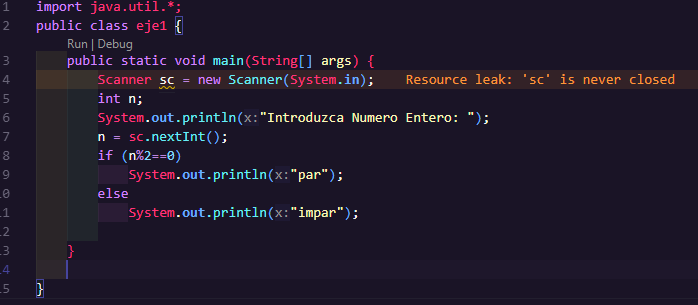
Informe:

EJERCICIO 1. Programa Java que lea un número entero por teclado y calcule si es par o impar.

Podemos saber si un número es par si el resto de dividir el número entre 2 es igual a cero. En caso contrario

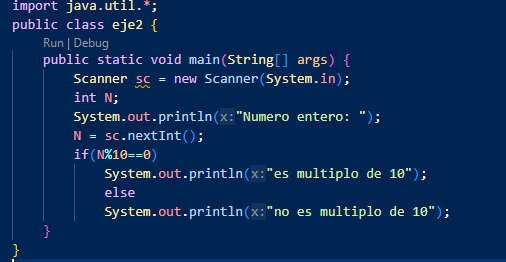
el número es impar



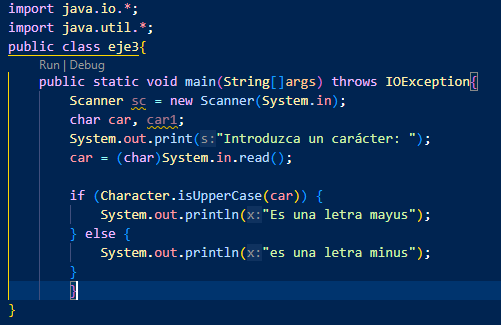
Descripcion del cod:

* Import java.util.\*; esto importa las clases del paquete java.util
* Public static es la entrada del programa el que lo ejecuta
* Scanner sc = new Scanner(System.in); crea el objeto sc tipo Scanner(captura datos del tipo int, double, string y etc.) que permite leer la entrada de datos que registra el usuario, System.in indica que los datos se leerán desde la entrada estándar (el teclado).
* n = sc.nextInt(); se le asigna a n el numero entero que digita el usuario, nextInt() lee el siguiente número de entrada.
* if (n%2==0) utiliza el condicional if para saber si el numero es par o impar, se utiliza el operador mod(%) que obtiene el residuo de la división de n.

EJERCICIO 2. Programa que lea un número entero y muestre si el número es múltiplo de 10.



* Scanner sc = new Scanner(System.in); **Crea un objeto Scanner para entrada**: La clase Scanner se utiliza para capturar la entrada del usuario. El parámetro System.in indica que se capturará desde la consola.
* System.out.println("Numero entero: "); **Muestra un mensaje en la consola**: Imprime el texto "Numero entero: " para que el usuario sepa que debe ingresar un número.
* N = sc.nextInt(); **Captura la entrada del usuario**: Utiliza el método nextInt() del objeto Scanner para leer un número entero ingresado por el usuario y lo almacena en la variable N.
* if (N % 10 == 0) verifica si el número ingresado (N) es múltiplo de 10. Esto se hace calculando el residuo de N entre 10 (N % 10), y si el resultado es 0, significa que N es un múltiplo de 10.

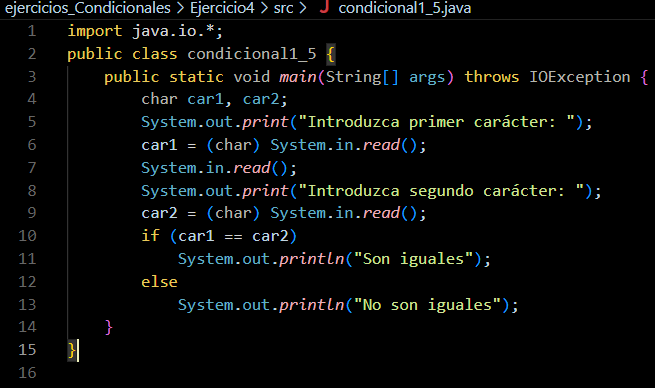
EJERCICIO 3. Programa que lea un carácter por teclado y compruebe si es una letra mayúscula

import java.io.\*;

import java.util.\*;

