EJERCICIO 1

Se necesita crear un **programa para la Concejalía de Turismo de Alcalá de Henares con el fin de informar sobre las plazas de aparcamientos públicos de la ciudad.**

La información sobre los **cuatro aparcamientos públicos de Alcalá de Henares y cada tipo de plaza disponible** es la siguiente:

PARKING	PLAZAS PARA COCHES	PLAZAS PARA MOTOS	PLAZAS PARA FURGONETAS	PLAZAS PARA CAMIONES
El Mercado	25	20	0	5
San Lucas	32	15	10	0
Obispado	47	25	15	10
La Paloma	35	18	12	0

1. Al iniciar el programa **debes cargar los datos automáticamente.** Para ello usa el método **cargarDatos()** de la clase Utilidades. Este método devuelve la matriz cargada con los datos de la tabla anterior.
2. Después debes **mostrar un menú** con la siguiente información:

```
run:
**BIENVENIDO AL SISTEMA DE INFORMACIÓN DE PLAZAS DE APARCAMIENTO DE ALCALÁ DE HENARES **
Datos cargados correctamente.

        ------------------------------------------------
        ---- [AYUNTAMIENTO DE ALCALÁ DE HENARES]  ----
        ---  INFORMACIÓN DE PLAZAS DE APARCAMIENTO ---
        ------------------------------------------------
        ----------------- [INFORMES] -----------------
        1. Informe general
        2. Número de plazas totales en Alcalá
        3. Número de plazas para un determinado tipo de vehículo en Alcalá
        4. Número de plazas totales de un parking concreto
        5. Número de plazas de un determinado tipo de vehículo de un parking concreto
        ------------------------------------------------
        6. [INTERCAMBIAR] Intercambiar plazas de aparcamiento de un mismo parking
        7. [ORDENAR] Mostrar número de plazas ordenadas descendentemente

        8. SALIR
        ------------------------------------------------
        Elige una opción:
```

Para ello usa el método **mostrarMenu()** que tienes disponible en la clase de Utilidades.

ESTOS SON LOS MÉTODOS QUE DEBES IMPLEMENTAR OBLIGATORIAMENTE EN LA CLASE Utilidades de tu proyecto:

`public static void mostrarInformeGeneral()`
[opción 1 del menú] Este método debe mostrar toda la información de todos los aparcamientos tal y como la siguiente captura:

```
        ----------------------------------------
        Elige una opción:
1
-------------------------------------------------
Información del parking El Mercado
Número de plazas de <coche>:25
Número de plazas de <moto>:20
Número de plazas de <furgoneta>:0
Número de plazas de <camión>:5
-------------------------------------------------
Información del parking San Lucas
Número de plazas de <coche>:32
Número de plazas de <moto>:15
Número de plazas de <furgoneta>:10
Número de plazas de <camión>:0
-------------------------------------------------
Información del parking Obispado
Número de plazas de <coche>:47
Número de plazas de <moto>:25
Número de plazas de <furgoneta>:15
Número de plazas de <camión>:10
-------------------------------------------------
Información del parking La Paloma
Número de plazas de <coche>:35
Número de plazas de <moto>:18
Número de plazas de <furgoneta>:12
Número de plazas de <camión>:0
```

`public static int obtenerNumPlazasTotal()`

[opción 2 del menú] Este método devuelve el número total de plazas de todos los tipos de vehículos de los cuatro aparcamientos públicos de Alcalá.

```
        ----------------------------------------
        Elige una opción:
2
        El número de plazas totales en Alcalá es: 269
```

`public static int obtenerNumPlazasPorTipoVehiculo(int tipoVehiculo)`

[opción 3 del menú] Este método devuelve el número total de plazas en Alcalá de un tipo de vehículo concreto.

```
        ----------------------------------------
        Elige una opción:
3
        Indica el tipo de vehículo ( 1: coche, 2: moto, 3: furgoneta, 4: camión)
2
        El número de plazas en Alcalá de <moto> es: 78
```

`public static int obtenerNumPlazasPorParking(int numParking)`
[opción 4 del menú] Este método devuelve el número total de plazas de un parking concreto.

```
        ----------------------------------------
        Elige una opción:
4
Indica el número de parking (1: El Mercado, 2: San Lucas, 3: Obispado, 4: La paloma):
3
El número de plazas del parking <Obispado> es: 97
```

public static int obtenerNumPlazasPorParkingPorTipoVehiculo(int numParking, int tipoVehiculo) y de un tipo de vehículo concreto.

```
        ----------------------------------------
        Elige una opción:
5
Indica el tipo de vehículo ( 1: coche, 2: moto, 3: furgoneta, 4: camión):
3
Indica el número de parking (1: El Mercado, 2: San Lucas, 3: Obispado, 4: La paloma):
4
El número de plazas de <furgoneta> del parking <La Paloma> es: 12
```

public static void intercambiarPlazas(int numParking, int tipoVehiculo1, int tipoVehiculo2)

