Excepciones

Programación I - UNGS

Excepciones

- Las excepciones permiten representar errores en tiempo de ejecución.

Excepciones

- Las excepciones permiten representar errores en tiempo de ejecución.
- Ejemplos de tipos de excepciones:

Excepciones

- Las excepciones permiten representar errores en tiempo de ejecución.
- Ejemplos de tipos de excepciones:
  1. **NullPointerException**: Cuando accedemos a un miembro o llamamos a un método de un objeto **null**.

Excepciones

- Las excepciones permiten representar errores en tiempo de ejecución.
- Ejemplos de tipos de excepciones:
  1. **NullPointerException**: Cuando accedemos a un miembro o llamamos a un método de un objeto **null**.
  2. **ArrayIndexOutOfBoundsException**: Cuando accedemos a un índice no válido de un arreglo.

Excepciones

- Las excepciones permiten representar errores en tiempo de ejecución.
- Ejemplos de tipos de excepciones:
  1. **NullPointerException**: Cuando accedemos a un miembro o llamamos a un método de un objeto **null**.
  2. **ArrayIndexOutOfBoundsException**: Cuando accedemos a un índice no válido de un arreglo.
  3. **NumberFormatException**: Cuando leemos de la consola algo que no es un número y lo intentamos convertir a int.

```
1   String s = "5d";
2   int x = Integer.parseInt(s);
```

Captura de excepciones

- Cuando se producía una excepción, hasta ahora se terminaba el programa.

Captura de excepciones

- Cuando se producía una excepción, hasta ahora se terminaba el programa.
- Los errores o "situaciones de excepción" son problemas comunes con los que deben lidiar los programas, y no es una buena medida que el programa termine abruptamente cuando se produce un error.

Captura de excepciones

- Cuando se producía una excepción, hasta ahora se terminaba el programa.
- Los errores o "situaciones de excepción" son problemas comunes con los que deben lidiar los programas, y no es una buena medida que el programa termine abruptamente cuando se produce un error.
- Se necesita un mecanismo para capturar excepciones y recuperarse del error, o bien, mostrar un mensaje amigable al usuario.

try ... catch

- La captura de excepciones se logra con la estructura **try ... catch**:

try ... catch

- La captura de excepciones se logra con la estructura **try ... catch**:

```
1  try
2  {
3     // Codigo normal
4  }
5  catch(TipoDeException nombreVariable)
6  {
7     // Codigo para manejar el caso de excepcion
8  }
```

Ejemplo

```
1  int[] primos = new int[]{2,3,5,7,11,13,17,19,23,29};
2  try
3  {
4    int i = Machete.pedirEntero("Ingrese el numero de primo " +
5      "que quiere consultar (1-10):");
6
7    System.out.println("El primo numero " + i + " es el " + primos[i-1]);
8  }
9  catch(ArrayIndexOutOfBoundsException ex)
10 {
11   System.out.println("El numero que ingresó no está entre 1 y 10");
12 }
```

Ejemplo: Recuperarnos del error

```
1  int[] primos = new int[]{2,3,5,7,11,13,17,19,23,29};
2  boolean listo=false;
3  while (!listo)
4  {
5    try
6    {
7      int i = Machete.pedirEntero("Ingrese el numero a consultar (1-10):");
8      System.out.println("El primo numero " + i + " es el " + primos[i-1]);
9      listo = true;
10   }
11   catch(ArrayIndexOutOfBoundsException ex)
12   {
13     System.out.println("El numero que ingresó no está entre 1 y 10");
14   }
15 }
```

Ejemplo: Capturar varios tipos de error

```
1  int[] primos = new int[]{2,3,5,7,11,13,17,19,23,29};
2  boolean listo=false;
3  while (!listo)
4     {
5     try
6     {
7        String stri = Machete.pedirTexto("Ingrese el numero a consultar (1-10):");
8        int i = Integer.parseInt(stri);
9        System.out.println("El primo numero " + i + " es el " + primos[i-1]);
10       listo = true;
11    }
12    catch(Exception ex)
13    {
14       System.out.println("Se produjo un error: " + ex.getMessage());
15    }
16 }
```

Lanzado de excepciones

- Las excepciones se lanzan utilizando la instrucción **throw**.

