1. **En programación orientada a objetos, ¿qué diferencia existe entre un objeto y un tipo de dato primitivo? (1.00 puntos)**
   - Los datos primitivos ocupan menos memoria que los objetos.
   - No hay ninguna diferencia.
   - Los objetos se almacenan en la pila y los atributos en el montón.
   - No es necesario llamar a un constructor para reservar memoria para los datos primitivos.
   - A través de los datos primitivos no se pueden invocar métodos (o funciones).
2. **En Java, los atributos de una clase: (1.00 puntos)**
   - Deberían ser siempre privados.
   - Si son primitivos se almacenan en el montón.
   - No tienen por qué tener siempre setters.
   - Deben inicializarse cuando se declaran.
   - Pueden ser accedidos por todos los métodos de la clase a laque pertenecen, independientemente de si esos métodos son públicos (public) o privados (private).
   - Tiene sentido que los getters sean privados (private).
   - Los únicos métodos que pueden modificar los valores de los atributos son los setters.
3. **¿Cuales de las siguientes afirmaciones sobre modificadores de acceso a atributos y métodos son FALSAS? (1.00 puntos)**
   - Si un atributo tiene acceso a paquete podrá ser accedido directamente por todos los métodos de cualquier clase.
   - El modificador de acceso a las constantes debería ser de tipo público (public).
   - La única forma de acceder a los atributos públicos (public) de una clase es a través de los métodos de dicha clase.
   - Un método privado (private) de una $Clase_A$ no puede acceder directamente a los atributos públicos (public) de una $Clase_B$.
   - Un método público (public) de una $Clase_A$ no puede acceder directamente a los atributos privados (private) de una $Clase_B$.
   - No tiene sentido que los métodos de una clase sean privados (private).
   - No tiene sentido que todos los métodos de una clase sean públicos (public).
4. **¿Cuáles de las siguientes sentencias sobre arrays (tipo[]), ArrayList y HashMap son CIERTAS? (1.00 puntos)**
   - int datos[10] reserva memoria para 10 enteros.
   - Los elementos de un ArrayList pueden ser arrays.
   - Si el tamaño del conjunto de datos es fijo, es preferible usar un array en vez de un ArrayList.
   - El acceso a un elemento de un array y de un ArrayList se realiza a través de un índice.
   - Si datos es una instancia de ArrayList <String>, for(String dat: datos) es correcto.

- Si datos es una instancia de HashMap<String, Integer>, for(Integer dat: datos) es correcto.
- Si datos es una instancia de HashMap<String, Integer>, se puede acceder a los datos a través de una clave de tipo entero.

5. **¿Cuáles de las siguientes afirmaciones sobre constructores son CIERTAS o CORRECTAS? (1.00 puntos)**

   - Un constructor puede tener acceso a paquete, pero no debería tenerlo.
   - Los constructores se invocan para reservar memoria para cualquier tipo de dato.
   - Un constructor puede devolver cualquier tipo de dato.
   - No es necesario implementar el constructor sin argumentos, ya que Java lo proporciona siempre por defecto.
   - En los constructores se deberían de reservar memoria para todos aquellos objetos de una clase que no sean constantes.
   - Un constructor puede tener acceso privado (private), pero no debería tenerlo.

6. **¿Cuál o cuáles de las siguientes situaciones se corresponden con lo que entendemos por aliasing? (1.00 puntos)**

   - La creación indiscriminada de referencias a una misma variable.
   - La creación de varias instancias de una clase usando el mismo constructor para todas esas instancias.
   - Dar dos o mas nombres a una misma variable que es del tipo objeto.
   - La instanciación de una variable en más de una ocasión.
   - Integer x = 8;
   - El uso del operador asignación (=) entre dos objetos del mismo tipo.
   - String x = new String ("Nombre").

7. **Dado un atributo att1 de tipo entero (int), ¿cuáles de las siguientes opciones son CIERTAS o CORRECTAS? (1.00 puntos)**

   - static int att1 no tiene por qué ser privado (private), (non).
   - final int att1 es una constante.
   - att1 = 10;
   - att1 = new Integer(10);
   - listaInt.add(att1) donde listaInt es una instancia de ArrayList
   - Si no se le asigna ningún valor en el constructor, el valor por defecto del atributo es 0.

8. **Teniendo en cuenta el siguiente extracto de código, (1.00 puntos)**

```
%public class TierraMedia{
    public static int TROPA_ORCOS = 10000;
    private ArrayList orcos;
    ArrayList <Rohirrim> rohirrims;
    private HashMap<ArrayList<Hombre>, Capitan> hombres;
    private HashTable<String, CiudadTierraMedia> ciudades;
```

¿Cuál o cuáles de las siguientes sentencias son CIERTAS?

- TROPA_ORCOS debería ser *final* para que pueda ser accedido por otras clases sin tener que instanciar la clase *TierraMedia*.
- El atributo *rohirrims* no puede ser accedido directamente por todas las clases del programa.
- Con el atributo *hombres* es posible obtener el conjunto de hombres que sirven a un capitán dado.
- El atributo *orcos* podría contener cualquier tipo de personajes del programa (*Hombres, enanos, Elfos, Magos, etc..*).
- El atributo *hombres* puede ser accedido directamente por todas las clases del programa.
- El atributo *ciudades* puede ser accedido directamente por los métodos públicos y privados de la clase *TierraMedia*.

9. Indicar y corregir los errores del código de la clase "Actor" asumiendo que, (2.00 puntos)

   - El método *actorMayorSueldo* devuelve CIERTO si existe algún actor quehaya participado en algunas de las películas del acto y que tenga un sueldo mayor que éste. En caso contrario, devuelve FALSO. Este método puede ser accedido desde cualquier clase.
   - La clase *Pelicula* tiene atributo *actores* que indica el conjunto de actores que han participado en la película.
   - Se considera que todos los atributos de las clases *Actor* y *Pelicula* tienen sus *setters* y *getters* implementados.
   - No hay que introducir ninguna línea de código adicional.

```
%public class Actor {

   String nombre; //private String nombre;
   private HashMap<Pelicula, String>peliculasParticipado; //
      HashMap<String,Pelicula>;
   private float sueldo=0; //private float sueldo;

   public void Actor (String nombre) {

      nombre = getNombre(); //this.nombre = nombre;
      peliculasParticipado = new ArrayList<Pelicula>(15);

   }

   boolean actorMayorSueldo () { //public boolean
      actorMayorSueldo()

      boolean mayor = false;
      Iterator<Pelicula> conjuntoP =
         peliculasParticipado.values();
         //peliculasParticipado.values().iterator()

      while(!conjuntoP.hasNext()) {
         //while(conjuntoP.hasNext());

         Pelicula peli; // Peli peli = conjuntoP.next();
         ArrayList<Actor> actores = peli.getActores();

         for(Actor actores : actor) { // for(Actor actor :
            actores)

            if(actor.getSueldo() > setSueldo()) //
               if(actor.getSueldo() > this.sueldo)
            mayor = false; // mayor = true;
         }
      }
   return mayor;
   }

}
```