IES ILIBERIS
Avda. Andalucía
Atarfe - Granada

RELACIÓN DE EJERCICIOS 13 - Excepciones

Resolver los siguientes problemas escribiendo el algoritmo con lenguaje Java.

Crear en GitHub un repositorio llamado **UD7-Excepciones** para subir cada uno de los ejercicios de la relación.

1. Observa el siguiente fragmento de código. Escríbelo en el editor de código y ejecuta el programa. Analiza el resultado y la excepción que se produce.

```
String [] arrayString = new String[25];

System.out.println (arrayString[3].length());
```

2. Igual que el anterior. Observa el siguiente fragmento de código y analiza el resultado.

```
String aux = "hola";

int aux2 = Integer.parseInt(aux);
```

3. Escribe un método auxiliar de nombre **caracterEn** en Java que realice la siguiente acción:
   - Recibe como parámetros una cadena (String) y un entero (int);
   - Si el entero está entre 0 y la longitud de la cadena (puedes hacer uso del método "length():int" de la clase "String") devuelve el carácter en la posición correspondiente (puedes hacer uso del método "charAt(int)" de la clase "String").
   - En caso contrario, construye y lanza una excepción de tipo **Exception**.

4. Construye un programa "main" en Java que realice las siguientes acciones:
   - Crea un objeto de la clase **Scanner** y lo asocia con la entrada estándar (la consola de MSDOS)
   - Lee un objeto de tipo **String** de la consola en un objeto **lecTeclado**
   - Invoca al método **caracterEn** definido en el ejercicio anterior, con la **String** leída de la entrada estándar y el entero **7**, mostrándolo por pantalla
   - Captura la posible excepción, mostrando por pantalla un mensaje: "Has intentado recuperar una posición de la cadena de caracteres que no existe".

5. Define una clase **NumeroNegativoExcepcion** que herede de **Exception** y que contenga un constructor sin parámetros y un constructor que reciba como parámetro una **String**, de tal modo que ambos invoquen a los constructores de la clase **Exception** correspondientes.

6. Analiza el resultado de ejecutar el siguiente fragmento de código y corrige lo que creas necesario

```
public static double accesoPorIndice(double[] v, int j) throws RuntimeException{
     try{
         if ((0 <= j) && (j <= v.length - 1)){
             return v[j];
         }
         else {
             throw new RuntimeException("El índice " + j + " no existe en el array.");
         }
```

RELACIÓN DE EJERCICIOS 13 - Excepciones

```
        }
        catch (RuntimeException exc){
            throw exc;
        }
    }
```

desde el siguiente main:

```
public static void main(String [] args){
        double[] v = new double[15];
        System.out.println(accesoPorIndice(v, 16));
    }
```

7. Analiza el resultado de ejecutar el siguiente fragmento de código y corrige lo que creas necesario

```
public static double accesoPorIndice(double[] v, int j) throws RuntimeException{
        try{
            if ((0 <= j) && (j <= v.length - 1)){
                return v[j];
            }
            else {
                throw new Exception("El índice " + j + " no existe en el array.");
            }
        }
        catch (RuntimeException exc){
            throw exc;
        }
    }
```

desde el siguiente main:

```
public static void main(String [] args){
        double[] v = new double[15];
        System.out.println(accesoPorIndice(v, 16));
    }
```

8. Analiza el resultado de ejecutar el siguiente fragmento de código y corrige lo que creas necesario

```
public static double accesoPorIndice(double[] v, int j) throws Exception{
        try{
            if ((0 <= j) && (j <= v.length - 1)){
                return v[j];
            }
            else {
                throw new Exception("El índice " + j + " no existe en el array.");
            }
        }
        catch (Exception exc){
            throw exc;
        }
    }
```

desde el siguiente main:

```
public static void main(String [] args){
     double[] v = new double[15];
     System.out.println(accesoPorIndice(v, 16));
 }
```

9. Analiza el resultado de ejecutar el siguiente fragmento de código y corrige lo que creas necesario

```
public static double accesoPorIndice(double[] v, int j) throws Exception{
     try{
         if ((0 <= j) && (j <= v.length - 1)){
             return v[j];
         }
         else {
             throw new RuntimeException("El índice " + j + " no existe en el array.");
         }
     }
     catch (RuntimeException exc){
         throw exc;
     }
 }
```

desde el siguiente main:

```
public static void main(String [] args){
     double[] v = new double[15];
     System.out.println(accesoPorIndice(v, 16));
 }
```

10. Analiza el resultado de ejecutar el siguiente fragmento de código. Modifica el parámetro pasado a los **parseInt** para provocar excepciones y observa cómo cambia el valor devuelto

```
public class Relacion13_ejercicio10 {

    public static void main(String[] args) {

        try {

            System.out.println(metodo());

        } catch (Exception e) {

            System.out.println("Excepción en método() ");

            e.printStackTrace();

        }

    }


    public static int metodo(){
```

RELACIÓN DE EJERCICIOS 13 - Excepciones

```
        int valor=0;
        try {
            valor = valor + 1;
            valor = valor + Integer.parseInt("42");
            valor = valor + 1;
            System.out.println("Valor al final del try: " + valor);
        } catch (NumberFormatException e) {
            valor = valor + Integer.parseInt("42");
            System.out.println("Valor al final del catch: " + valor);
        } finally {
            valor = valor + 1;
            System.out.println("Valor al final del finally: " + valor);
        }
        valor = valor + 1;
        System.out.println("Valor antes del return: " + valor);
        return valor;
    }
}
```

11. Analiza el resultado de ejecutar el siguiente fragmento de código. Modifica el parámetro pasado a los **parseInt** para provocar excepciones y observa cómo cambia el valor devuelto

```
public class Relacion13_ejercicio11 {
    public static void main(String[] args) {
        try {
            System.out.println(metodo());
        } catch (Exception e) {
            System.out.println("Excepción en método() ");
            e.printStackTrace();
        }
    }

    public static int metodo()throws NumberFormatException{
        int valor=0;
```

RELACIÓN DE EJERCICIOS 13 - Excepciones

```
            try {
                valor = valor + 1;
                valor = valor + Integer.parseInt("W");
                valor = valor + 1;
                System.out.println("Valor al final del try: " + valor);
            } catch (NumberFormatException e) {
                valor = valor + Integer.parseInt("42");
                System.out.println("Valor al final del catch: " + valor);
                throw e;
            } finally {
                valor = valor + 1;
                System.out.println("Valor al final del finally: " + valor);
            }
            valor = valor + 1;
            System.out.println("Valor antes del return: " + valor);
            return valor;
        }
    }
```

12. Realiza un programa que pida 6 números por teclado y nos diga cuál es el máximo. Si el usuario introduce un dato erróneo (algo que no sea un número entero) el programa debe indicarlo y debe pedir de nuevo el número