1.- Se pide hacer una aplicación para controlar las entradas y salidas de coches de un aparcamiento con N plazas (lo establecemos con una constante).

La aplicación presenta el siguiente menú:

Elija una opción:

1) Entrada de coche
2) Salida de coche
3) Mostrar situación del aparcamiento
4) Salir del programa

Opción 1: Se elige cuando un coche intenta entrar y aparcar en una plaza. La persona que controla la entrada toma nota de la matrícula y del NIF del conductor y le pregunta si es socio. Después le adjudica un número de plaza (la que le parezca). Introduce los datos en la aplicación, que hará a continuación las siguientes comprobaciones. Para controlar los errores crearás una excepción propia de nombre "ExcepcionAparcamiento", que mostrará en cada caso los mensajes indicados entre paréntesis, haciendo uso del método getMessage():

1.- El número de plaza es válido (1...N). ("Plaza inexistente")
2.- No existe ningún coche aparcado (en cualquier plaza) con esa matrícula. ("Matricula repetida")
3.- La matrícula tiene al menos 4 caracteres. ("Matricula incorrecta");
4.- La plaza adjudicada está libre. ("Plaza ocupada");
5.- Si el número de plaza se da en un formato no numérico se debe advertir. (Capturamos la excepción de Java y personalizamos el mensaje)

En cualquiera de los 5 casos, el programa vuelve a mostrar el menú y pedirá de nuevo todos los datos para registrarlos.

Los datos del conductor (si es socio (booleano) y su NIF (String)) los guardarás en una clase de nombre Conductor. De esta forma, en la clase donde guardamos los datos de cada coche aparcado (clase Coche) guardaremos el número de la matrícula y un objeto Conductor cuando se haya conseguido aparcar.

Para controlar en todo momento la plaza en la que el coche aparca, esta se ligará a la posición que ocupa el objeto coche aparcado en un ArrayList, donde se colocará cada coche que se aparca. De esta forma, tendremos que la posición 0 corresponde a la plaza 1 y estará ocupada por el coche que consiga aparcar en dicha plaza, y así con el resto de las plazas. Detectamos que una plaza está libre porque la posición correspondiente apunta a null.

Una vez que el coche ha conseguido aparcar, se indicará que el coche de la matrícula dada ha aparcado en la plaza que se le haya asignado.

Opción 2: Se pedirá el número de matrícula del coche que sale, y si puede salir se mostrará el mensaje comunicando que el coche ha salido indicando la plaza que ocupaba.

Si el sistema detecta que no hay ningún coche aparcado con el número de matrícula introducido se lanzará la excepción "ExcepcionAparcamiento" con el mensaje "Matricula no existente".

Si la matrícula del coche corresponde con la matrícula de algún coche que está en nuestro aparcamiento, se procederá a reflejar en nuestro ArrayList que la plaza vuelve a estar vacía haciendo que la posición del ArrayList de la plaza apunte a null.

Cuando el coche consigue salir del aparcamiento y el hecho registrado, la aplicación indica qué matrícula ha salido y qué plaza ha quedado libre.

Opción 3: La aplicación muestra los datos de cada plaza, indicando si está vacía o si está ocupada. En caso de estar ocupada, qué matrícula la ocupa y si se trata de un socio del aparcamiento o no y su NIF. Se muestra un ejemplo después de haber aparcado un coche en la plaza 5 (solo he puesto 5 plazas para el ejemplo):

Descripción de todas las plazas aparcamiento "El Económico"

Plaza 1: (vacía)
Plaza 2: (vacía)
Plaza 3: (vacía)
Plaza 4: (vacía)
Plaza 5: Matrícula BCD - 0909 NIF del conductor 111111-A Es socio
---

Opción 4:
La aplicación termina, pero antes mostrará los datos de los coches que hayan aparcado en el aparcamiento y ya hayan salido. Por ejemplo, si los coches "AAA – 3333" y "BLK – 8888" hubieran aparcado y al terminar la aplicación ya hubieran salido, se mostraría:

Estos son los coches que han dejado ya el aparcamiento:
Matrícula AAA - 3333 NIF del conductor 111111 Z Es socio
Matrícula BLK - 8888 NIF del conductor 222222 X No es socio

Si no ha salido ningún coche, se muestra el mensaje correspondiente indicándolo.

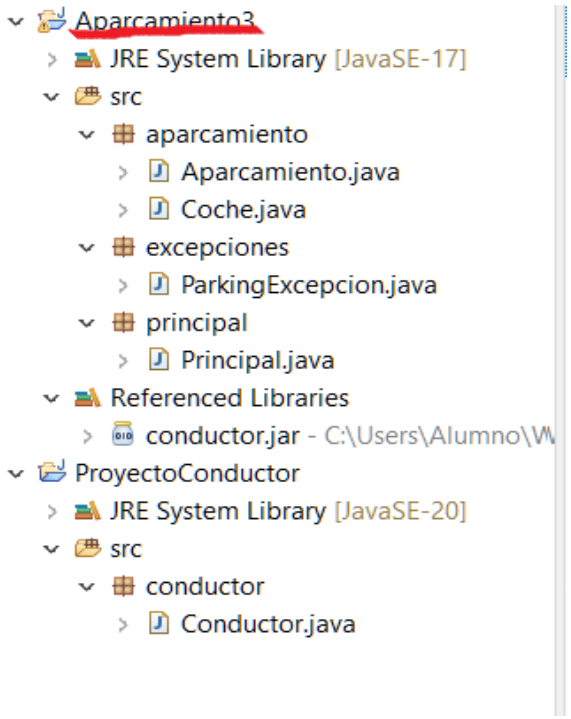
Otras consideraciones:

- Organiza el código de manera que el programa principal no tenga constancia del tipo de colección usada para guardar los datos de los coches aparcados.

- En el aparcamiento guardamos su nombre y un ArrayList con los coches aparcados en un momento dado.

- Crea la clase Conductor en un proyecto aparte y generando el .jar correspondiente lo usas en el proyecto del examen.

- La organización de clases la harás según este esquema (en vez de Aparcamiento3, tu nombre y apellido). Si te resulta más fácil puedes incluir una clase Lector, pero no es obligatorio:



Distribución de la nota por tareas realizadas correctamente (TOTAL 5,1 puntos):

1. Clase Conductor en su proyecto y bien importada en el proyecto principal **(0,5)**
2. Clase Coche (**0,35**)
3. Clase Aparcamiento bien definida y usando correctamente el ArrayList (salvo los puntos 6, 7, 8 y 9, que se puntúan por separado) **(0,5)**
4. Clase Excepción bien declarada y usada (**0,5**)
5. Ajustarse totalmente al esquema de organización pedido **(0,5)**
6. Gestión correcta de la entrada de coche **(1)**
7. Gestión de la salida de coche **(0,75)**
8. Esquema de aparcamiento **(0,5)**
9. Mostrar al salir los coches que han abandonado el aparcamiento **(0,5)**