Programación II

Curso 21-22

Práctica 3 – Parte a

1 Cuestiones previas

Crea el directorio P3 dentro del directorio `Practicas`. Guarda en P3 todos los ficheros fuente y **.txt** de esta práctica y los **.class** generados, conformando el paquete P3.

2 Actividad 1: crear la clase `ElectricCar`

Siguiendo el esquema de la información presentada en la práctica 2, un coche eléctrico lo caracterizamos por los siguientes atributos básicos (a los que no se podrá acceder directamente desde el exterior de la clase):

- **plate**: una cadena de longitud 7 para la matrícula (4 dígitos y 3 letras).
- **manufacturer**: una cadena de longitud arbitraria.
- **electricPower**: un número entero comprendido entre 50 y 800 (en CV).
- **batteryCharge**: un número real entre 0.0 y 100.0, que indica el porcentaje de carga de la batería.

En los rangos anteriores, los valores mínimos y máximos han de ser valores comunes para todos los objetos de la clase (el valor de cada atributo será compartido por todos los objetos). Los valores de la potencia eléctrica han de poder modificarse. Los de la carga de la batería serán constantes.

Para representar un coche eléctrico creamos la clase `ElectricCar`, parcialmente definida en el siguiente cuadro:

```
public class ElectricCar {
        // ---- Atributos de instancia (completar con el resto de atributos de instancia)
        … plate; // completar visibilidad y tipo

        // ---- Atributos estáticos (completar con el resto de atributos estáticos)
        … PLATE_FORMAT = "DDDDLLL"; // D=0..9, L=A..Z, constante accesible desde el exterior de la clase

        // constructor vacío
        public ElectricCar () {
        }
        // Constructor con argumentos
        public ElectricCar (… plate, … manufacturer, … power, … batteryCharge) { // completar tipos
                this.plate = plate;   // Completar el código
        }

        // Métodos. Completar tipos y visibilidad para que sean accesibles desde el exterior de la clase
        // ---- Métodos estáticos o de clase (completar con el resto de métodos estáticos)
        … boolean isValidPlate (… plate) { // completar la visibilidad y los tipos
                … // completar el código
        }

        // ---- Métodos de instancia (completar getters, setters y toText)
        … getPlate () {
                return this.plate;
        }

        … setPlate (… plate) {
                 this.plate = plate;
        }
        … String toText () {
                … // completar el código, generando una cadena conforme al formato de representación
                  // de un coche en fichero de la práctica 2 (sin las multas), por ejemplo: E;MAT;FAB;PE;POR
        }
}
```

Todos los métodos (de instancia y de clase) serán accesibles desde el exterior de la clase.

Copia el código anterior y complétalo (en los puntos suspensivos), según se indica:

- Completa los atributos de instancia y estáticos con su visibilidad correcta.
- Completa el código del constructor con argumentos y el del método `isValidPlate`.
- Crea el resto de métodos estáticos con su código de validación correspondiente.
- Completa los getters, setters y el método `toText`.

3 Actividad 2: crear y manipular objetos de clase `ElectricCar`

Copia la clase `P3a`, parcialmente definida en el siguiente cuadro, que contiene el método `main`.

```
public class P3a {
    public static void main(String[] args)  {
    // atributos de varios coches
        String plate3="6288LYM", manufacturer3="Ford";     int power3=110;  float batteryCharge3=120.5F;
        String plate4="07451JMR", manufacturer4="Seat";    int power4=60;   float batteryCharge4=10.1F;
        String plate5="3400JXK", manufacturer5="Peugeot";  int power5=70;   float batteryCharge5=50F;
        String plate6="0041LDR", manufacturer6="Seat";     int power6=20;   float batteryCharge6=10.1F;
    }
}
```

A partir de esta clase, crea código en el método `main` para lo siguiente:

1. Usando el constructor con argumentos de la clase `ElectricCar`, crea un objeto con los siguientes valores: *plate*="1111KLS", *manufacturer*="SEAT", *electricPower*=220, *batteryCharge*= 30,5.
2. Usando el constructor vacío y los setters, crea otro objeto con los siguientes valores para sus atributos: *plate*="2222LSX", *manufacturer*="FORD", *electricPower*=220, *batteryCharge*= 30,5.
3. Usando el método `toText`, saca por pantalla los datos de los dos coches creados anteriormente, tal como lo genera el método.
4. Usando cualquiera de los constructores que desees, crea objetos según los datos para los coches 3, 4, 5 y 6 detallados en el código anterior, siempre y cuando todos los valores de sus atributos sean correctos. Para comprobar si son correctos, usa los atributos y métodos estáticos de la clase.

