1. **¿Cuál o cuáles de las siguientes afirmaciones relativas a clases abstractas son FALSAS? (1,0 puntos)**
   - □ Las clases abstractas no se pueden instanciar
   - □ Todos los métodos de las clases abstractas deben ser métodos abstractos
   - □ Las clases abstractas no pueden ser subclases de clases no abstractas
   - □ Las clases abstractas no deberían ocupar el último nivel de una jerarquía de clases
   - □ El objetivo de las clases abstractas es favorecer la reutilización
   - □ Las clases abstractas pueden tener métodos finales
   - □ En las clases abstractas no puede realizarse polimorfismo de métodos
   - □ Las únicas clases que pueden tener métodos abstractos son las clases abstractas

2. **¿Cuál o cuáles de las siguientes afirmaciones relativas a las interfaces y a herencia son FALSAS? (1,0 puntos)**
   - □ Los interfaces no se pueden heredar por ninguna clase
   - □ Una clase abstracta no puede implementar ningún interface
   - □ Un interface no puede ser heredado por ningún interface
   - □ Cualquier clase de una jerarquía de clases puede invocar los métodos de un interface
   - □ El uso de los interface habilita la construcción de jerarquías con herencia múltiple
   - □ Un interface no puede heredar los métodos de una clase salvo que esa clase sea abstracta
   - □ Cualquier clase de una jerarquía de clases puede implementar un interfaz

3. **¿Cuál o cuáles de las siguientes afirmaciones relativas a jerarquías de clases son FALSAS?**
   - □ Las clases finales no pueden tener subclases
   - □ Las clases que no son ni finales ni abstractas no contribuyen a la reutilización de código
   - □ Las clases finales pueden ser clases abstractas
   - □ En una jerarquía debería haber siempre una única clase abstracta que se sitúa en el primer nivel (en la raíz) de la jerarquía
   - □ Las clases que no son finales pueden tener métodos finales
   - □ Una clase final podría ocupar el primer nivel de la jerarquía de clases

4. **¿Cuáles de las siguientes afirmaciones sobre herencia en Java son CIERTAS?**
   - □ Las clases padre deberían tener más atributos que las clases hijas
   - □ Las clases hijas son una extensión de las clases padre
   - □ Los constructores se heredan para favorecer la reutilización de código
   - □ Los modificadores de los atributos deben ser *private* para prevenir la copia indiscriminada de atributos en las clases
   - □ Si la $Clase_A$ es una subclase de la $Clase_B$, entonces en la clase $Clase_A$ la palabra reservada *super* se podrá usar para llamar únicamente a los métodos de la $Clase_B$, que son protegidos (*protected*) y públicos (*public*)
   - □ El puntero *super* favorece la reutilización de código entre las clases de la jerarquía

5. **Teniendo en cuenta el siguiente extracto de código (1,0 puntos)**

```
public class Mensajeria {
        public static float MAXIMO_CLIENTES_LIBRERIA = 500;
        private doublé paquetesNoRecibidos;
        protected Vector paquetesEnviadosPorDia;
        Hashtable <Vector<Paquete>, Fecha> paquetesEnviados;
        …
```

¿Cuál o cuáles de las siguientes afirmaciones son ciertas?

- □ El atributo *paquetesNoRecibidos* no podrá ser heredado por ninguna clase hija de la clase *Mensajeria*
- □ El atributo *paquetesEnviados* puede ser heredado por cualquier clase
- □ El atributo *paquetesNoRecibidos* no puede ser utilizado por los métodos de una clase hija de *Mensajeria*
- □ En el atributo *paquetesEnviadosPorDia* puede ser heredado por cualquier clase hija de la clase *Mensajeria*
- □ El atributo *MAXIMO_CLIENTES_LIBRERIA* no puede ser heredado por ninguna clase hija de la clase *Mensajeria* como una constante

- El interface y las clases se encuentran en archivos distintos con sus nombres correctos y las dos clases se encuentran en paquetes distintos.
- La clase *Libro* es la clase raíz de una jerarquía de clases sobre tipos de libros y tiene todos los *setters* y *getters* implementados.
- Es posible que existan tiendas en las que solo se vendan revistas de diferente tipo (por ejemplo, revistas científicas, del corazón, económicas, etc.) Además, la clase *TiendaRevistas* no puede ser abstracta.
- No hay que introducir ninguna línea de código adicional, pero es posible que haya que borrar alguna línea de código.

```
1   public interface Librería {
2       private static String DOMINIO= “Temas bibliográficos”; /* public final static */
3       public int getNumeroLibros( ) { };
4       public String getISBNLibro(String idLibro); /* Libro idLibro */
5       public ArrayList<Libro> obtenerLibros( );
6   }
7   public abstract clas LibreriaImpl extends Librería { /* implements */
8       ArrayList<Libro> listaLibros; /* falta protected o private */
9       public ArrayList<Libro> obtenerLibros( ) { return listaLibros; }
10      public String getISBNLibro(Libro libro) { return libro.getISBN( ); }
11      public getNumeroLibros( ); /* public int getNumeroLibros( ) { return listaLibros.size( ); } */
12      public int numeroLibrosAccion( ); /* public abstract int numeroLibrosAccion( ); */
13  }
14  public ~~final~~ class TiendaRevistas extends LibreriaImpl {
15      public int numeroLibrosAccion( ) {
16          super.numeroLibrosAccion( );
17          int num= 0;
18          for(int i=0; i<listaLibros.size( ); i++)
19              If(listaLibros.get(i).equals(“Accion”)) num++; /* listaLibros.get(i) instanceof Accion */
20          return num;
21      }
22      public int getNumeroLibros( );
23  }
```