* **Importa las librerías necesarias**: java.io.\* y java.util.\* son bibliotecas de Java. La primera incluye clases para la entrada/salida como IOException y métodos de lectura como System.in.read(). La segunda incluye utilidades como Scanner.
* public static void main(String[] args) throws IOException { **Método principal main con excepción**: Este es el punto de entrada del programa. El throws IOException indica que el método puede lanzar una excepción si hay un error de entrada/salida, específicamente relacionado con la lectura de datos.
* char car, car1; **Declara dos variables de tipo char**: Se declaran las variables car y car1 de tipo carácter.
* car = (char)System.in.read(); **Lee el carácter ingresado por el usuario**: Utiliza System.in.read() para leer el valor del carácter. Como System.in.read() devuelve un entero, se hace un casting explícito a char para convertirlo a un carácter.
* if (Character.isUpperCase(car)) { **Condicional para verificar si el carácter es mayúscula**: Usa el método estático isUpperCase de la clase Character para verificar si el carácter ingresado es una letra mayúscula. Si lo es, ejecuta el bloque if.

**EJERCICIO4**. Programa que lea dos caracteres y compruebe si son iguales.

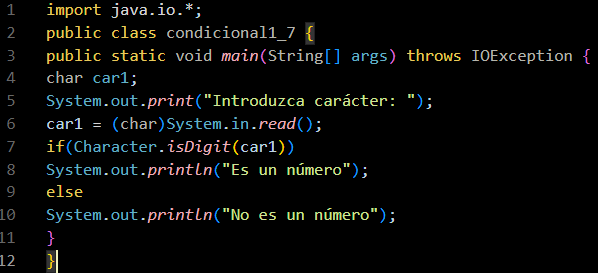


* **Imports**: Se importa la clase java.io.\* para poder utilizar la funcionalidad de entrada y salida de datos, necesaria para leer los caracteres ingresados.
* El programa se ejecuta dentro de la clase **condicional1\_5**, y la función **main** es el punto de entrada del programa.
* Se declara que el método **main** puede lanzar una excepción de entrada/salida **(IOException)** debido al uso de la función **System.in.read().**
* Se declaran dos variables de tipo **char, car1 y car2,** para almacenar los caracteres que el usuario ingresará.
* El programa solicita al usuario que ingrese el primer carácter, lo lee usando **System.in.read(),** y lo guarda en **car1.**
* Luego se llama a **System.in.read()** de nuevo para descartar el carácter de "intro" (Enter) que se queda en el búfer después de la primera lectura.
* El programa repite el proceso para el segundo carácter, guardándolo en car2.

**Condición:**

El programa compara si **car1** es igual a **car2** usando el operador ==. Si son iguales, imprime "Son iguales", y si no, imprime "No son iguales".

**EJERCICIO5**. Una forma alternativa de hacer este programa es creando dos objetos Character a partir de los caracteres que se han leído y compararlos utilizando el método equals.

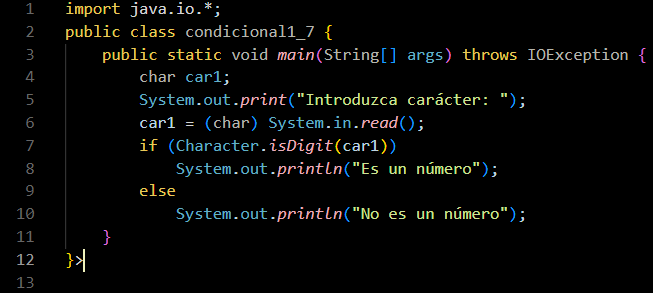


Este código Java es una versión modificada del anterior, que introduce el uso de la clase Character para comparar los caracteres en lugar de hacerlo directamente con el operador ==. Aquí están los cambios y su descripción

**Diferencias:**

* **Uso de Character en lugar de char:** Ahora los caracteres se envuelven en objetos de la clase Character, lo que permite utilizar el método equals() para realizar la comparación.
* **Método equals():** Se utiliza el método equals() en lugar de == para comparar los dos objetos Character, que es la forma correcta de comparar objetos en Java.

**Ejercicio6.** Programa que lea dos números por teclado y muestre el resultado de la división del primer número por el segundo. Se debe comprobar que el divisor no puede ser cero.



* Se importa java.io.\* para permitir la entrada de datos desde el teclado.
* El método principal main puede lanzar una excepción **IOException** debido al uso de **System.in.read().**
* Se declara una variable de tipo **char** llamada car1 para almacenar el carácter ingresado por el usuario.
* El programa muestra un mensaje pidiendo al usuario que introduzca un carácter. Después de que el usuario lo hace, el carácter se lee y se guarda en car1 utilizando **System.in.read().**
* El programa verifica si el carácter ingresado es un número utilizando el método estático **Character.isDigit(car1**). Este método devuelve true si el carácter es un dígito (del 0 al 9).
* Si la condición es verdadera, imprime "Es un número".
* Si la condición es falsa, imprime "No es un número".