   Si el valor de la matrícula es incorrecto, saca el siguiente mensaje por pantalla:

   "Incorrect plate value: <PLATE>. Not comply with format <PLATE_FORMAT>"

   donde <PLATE> representa la matricula incorrecta y <PLATE_FORMAT> el valor obtenido del atributo estático correspondiente.

   Si el valor de potencia es incorrecto, saca el siguiente mensaje por pantalla:

   "Incorrect power value: <NNN>. Valid range is <MIN>-<MAX>"

   Si el valor de carga de batería es incorrecto, saca el siguiente mensaje por pantalla:

   "Incorrect battery charge value: <NNN>. Valid range is <MIN>-<MAX>"

   donde <NNN> representa el valor incorrecto, y <MIN>/<MAX> representan los correspondientes valores de mínimo/máximo, obtenidos del atributo estático correspondiente de la clase.

   Si todos los datos son correctos crea el objeto y preséntalo en pantalla usando el método `toText`, como en el punto 3 anterior.

   Prueba adecuadamente el código, modificando manualmente los valores de los datos de los coches.

4 Actividad 3: Crear las clases `CombustionCar` y `HybridCar`

Tomando como modelo todo lo desarrollado para la clase `ElectricCar`, crea ahora dos clases nuevas para modelar los coches de combustión interna y los coches híbridos:

- Clase `CombustionCar`, con los siguientes atributos básicos:
  - **plate**.
  - **manufacturer**.
  - **mechanicalPower**: un número entero comprendido entre 60 y 500.
- Clase `HybridCar`, con los siguientes atributos básicos:
  - **plate**.
  - **manufacturer**.
  - **mechanicalPower**: un número entero comprendido entre 60 y 500.
  - **electricPower**: un número entero comprendido entre 20 y 100.
  - **batteryCharge**: un número real entre 0.0 y 100.0.

5 Actividad 4: Trabajar con arrays de coches

Añade código en la clase `P3a` para realizar lo siguiente:

1. Crear 3 arrays: uno de objetos `ElectricCar`, otro de objetos `CombustionCar` y otro de objetos `HybridCar`. Todos de tamaño 25. Llama a estos arrays *electricCityCars*, *combustionCityCars* y *hybridCityCars*, respectivamente. Decláralos en la zona de datos globales de la clase `P3a`, de forma que sean visibles para todos los métodos de esta clase.

2. Crear un nuevo método:

   ```
   public void readCityCarsFile (String filename) {...}
   ```

   que lea los coches del fichero *cityCars.txt*[1], pasándole en el argumento el nombre de este fichero. Si el coche es eléctrico, crea un objeto `ElectricCar` y añádelo al array de coches eléctricos; si es de combustión, añádelo al de coches de combustión; y si es híbrido, al array de coches híbridos. Crea los objetos y añádelos a los arrays siempre y cuando todos sus datos sean correctos; si alguno de los datos de cualquiera de los coches es incorrecto[2], simplemente descarta esa línea (coche) del fichero. Puedes considerar que en el fichero no existen coches con matrículas repetidas.

3. Calcula la suma de la potencia total (eléctrica + mecánica) de todos los coches de los 3 arrays, y muéstrala por pantalla según el formato *"Suma de potencia = resultado"*. Para ello, crea un nuevo método que calcule y devuelva la suma:

   ```
   public int computeTotalPower () {…}
   ```

   Pista: los dos anteriores métodos no son estáticos

4. Incrementa el nivel de batería de todos los coches eléctricos un 10% y después guarda en el fichero *ElectricCarsOutput.txt* los coches del array de coches eléctricos, manteniendo el formato del fichero de coches. Guárdalos en el fichero de salida en el mismo orden en que los leíste del fichero de entrada. Si el fichero ya existe, sobreescríbelo. Crea para guardar los coches el siguiente método en la clase P3a:

   ```
   public static void writeElectricCityCarsFile (String fileName) {...}
   ```

Para implementar el apartado 4, crea un nuevo método en las clases `ElectricCar` y `HybridCar`, llamado `increaseBatteryChargeLevel`, pasándole como argumento un número real[3].

```
public void increaseBatteryChargeLevel (float increment) {...}
```

El código de la clase `P3a` llamará a este método sobre todos los objetos `ElectricCar`, y luego guardará los coches eléctricos en el fichero.

---

[1] Este fichero está en el mismo formato que el del fichero *cars.txt* de la práctica 2 (sin la secuencia de multas).

[2] Usa los **métodos estáticos** de las clases para determinar si los datos de los coches son correctos.

[3] Si el nuevo nivel calculado es mayor de 100, debes truncarlo a 100.