Lanzado de excepciones

- Las excepciones se lanzan utilizando la instrucción **throw**.
- Una excepción es un objeto de una clase que representa una situación anormal.

Lanzado de excepciones

- Las excepciones se lanzan utilizando la instrucción **throw**.
- Una excepción es un objeto de una clase que representa una situación anormal.
- Para lanzar una excepción, hace falta crear un objeto del tipo de excepción que se quiere lanzar:

```
1 throw new TipoException(<parametros del constructor>);
```

Lanzado de excepciones

- Las excepciones se lanzan utilizando la instrucción **throw**.
- Una excepción es un objeto de una clase que representa una situación anormal.
- Para lanzar una excepción, hace falta crear un objeto del tipo de excepción que se quiere lanzar:

```
1 throw new TipoException(<parametros del constructor>);
```

- Cuando se lanza una excepción se corta la ejecución del método y se busca si se produjo en el contexto de un try ... catch que capture ese tipo.

Lanzado de excepciones

- Las excepciones se lanzan utilizando la instrucción **throw**.
- Una excepción es un objeto de una clase que representa una situación anormal.
- Para lanzar una excepción, hace falta crear un objeto del tipo de excepción que se quiere lanzar:

```
1 throw new TipoException(<parametros del constructor>);
```

- Cuando se lanza una excepción se corta la ejecución del método y se busca si se produjo en el contexto de un try ... catch que capture ese tipo.
- En caso de que no sea capturada la excepción, se termina la aplicación con un mensaje de error.

Ejemplo

```
1  public static int max(int[] arr)
2  {
3    if(arr.length == 0)
4      throw new RuntimeException("El arreglo no tiene elementos");
5
6    int m = arr[0];
7
8    for(int i=1; i<arr.length; i++) if (arr[i]>m)
9        m=arr[i];
10
11   return m;
12 }
```

Ejemplo

- Su uso sería:

Ejemplo

- Su uso sería:
-

```
1  int[] arr = new int[]{−12,−5,−7,−6,−8};
2  try {
3     System.out.println("El máximo del arreglo es:");
4     int m = max(arr);
5     System.out.println(m);
6  } catch(Exception ex) {
7     System.out.print("Se produjo un error: " + ex.getMessage());
8  }
```

Ejemplo

- Su uso sería:
-

```
1  int[] arr = new int[]{−12,−5,−7,−6,−8};
2  try {
3     System.out.println("El máximo del arreglo es:");
4     int m = max(arr);
5     System.out.println(m);
6  } catch(Exception ex) {
7     System.out.print("Se produjo un error: " + ex.getMessage());
8  }
```

- Este ejemplo termina correctamente e imprime:

  El máximo del arreglo es:
  -5

Ejemplo

- Probemos con un arreglo vacío:

Ejemplo

- Probemos con un arreglo vacío:
-

```
1  int[] arr = new int[]{};
2  try {
3     System.out.println("El máximo del arreglo es:");
4     int m = max(arr);
5     System.out.println(m);
6  } catch(Exception ex) {
7     System.out.print("Se produjo un error: " + ex.getMessage());
8  }
```

Ejemplo

- Probemos con un arreglo vacío:
-

```
1  int[] arr = new int[]{};
2  try {
3     System.out.println("El máximo del arreglo es:");
4     int m = max(arr);
5     System.out.println(m);
6  } catch(Exception ex) {
7     System.out.print("Se produjo un error: " + ex.getMessage());
8  }
```

- Aquí se produce una excepción en la función max que interrumpe el flujo de ejecución y ejecuta el código del catch:

  El máximo del arreglo es:
  Se produjo un error: El arreglo no tiene elementos