[opción 6 del menú] Este método intercambia el número de plazas de un tipo de vehículo por otro dentro de un mismo parking (intercambia el valor de las celdas)

```
       Elige una opción:
6
Indica el número de parking donde intercambiar plazas (1: El Mercado, 2: San Lucas, 3: Obispado, 4: La paloma):
3
       Indica el primer tipo de vehículo  ( 1: coche, 2: moto, 3: furgoneta, 4: camión)
2
       Indica el segundo tipo de vehículo para intercambiar las plazas ( 1: coche, 2: moto, 3: furgoneta, 4: camión)
3
Se han intercambiado correctamente las plazas de <moto> a <furgoneta> en el parking <Obispado>
```

public static void mostrarNumPlazasPorParkingOrdenadasDesc()

[opción 7 del menú] Este método muestra por consola el número total de plazas por parking de forma descendente.

```
        Elige una opción:
7
Número de plazas por parking ordenadas descendentemente:
 [97]  [65]  [57]  [50]
```

ADEMÁS:

- Damos por hecho que **el usuario va a introducir siempre valores correctos**, tanto del número de parking como el tipo de vehículo.
- Además de los métodos obligatorios **puedes implementar los métodos de utilidades que consideres necesarios**.
- **El menú debe mostrarse hasta elegir la opción de salir**. En ese caso el programa se cerrará. **En el caso de elegir una opción incorrecta** debe avisarse al usuario tal y como puedes ver en la siguiente captura:

```
        Elige una opción:
9
Elige una opción correcta del menú

        ------------------------------------------------
        ---- [AYUNTAMIENTO DE ALCALÁ DE HENARES]  ----
        ---  INFORMACIÓN DE PLAZAS DE APARCAMIENTO ---
        ------------------------------------------------
        ----------------- [INFORMES] -----------------
        1. Informe general
        2. Número de plazas totales en Alcalá
        3. Número de plazas para un determinado tipo de vehículo en Alcalá
        4. Número de plazas totales de un parking concreto
        5. Número de plazas de un determinado tipo de vehículo de un parking concreto
        ------------------------------------------------
        6. [INTERCAMBIAR] Intercambiar plazas de aparcamiento de un mismo parking
        7. [ORDENAR] Mostrar número de plazas ordenadas descendentemente

        8. SALIR
        ------------------------------------------------
        Elige una opción:
8
Se cierra el programa!!!!
```

IMPORTANTE!!!! Por cada opción del menú **debes devolver la información tal y como especifica el enunciado** (observa las capturas)**,** con el nombre del parking (si procede) y el nombre del tipo de vehículo (si procede). **Para ello usa los arrays NOMBRE_PARKINGS y NOMBRE_VEHICULOS** que se proporcionan en la clase principal.

EJERCICIO 2

Se necesita crear una aplicación para **gestionar listas TO-DO de tareas**. Para ello es necesario implementar una clase llamada **Tarea** con las siguientes características:

Todas las tareas tendrán:

- **una descripción:** cadena de texto con la descripción de la tarea.
- **fecha de finalización:** fecha límite de finalización de la tarea. Por ejemplo:
  `LocalDate date = LocalDate.parse("2018-10-30");  //formato YYY-MM-DD`
- **prioridad:** valor entero comprendido entre 1 y 3. Una tarea con valor 1 es más prioritaria que otra con valor 3.
- **finalizada:** un atributo booleano para indicar si ha finalizado o no la tarea.

Se podrán crear tareas de tres formas;

- Indicando obligatoriamente la descripción, la fecha de finalización y la prioridad
- Indicando solo la descripción y la fecha de finalización. En este caso la tarea se inicializará por defecto con la prioridad más baja. (valor constante que no va a cambiar).
- Indicando solo la descripción. En este caso la tarea se inicializará:
  - por defecto con la prioridad más baja
  - tendrá como fecha de finalización la fecha actual (LocalDate.now())

  Usa la función this() para las llamadas entre constructores.

Ten en cuenta también que al crear la tarea:

- su **estado** por defecto será no finalizada.
- si la **descripción supera los 40 caracteres** (valor constante) se cortará el texto hasta ese límite. Por ejemplo si la tarea es:

  "Tengo que estudiar mucho Programación para aprobar" (50 caracteres), la descripción con la que se crea la tarea será *"Tengo que estudiar mucho Programación pa"* (40 caracteres).
- Al crear la tarea se da por hecho que el resto de los valores introducidos son correctos.

