RELACIÓN DE PROBLEMAS 10
CONCEPTOS AVANZADOS DE P.O.O.

1. Realizar una clase **`CuentaCredito`** que herede de la clase Cuenta y que cumpla:
   - Al crear una cuenta de crédito se indica de qué cantidad de crédito se dispone, es decir, a cuánto puede quedar la cuenta en números rojos. Por ejemplo, si el crédito es de 100€ la cuenta podrá llegar a tener un saldo igual a –100.
   - Inicialmente, si no se indica nada, el **saldo** de la cuenta es 0€.
   - Inicialmente, si no se indica nada, el **crédito** es de 100€.
   - Se deben incluir los métodos get y set para el crédito. El crédito nunca puede superar los 300€. También habrá que tener en cuenta el saldo actual de la cuenta.
   - Se deberá modificar los métodos de **`sacarDinero`** para incluir el crédito.

   Realizar una clase de prueba **`MenuCuentaCredito`** que cree una cuenta de crédito y presente un menú con estas opciones.
   1. Ingresar dinero
   2. Sacar dinero
   3. Mostrar saldo y crédito
   4. Salir

   Realizar un programa Principal que cree una cuenta ahorro joven y pruebe sus métodos

2. Se desea desarrollar una aplicación que permita calcular el precio en una empresa de alquiler de vehículos. Cada vehículo se identifica por su matrícula. Los vehículos pueden ser de gama alta, media o baja. La empresa alquila 3 tipos de vehículos: coche, microbús y furgoneta de carga.
   El precio base de alquiler del vehículo es de 30 euros al día si es de gama baja, 40 si el de gama media y 50 si es de gama alta. En caso de alquiler de un coche al precio base se le suma la cantidad de 3.5 euros por día si el vehículo es gasolina y 2 euro por día si el vehículo es diesel. En caso de alquiler de un microbús se le añade la cantidad de 5 euros por plaza que disponga el microbús.
   En el caso de furgonetas, al precio base se le añade 0,5 euros * PMA (peso máximo autorizado)

   Se debe presentar un menú con las siguientes opciones:

   1. **Alta de vehículo**: Se solicitará el tipo de vehículo y sus datos y lo dará de alta.
   2. **Cálculo de precio de alquiler**: Se solicitará la matrícula del vehículo, el número de días que ha sido alquilado y se mostrará el precio del alquiler
   3. **Salir**.

Considerar que la empresa trabajará con un máximo de 200 vehículos

3. Se va a programar un juego de rol, del cual se nos ha encargado programar en Java parte del esquema de personajes. Para ello se nos han dado las siguientes directrices:
    - Tenemos que programar 2 tipos de personajes: los **`Magos`** y los **`Clerigos`**.
    - Todos los personajes cuentan con los siguientes datos:
        - Nombre: una cadena.
        - Raza: una cadena que puede tomar los valores “humano”, “elfo”, “enano” u “orco”.
        - Fuerza: un entero entre 0 y 20.
        - Inteligencia: un entero entre 0 y 20.
        - Puntos de vida máximos: un entero entre 0 y 100.
        - Puntos de vida actuales: un entero entre 0 y puntos de vida máximos.

    Además cada tipo de personaje tiene atributos y restricciones específicos que se detallarán en los apartados correspondientes.

    Se pide:

    Apartado 1:
    Escribe una clase **`Personaje`** que reúna los atributos mencionados en el enunciado. Dicha clase debe incluir:
    - Un constructor para poder inicializar los atributos (se supone que los puntos de vida actuales son iguales a los máximos al inicializarse).
    - Métodos **`get`** y **`set`** para todos los atributos de la clase y **`toString`**

    Apartado 2:
    Escribe la clase **`Mago`** teniendo en cuenta las siguientes restricciones:
    - Un Mago no puede tener en inteligencia un valor menor que 17 ni en fuerza un valor mayor que 15.
    - Además un `Mago` almacena los nombres de los hechizos que ha memorizado. Un mago sólo puede memorizar a la vez un máximo de 4 hechizos. Impleméntalo como un array y añade los siguientes métodos:
        - **`aprendeHechizo`**: que tiene un parámetro de tipo `String` y que añade un hechizo al array (deberá buscar un hueco libre en el array).

