DeTodo

Ejercicios Variados

1. **Clases básicas, composición de clases y ArrayList**
2. **Herencia**
3. **Matrices**
4. **Clases Abstractas y polimorfismo**
5. **Excepciones**
6. **HashMap y archivos de texto**
7. **Fuentes de información**

1. Clases básicas, composición de clases y ArrayList

Ejercicio01. **Figuras de superhéroes**.

2. Herencia

Ejercicio02. **Electrodomésticos**.

3. Matrices

Ejercicio03. **Juego 2048**.

4. Clases Abstractas y polimorfismo

Ejercicio04. **Servicios**.

5. Excepciones

Ejercicio04. **Parking**.

Crear una aplicación sencilla en Java que controle la ocupación de un Parking, a través de la entrada y salida de coches. Esta aplicación podrá lanzar errores que serán gestionados a través de Excepciones Java.

Se lanzarán distintas excepciones que luego se capturarán usando las instrucciones `try…catch`.

También se creará una clase de excepción propia que se usará para generar errores propios.

Clase Parking

Atributos:

- `matrículas`: un ArrayList de String llamado matrículas.

  Este ArrayList representará las distintas plazas del parking, y contendrá las matrículas de los coches aparcados en cada plaza.

  Por ejemplo, si el coche con matrícula 1234-KW aparca en la plaza 23, entonces introduciremos en el ArrayList la cadena "1234-KW" en la posición 23 del array.

  Si la plaza 45 estuviera libre, entonces la posición 45 del array contendrá el valor null.
- `nombre`: será un String y se corresponderá con el nombre del parking.

Métodos:

- *Constructor*:recibirá el nombre del parking y un entero, indicando el número de plazas del parking, y construirá el ArrayList matrículas con ese tamaño e inicializando cada posición a null.
- `public String getNombre()`: devuelve el nombre del parking.
- `public void entrada(String matricula, int plaza)`: este método introduce el coche con la matricula indicada en la plaza del parking indicada.

  Control de errores:
  - Si la matrícula tiene un tamaño menor de 4 caracteres o es null, se lanzará una excepción con el mensaje "Matrícula incorrecta" .
  - Si la plaza ya estuviera ocupada se lanzará una excepción con el mensaje "Plaza ocupada"
  - Si la matrícula existiera dentro del parking se lanzará una excepción con el mensaje "Matrícula repetida".
- `public int salida(String matricula)`: este método hace que el coche con la matrícula indicada salga del parking, liberando la plaza correspondiente. Devolverá el número de la plaza liberada.

  Control de errores:
  - Si la matrícula no existe dentro del parking se lanzará una excepción con el mensaje "Matrícula no existente".

- `public int getPlazasTotales()`: este método devuelve las plazas totales del parking.
- `public int getPlazasOcupadas()`: este método devuelve las plazas ocupadas del parking.
- `public int getPlazasLibres()`: este método devuelve las plazas libres del parking.
- `public String toString()`: la sobrecarga de este método devolverá una cadena que contendrá el nombre del parking y un listado de cada plaza y la matrícula que la ocupa.

  Por ejemplo:

```
Parking Avenida
---------------
Plaza 0: (vacía)
Plaza 1: 3322AB
Plaza 2: 5342HW
Plaza 3: (vacía)
…
```

Clase principal

En la clase principal realice un programa de prueba que haga lo siguiente:

Crear un objeto Parking:

- Crea un objeto Parking llamado “Parking Centro” con 10 plazas.

Menú de opciones:

- Después de crear el objeto Parking, se mostrará al usuario un menú con cuatro opciones: 1) Entrada de coche y 2) Salida de coche, 3) Mostrar parking y 4) Salir del programa.
- Se le pedirá al usuario que introduzca un valor entero, entre 1 y 4, correspondiente a una de las opciones (use la clase Scanner y el método nextLine, convirtiendo la entrada a entero).

Opción 1. Entrada de coche:

- La opción Entrada de coche le pedirá al usuario que introduzca por teclado la matrícula del coche que entra y la plaza donde se colocará, y a continuación se introducirá el coche en el parking en dicha plaza. La opción mostrará luego el número de plazas totales, ocupadas y disponibles después de la entrada.
- Para pedir la plaza, use el método nextLine de la clase Scanner y convierta el dato introducido por teclado a entero.
- Se capturarán las excepciones si las hubiera, y en ese caso se mostraría el mensaje de error y la entrada no se llevaría a cabo.