Una vez creada la tarea se podrá modificar cualquier dato de la misma menos su descripción, porque si se necesitase modificar la descripción se crearía una nueva tarea.

La clase tendrá la funcionalidad de **codificar** y **descodificar** la descripción. Para ello **implementa los dos métodos de comportamiento proporcionados**, uno para codificar y otro para descodificar, teniendo en cuenta:

- los caracteres a codificar/descodificar vienen especificados en los arrays constantes LETRAS y CODIGOS. Cada letra se corresponde con el código en su misma posición. Sustituye cada letra por su código y al revés según corresponda.
  El método debería de funcionar aunque se modifique el contenido de dichos arrays (añadiendo o quitando letras y códigos).
- se podrá codificar/descodificar la descripción cuando se desee pero **debes controlar que solo se podrá codificar si la descripción no está codificada y solo podrá descodificar si la descripción está codificada.** Devuelve false en caso de error.

Finalmente **sobreescribe el método toString()** para obtener la información de cada Tarea con el siguiente formato:

```
TAREA:
        Descripción: Mi primera tarea
        Prioridad: 3
        Fecha límite de finalización: 2022-12-01
        La tarea no está finalizada
```

Como puedes observar se indica si la tarea está finalizada o no. Usa la clase **StringBuilder**.

CLASE PRINCIPAL EJECUTABLE

Vas a simular una **lista de tareas TODO usando un array de objetos Tarea**.
En el simulacro, dicha lista estará compuesta por **5 tareas** con los siguientes datos:

Tarea 1:
Descripción: Mi primera tarea
Fecha límite: 2022-12-01
Prioridad: 3
Tarea 2:
Descripción: Tengo que estudiar mucho Programación para aprobar
Fecha límite: 2022-12-02
Prioridad: 2
Tarea 3:
Descripción: Tengo que estudiar mucho LM
Fecha límite: 2022-12-03
Tarea 4:
Descripción: En Navidad voy a repasar todos los ejercicios del GitHub
Fecha límite: 2022-12-04
Prioridad: 1
Tarea 5:
Descripción: Mi última tarea

PASO 1: Da de alta los diferentes objetos Tarea en el array en base a la información proporcionada.
PASO 2: Codifica la descripción de la última tarea y haz las pruebas pertinentes según lo especificado. Por ejemplo:

```
**********************************************************************
* Descripción original de la última tarea <Mi última tarea>
* CODIFICADA la última tarea...
* Descripción de la última tarea codificada <M$ últ$m& t&r%&>
* Intentando codificar de nuevo la última tarea...
* [ERROR] No se puede codificar la descripción de la tarea porque ya está codificada
* DESCODIFICADA la última tarea...
* Descripción de la última tarea descodificada <Mi última tarea>
* Intentando descodificar de nuevo la última tarea...
* [ERROR] No se puede descodificar la descripción de la tarea porque no está codificada
**********************************************************************
```

PASO 3: muestra la información de todas las tareas. Crea un método estático en la propia clase principal. Debes obtener las siguientes trazas por consola:

```
**************************************************************
********** INFORMACIÓN DE TODAS LAS TAREAS ******** **********
TAREA:
        Descripción: Mi primera tarea
        Prioridad: 3
        Fecha límite de finalización: 2022-12-01
        La tarea no está finalizada

TAREA:
        Descripción: Tengo que estudiar mucho Programación pa
        Prioridad: 2
        Fecha límite de finalización: 2022-12-02
        La tarea no está finalizada

TAREA:
        Descripción: Tengo que estudiar mucho LM
        Prioridad: 3
        Fecha límite de finalización: 2022-12-03
        La tarea no está finalizada

TAREA:
        Descripción: En Navidad voy a repasar todos los ejerc
        Prioridad: 1
        Fecha límite de finalización: 2022-12-04
        La tarea no está finalizada

TAREA:
        Descripción: Mi última tarea
        Prioridad: 3
        Fecha límite de finalización: 2022-12-06
        La tarea no está finalizada
```

PASO4: finaliza la segunda y tercera tarea y muestra solo las tareas que han finalizado indicando también cuántas han finalizado. Crea un método estático en la propia clase principal. Debes obtener las siguientes trazas por consola:

```
**************************************************************
********** INFORMACIÓN DE SOLO LAS TAREAS FINALIZAS **********
TAREA:
        Descripción: Tengo que estudiar mucho Programación pa
        Prioridad: 2
        Fecha límite de finalización: 2022-12-02
        La tarea está finalizada

TAREA:
        Descripción: Tengo que estudiar mucho LM
        Prioridad: 3
        Fecha límite de finalización: 2022-12-03
        La tarea está finalizada

Número de tareas finalizadas: 2
**************************************************************
```