    - **`lanzaHechizo`**: que tiene como parámetro un objeto de tipo Personaje que será el personaje sobre el que recae el hechizo y el `String` correspondiente a un hechizo. Las acciones a tomar serán las de restar 10 de los puntos de vidas actuales de dicho personaje y olvidar el hechizo (borrarlo del array).
- Escribe el constructor de la clase (puedes suponer que en el momento de su creación, un mago no conoce ningún hechizo).
- Reescribe el método `toString` para que se muestren los nuevos datos incluyendo la lista de hechizos.

Apartado 3:
Escribe la clase **`Clerigo`** teniendo en cuenta las siguientes restricciones:

- Un `Clerigo` no puede tener una fuerza con un valor menor que 18 y una inteligencia con un valor menor que 12 ni mayor que 16, ambos incluidos.
- El `Clerigo` reza a un dios para obtener el don de la curación. Por lo tanto se deberá modificar el constructor genérico para aceptar el nombre del dios, del cual el `Clerigo` es devoto y así poder almacenarlo.
- Un `Clerigo` tiene el don de curar, por lo tanto, la clase deberá tener un método **`curar`**, que recibe como parámetro un objeto de tipo Personaje sobre el que recae la acción de curar y que aumenta en 10 los puntos de vida.
- Reescribe el método `toString` para que muestre además de los datos básicos el nombre del dios.

Apartado 4:
Escribe una clase de prueba con un método **`main`** en la que se creen 2 `Magos` (A y B) y un `Clerigo` (C) y en el que tengan que realizar las siguientes acciones:

- Imprimir los datos de los tres personajes.
- El `Mago` A aprende 2 hechizos.
- El `Mago` B aprende 1 hechizo.
- Imprimir los datos de los `Magos`.
- El `Mago` A lanza un hechizo sobre el `Mago` B.
- El `Mago` B lanza un hechizo sobre el `Mago` A.
- El `Clerigo` cura al `Mago` B.
- El `Mago` A lanza un hechizo sobre el `Mago` B.
- Imprimir los datos de los tres personajes.

NOTA: los casos en los que un atributo pueda tomar valores no permitidos hay que controlarlos lanzando la excepción **`PersonajeException.`**

4. Sobre el ejercicio anterior, realizar un programa principal con el siguiente menú:

    1. **Alta de personaje**: Se solicitará si desea crear un `Mago` o un `Clérigo`. Se solicitarán los datos correspondientes y se creará el `Personaje`.
    2. **Mago aprende hechizo**: Se solicitará el nombre del `Mago` y el nombre del hechizo que aprende. Un `Mago` no puede aprender un hechizo que ya tenía aprendido.
    3. **Mago lanza hechizo**: Se solicitará el nombre del `Mago` y el nombre del `Personaje` al que quiere lanzar un hechizo. Debe controlarse que no puede lanzarse un hechizo a él mismo. Si al lanzar el hechizo, el personaje se queda con menos de cero puntos, el personaje muere pero no se eliminará.
    4. **Clérigo cura al mago**: Se solicitará el nombre del `Clérigo` y el nombre del `Mago` al que quiere curar, y se curará. Un `Clérigo` puede curar a un `Personaje` que ya está muerto.
    5. **Mostrar el listado de personajes**
    6. **Mostrar el listado de personajes ordenados por puntos actuales de mayor a menor**
    7. **Salir**

    Existirán un máximo de 100 personajes en el juego.

5. Se desea desarrollar un programa Java que permita representar la siguiente situación. Una instalación deportiva es un recinto delimitado donde se practican deportes, en Java interesa disponer de un método int getTipoDeInstalacion(). Un edificio es una construcción cubierta y en Java interesa disponer de un método double getSuperficieEdificio(). Un polideportivo es al mismo tiempo una instalación deportiva y un edificio; en Java interesa conocer la superficie que tiene y el nombre que tiene. Un edificio de oficinas es un edificio; en Java interesa conocer el número de oficinas que tiene.

    Definir las interfaces y las clases para representar la situación anterior. En una clase Principal con el método main que cree un array de tamaño 5 que contenga tres polideportivos y dos edificios de oficinas y después se muestre el contenido del array.

6. Dada la siguiente interfaz:

```
public interface CreableEstadisticas
{
        double minimo();
        double maximo();
        double media();
```

```
}
```

- Modificar la clase **`ArrayPersonajes`** del Ejercicio 5 para que implemente también la interfaz. Devolverá los puntos de vida mínimos, puntos de vida máximo y media de los puntos de vida de los personajes.
- Probar los métodos anteriores.