Ejercicio 1

Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
resultado = 10/0
```

Ejercicio 2

Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
lista = [1, 2, 3, 4, 5]
lista[10]
```

Ejercicio 3

Realiza una función llamada agregar_una_vez(lista, el) que reciba una lista y un elemento. La función debe añadir el elemento al final de la lista con la condición de no repetir ningún elemento. Además, si este elemento ya se encuentra en la lista se debe lanzar un error de tipo ValorDuplicado, excepción propia que debes capturar y mostrar en la función principal.

Ejercicio 4

Crear un programa que almacene la información de un edificio

- Dirección
- Municipio

- Código postal (entero)
- Lista de apartamentos
    - Número de planta (entero)
    - Puerta
    - Lista de Propietarios
        - Nombre
        - Apellidos

Crea en el main los métodos para pedir al usuario la información del edificio. En caso de que el usuario meta alguna información mal, se debe capturar la excepción (try y catch) y volver a pedir el dato.

Métodos Auxiliares

- getApartmento(int piso, String puerta): devuelve el apartamento en esa planta y esa puerta. Si no encuentra el apartamento, lanza la excepción ApartmentoNotFoundException
- getPropietarios(int piso, String puerta): devuelve los propietarios del apartamento en esa planta y puerta. Si no encuentra el apartamento, lanza la excepción ApartmentoNotFoundException

Ejercicio 5

Crea un programa que permita representar los datos de una empresa:

- Nombre de la empresa
- CIF
- Departamentos. Por cada departamento: nombre y listado de empleados
    - Por cada empleado: nif, nombre, apellidos y puesto

Crea los siguientes métodos en Compañía:

- Devuelve los empleados de un departamento dado. En caso de que el departamento no exista lanza la excepción DepartmentoNotFoundException.
- Dado un nombre de departamento, devuelve el Departamento con ese nombre. En caso de que el departamento no exista lanza la excepción DepartamentoNotFoundException
- Devuelve los datos de un empleado a partir de un NIF. E En caso de que el empleado no exista lanza la excepción EmpleadoNoFoundException

Ejercicio 6

Crea un programa que permita representar los datos de una empresa:

- Nombre de la empresa
- CIF
- Departamentos. Por cada departamento: nombre y listado de empleados
    - Por cada empleado: nif, nombre, apellidos y puesto

Crea los siguientes métodos en Compañia:

- Devuelve los empleados de un departamento dado. En caso de que el departamento no exista lanza la excepción DepartamentoNotFoundException.
- Dado un nombre de departamento, devuelve el Departamento ese nombre. En caso de que el departamento no exista lanza la excepción DepartmentoNotFoundException
- Devuelve los datos de un empleado a partir de un NIF. E En caso de que el empleado no exista lanza la excepción EmpleadoNoFoundException.

Ejercicio 7

Crea un programa que maneje los datos de una biblioteca:

- Nombre de la biblioteca
- Libros del catálogo. Por cada libro se guardará: ISBN, título, autor, géneros (listado)
- Socios de la biblioteca. Por cada socio se guardará: nif, nombre, apellidos, número de socio, código postal.
- Se almacenará un historial de préstamos: ISBN, fecha préstamos, nif (del socio que lo ha tomado prestado), fecha de devolución

Implementar los métodos en Biblioteca:

- Un método que dado un ISBN, devuelve el libro. Si no existe el libro se lanza la excepción LibroNotFoundException(isbn)
- Dado un nif, devuelve el socio. Si no existe SocioNotFoundException(nif)
- Comprueba que, dado un nif y un isbn, el socio ha tomado prestado un libro. Si no existe el socio SocioNotFoundException(nif) y si no existe el libro LibroNotFoundException(isbn)