Opción 2. Salida de coche:

- La opción Salida de coche le pedirá al usuario que introduzca por teclado la matrícula del coche que sale y a continuación se mostrarán las plazas totales, libres y ocupadas del parking después de la salida.
- Se capturarán las excepciones si las hubiera, y en ese caso se mostraría el mensaje de error y la entrada no se llevaría a cabo.

Opción 3. Mostrar parking:

- Esta opción muestra todas las plazas y las matrículas que las ocupan (se usará el método toString de la clase Parking).

Opción 4. Salir del programa:

- El programa mostrará el menú repetidas veces hasta que se pulse esta opción.

Mejoras Finales

1. La entrada de la opción numérica del menú puede causar una caída del programa si el usuario introduce un texto en vez de un entero. Evite esto usando un try...catch en la línea donde se pide la opción del menú elegida.
2. Es posible crear clases de Excepciones propias. Cree una clase ParkingException que construya una excepción con un mensaje y una matrícula, y contenga un método getMensaje que devuelva el mensaje y otro método getMatricula que devuelva la matrícula del coche que ha producido el error.

   > Nota: esta clase heredará de la clase Exception.

   Una vez realizada esta clase, úsela para lanzar las excepciones de la clase Parking.

   Modifique también el programa principal de forma que los `try…catch` de la entrada de coches y la salida sean capaz de capturar excepciones de tipo ParkingException y de otros tipos.

6. HashMap y archivos de texto

Ejercicio05. **Diccionario**.

Diseñar un diccionario español-inglés. El ejercicio hace uso de un **HashMap** para almacenar las parejas de palabras español-inglés del diccionario. Además, todas esas parejas se cargan inicialmente desde un **archivo de texto**.

Se creará una clase «**Diccionario**» que envuelva el HashMap y que sirva para dar funcionalidades a la clase principal. Como prueba de esta clase se creará un pequeño juego en el que se harán preguntas de palabras en español al usuario y éste tendrá que escribir la palabra correspondiente en inglés.

Clase Diccionario

La clase definirá una colección de parejas de palabras en español-inglés.

Por ejemplo:

```
Coche – Car
Perro – Dog
Mesa – Table
Etc.
```

Propiedades:

- Un HashMap con pares clave-valor ambos del tipo String.

Métodos:

- *Constructor*: el Diccionario construye la propiedad diccionario como un HashMap vacío.
- `nuevoPar`: este método recibirá una palabra en español y otra en inglés y las introducirá en el HashMap como nuevo par clave-valor.
- `traduce`: este método recibirá una palabra en español, la buscará en la propiedad diccionario y devolverá su correspondiente valor en inglés.
- `palabraAleatoria`: este método no recibirá ningún parámetro. El método devolverá aleatoriamente una de las palabras en español del diccionario.
- `primeraLetraTraduccion`: este método recibirá una palabra en español y devolverá la primera letra de su correspondiente palabra en inglés.

Clase Principal

La clase Principal debe instanciar un objeto de la clase Diccionario, y rellenarlo con parejas de palabras español-inglés. Estas palabras deben obtenerse de un archivo *palabras.txt* que estará almacenado en la carpeta del proyecto. El contenido del archivo palabras.txt tendrá el siguiente formato:

```
Coche;Car
Perro;Dog
Mesa;Table
…
```

Una vez cargado el objeto Diccionario con las palabras obtenidas del archivo de texto, el programa iniciará un pequeño cuestionario, en el que le preguntará al usuario la traducción de una palabra en español. El usuario escribirá la respuesta en inglés y el programa le dirá si acertaste o no. La salida del programa será similar a la siguiente:

```
Perro: D…
Indique la respuesta: dog
¡CORRECTO!

Mesa: T…
Indique la respuesta: Tabla
¡NO! La respuesta correcta es Table

Coche: C…
Indique la respuesta: fin
FIN DEL PROGRAMA

Total preguntas: 2
Total aciertos: 1
Total errores: 1
Aciertos: 50%
```

Observa los siguientes detalles:

- Se proporciona la primera letra de la palabra en inglés como pista.
- El cuestionario continua hasta que el usuario introduce la palabra “fin”.
- Al finalizar se muestra un resumen con el total de preguntas, aciertos, errores y el porcentaje de aciertos obtenido.

7. Fuentes de información

- EjerciciosMesa