta +1, respuesta incorrecta -0.25, no contestado 0
Pregunta 11: 3
Pregunta 12: 2

Duración: 2 horas