EJERCICIO 1

Crear un programa que sustituya el siguiente switch-case por un HashMap de manera que el código sea equivalente:

```
switch(opcion.toLowerCase()){
  case "fruta":
      res="Manzana, platano o pera";
    break;
  case "verdura":
      res="Tomate, lechuga o zanahoria";
    break;
  case "carne":
      res="Cerdo, ternera o pollo";
    break;
  case "pescado":
      res="Sardinas, caballa ó salmonete";
    break;
  default:
      res="No tenemos de ese tipo de comida";
}
```

opcion se coge del teclado y res es una variable de tipo String

EJERCICIO 2

Crear la clase Joyería que contiene un nombre y un **HashMap** de objetos de la clase Joya donde la clave es el nombre de la Joya. Recordando los elementos que tiene la clase Joya:

- Orden de creación (numero incrementado en cada nuevo objeto)
- Nombre.
- Tipo (por ejemplo collar, anillo, brazalete, diadema, tiara, medallón, pendientes o más cosas) como String.
- Precio.
- Peso.
- Material (solo se permite oro, plata, platino, rodio e iridio) como un char.

La clase **Joyeria** debe tener los siguientes métodos:

- Constructor que inicializa la joyería y el **HashMap** a vacío.
- Método **encontrarJoya** privado que dado el nombre de una joya nos devuelve el objeto Joya o null dependiendo si existe o no ese nombre de joya en la colección.
- Método **añadirJoya** que recibe los datos de una joya y la añade a la colección, comprobando previamente que el nombre no está ya en la lista.
- Modificar el método **toString** de la clase Joya para usar **HashMap** final static para el material.
- Método **toString** que muestre el nombre de la Joyeria y posteriormente los datos de cada objeto Joya **(values)** que contenga el **HashMap**. Si no hay ninguna devolver “SIN JOYAS”.
- Método **buscarJoya** que dado un String devuelva un String con los datos de las Joyas cuyo nombre coincida total o en parte con el String de entrada. Si no encuentra ninguna coincidencia devolver “SIN RESULTADOS“.

- Método **verPorMaterial** que recibe como entrada un material válido y devuelve un String con los datos de todas la Joyas que estén hechas de dicho material.
- Método **borrarJoya** que a partir del nombre de una joya y la borra de la lista, si no existe la joya mostrar “NO EXISTE LA JOYA A BORRAR”.
- Método **sumarPrecio** que devuelve la suma de todas los joyas de collares y anillos, si hay.
- Método **subirPrecio** que tiene como parámetro un porcentaje y les aumenta a todas las joyas de la tienda el precio en dicho porcentaje.
- Método **mayorPrecio** que devuelve un String con los datos de la joya de mayor valor de toda la tienda. Si no hay joyas en la tienda, devolver “NO HAY JOYAS”.
- Método **menorMaterial** que devuelve un String con los datos de la joya con el menor precio de un material que se le pasa como parámetro. Si no hay de dicha categoría devolver “NO HAY JOYAS DE ESE MATERIAL”.
- Método **borrarBaratas** que borra del ArrayList las joyas cuyo precio no supere los 20€.
- Método **contadorMateriales** que devuelve un **String** con el número de joyas de cada material. Si no hay ninguna joya mostrar “NO HAY JOYAS”.
- Método **tipoMasValiosa** que devuelve un **String** con el nombre del tipo de joya, que suma más dinero en precios de sus joyas. Si no hay ninguna joya mostrar “NO HAY JOYAS”.

EJERCICIO 3

Queremos mantener una colección de los libros que hemos ido leyendo, poniéndoles una calificación según nos haya gustado más o menos al leerlo. La implementación de la clase Libro la tenéis en el proyecto ejercicio 3.

Para ello, tenemos la clase libro con los siguientes atributos:

- Título
- ISBN.
- Autor.
- Número de páginas.
- Calificación que le damos **entre 0 y 5 estrellas**.

También dispone de los siguientes métodos

- Constructor con todos los atributos.
- Constructor de copias.
- Métodos selectores y modificadores.
- Método **toString** que devuelve un String.

Crear una clase **ColeccionLibros**, que almacena un conjunto de libros (con un **HashMap**) donde la clave es el ISBN. Crear también los siguientes métodos:

- Constructor sin parámetros que cree el **HashMap** por defecto vacío.
- Método que reciba como parámetros todos los datos de un libro para crear un objeto de tipo libro y lo añada al **HashMap**. Comprobando previamente que no lo teníamos ya (no existe el ISBN en la colección) y mostrando un mensaje informativo.
- Método **toString()** que devuelve incluye la información de todos los libros. Si no hay libros devolver **"NO HAY LIBROS"**
- Método que dado el título de un libro lo borre de la lista y en caso de que no exista muestre **"NO EXISTE EL LIBRO".**
- Método que nos devuelva un String con la información de los libros con entre **3 y 5 estrellas**.
- Método que nos devuelva un String con la información de los libros que tengan menos de un número de páginas que se le pasa como parámetro.
- Método que devuelva los datos del objeto Libro con mayor número de páginas. Si no hay libros devolver **"NO HAY LIBROS"**
- Método que devuelva el objeto Libro con mayor número de páginas de un determinado autor que se pasa como parámetro. Si no existe el autor devolver **"NO HAY LIBROS DEL AUTOR"**.
- Crear un método que borre todos los libros que tengan una calificación entre 0 y 2 estrellas.
- Crear un método que devuelve un **String** con cada autor y el número medio de estrellas (double) que tienen sus libros. Si no hay libros devolver **"NO HAY LIBROS".**
- Método que devuelva el autor que más páginas escribe. Si no hay libros devolver **"NO HAY LIBROS"**

EJERCICIO 4

Crear la clase **OrganizadorEventos** que contiene un HashMap de objetos de la clase **Eventos** donde la clave será el nombre del evento. Todos los métodos que devuelvan un String, devolverán un mensaje especial en caso de que la colección estuviera vacía al invocar al método. La clase debe tener los siguientes métodos:

- Constructor sin parámetros que cree el **HashMap** por defecto vacío.
- **añadirEvento** que dado los datos de un evento añadirlo a la colección comprobando que no existe el nombre previamente. **Modularizar** la búsqueda mediante un método privado.
- Modificar el método **toString** de la clase Evento para usar **HashMap** para el tipo de evento.
- **toString()** con los datos de todos los eventos que haya en la lista en ese momento.
- Un método que permita **buscar eventos,** que dado un String devuelva otro String con la información de todos los eventos cuyo nombre coincide con el String dado como parámetro.
- Un método que dado el nombre de un evento y un número de entradas busque el evento e intente comprar entradas. En caso de que no encuentre el evento informará de ello al usuario.

- En caso de que exista el comportamiento será el que tenga la clase evento (caso en el que no hay suficientes entradas para vender).
- Un método que **borre un evento** por nombre.
- Método que devuelva un String con los nombres de los eventos que tienen aforo completo.
- Método que devuelva cuantos eventos hay en la lista que no han vendido nada del aforo.
- Método que devuelva un String con la información del Evento con más entradas vendidas.
- Borrar todos los eventos cuyo número entradas vendidas sea menos del 50% del total.
- Método que devuelva un String con los días que haya eventos (día/mes/año) y para cada día la recaudación total de todos los eventos programados para ese día (por cada uno de los eventos sumar entradas vendidas x precio entrada)
- Método que devuelva un String con el día (día/mes/año) que tiene más eventos programados