Programación Interactiva
Manejo de Excepciones

Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle

¿Qué es una excepción?

- Las excepciones son las formas en que los programas de Java manejan los diferentes errores que pueden ocurrir. Puede pensarse en una excepción como en un “error”.

- Cuando una excepción ocurre decimos que fue “lanzada”, y cuando manejamos dicha excepción, es decir hacemos algo al respecto del error, decimos que fue “capturada”

- Ejemplos de excepciones son:
  - Tratar de convertir la cadena “123A4” a número
  - Tratar de llamar a un método no estático de una referencia *null*

¿Qué es una excepción?

- Para capturar excepciones en un bloque de código susceptible debemos usar usar un bloque *try..catch*, que radica en un bloque de instrucciones en el que se capturarán las excepciones (bloque *try*), y uno o más bloques de manejo de excepciones (bloques *catch*)

- Por ejemplo:

```
try {
    // sentencias a monitorear el error
}
catch (tipoexcepcion nombrevar) {
    // sentencias de manejo de la excepción
}
finally {
    //sentencias a ejecutar ocurran o no excepciones
}
```

Una o más

Opcional

¿Qué es una excepción?

- Por ejemplo:

```
try {
        int i = Integer.parseInt(“123A4”);
} catch (NumberFormatException nfe) {
        System.out.println(“El formato del número es erroneo”);
}
```

Ejemplo práctico

```
try {
    String input = JOptionPane.showInputDialog("Digite un "+
        "número:");
    int i = Integer.parseInt(input);
    System.out.println("El número es "+i);
} catch (NumberFormatException nfe) {
    System.out.println("El formato del número es erroneo");
} catch (NullPointerException npe) {
    System.out.println("Usted no ha digitado ningún número");
}
```

- Este código pediría al usuario un número, en el caso de que el número sea válido, el programa imprime el primer mensaje.
- Si el número es inválido, el segundo mensaje es mostrado.
- Y si el usuario cierra el cuadro de dialogo (caso en el cual showInputDialog retorna null), entonces el tercer mensaje sería mostrado.

Excepciones no tratadas

- Cuando en un programa se arroja una excepción y esta no es capturada, la excpeción supera los límites del programa y es capturada por la JVM, mostrando un mensaje parecido a este:

```
Exception in thread "main“
    java.lang.NullPointerException
        at MiClase.main(MiClase.java:17)
```

¿Cómo arrojar una excepción?

- En ocasiones no solo debemos capturar excepciones predefinidas, sino que debemos crear nuestras propias excepciones y arrojarlas.
- Para arrojar una excepción debe usarse la palabra reservada ***throw***, que funciona se usa así:

```
…
  if (elNumeroNoMeGustó)
          throw new NumberFormatException()
…
```

- En el ejemplo estamos arrojamos una de las excepciones predefinidas que es usada para informar de errores en el formato de conversiones de String a número.

Tipos de excepciones

- Pueden distinguirse dos tipos de excepciones:
  - ***Runtime Exceptions:*** Son excepciones que se producen en el sistema de ejecución de Java. Tal como usar referencias null, hacer una dividisión entre cero, acceder a un elemento inexistente en un array.
  - ***NonRuntime Exceptions:*** Son excepciones que se producen fuera del sistema de ejecución de Java. Son ejemplo de estas las excepciones que se producen por acceso a archivos (IOExceptions)

- En el segundo tipo de excepciones el compilador se asegura de que el programador maneje la excepción (es decir, que cree un bloque *try…catch*)

Tipos de excepciones

- La manera de distinguir ambos tipos de excepciones es mediante la clase de las que estas extienden (si, todas las excepciones son clases).

- Las excepciones del tipo Runtime deben extender de la clase *RuntimeException*, mientras Isas de tiopo NonRuntime deben extender de *Exception.*

Excepciones comunes

- ***IOException Generalmente fallas de entrada o salida*, tal como la inhabilidad de leer desde un archivo.**
- ***NullPointerException:* Referencia a un objeto NULL**
- ***NumberFormatException:* Una conversión fallida entre Strings y números**
- ***OutOfMemoryException:* Muy poca memoria para instanciar un objeto nuevo (new)**

Excepciones comunes

- ***SecurityException:*** Un applet tratando de realizar una acción no permitida por la configuración de seguridad del browser

- ***StackOverflowException:*** El sistema corriendo fuera de espacio en Stack (stack space)

Errores

- Además de las excepciones, en Java existen los *Errores*, clases parecidas a las excepciones, pero su objetivo es informar de una situación anormal grave, algo así como situaciones que nunca debieron ocurrir.

- Son ejemplos de errores las clases:
  - *ThreadDeath*
  - *VirtualMachineError*
  - *AssertionError*

- Debido a que los Errores son “inesperados”, el compilador no exige al compilador que los capture.

- Todos los Errores deben extender de la clase ***Error***

Excepciones NonRuntime

- Si tratáramos de compilar una clase que tuviera el siguiente método:

```
public boolean abreArchivo() {
    new FileInputStream("archivo.txt");
    return true;
}
```

- El compilador nos daría este error

```
unreported exception java.io.FileNotFoundException;
must be caught or declared to be thrown
```

- Debido a que no estamos capturando la excepción FileNotFoundException que puede ser lanzada

Excepciones NonRuntime

- Para librarnos del problema tenemos dos opciones:
  - Capturar la excepción:

```
public boolean abreArchivo() {
    try {
        new FileInputStream("archivo.txt");
        return true;
    } catch (IOException ioe) {
        return false;
    }
}
```

- O indicar en la declaración del método, que la excepción puede ser lanzada:

```
public boolean abreArchivo() throws IOException {
    new FileInputStream("archivo.txt");
    return true;
}
```

Arrojando Excepciones

- Para arrojar una excepción debemos usar la palabra ***throw*** seguida de un objeto del tipo Excepción (o Error)

- Esto es útil cuando queremos informar al programa invocador que ocurrió una situación anómala en nuestro código

- Por ejemplo:

```
public boolean debitarCuenta (int valor)
     throws InvalidAmountException {
     …
     if (balance+sobregiro+sobrecanje < valor)
          throw new InvalidAmountException()
     …
}
```

Creación de Excepciones

- Podemos crear una excepción tan solo creando una clase que extienda de otra excepción ya existente:

```
public class InvalidAmountException extends RuntimeException {}

public class InvalidAmountException extends Exception {}
```

- Y como es una clase, podemos colocar cualquier método y atributo que se nos ocurra:

```
public class InvalidAmountException extends RuntimeException
{
    protected int valorInvalido;
    public InvalidAmountException(int valorInvalido) {
        this.valorInvalido = valorInvalido;
    }
    public int getValorInvalido() {
        return valorInvalido;
    }
}
```

Excepciones comunes

- ***ArithmeticException:*** Errores Matemáticos, como división por cero.
- ***ArrayIndexOutOfBoundsException:*** Un programa tratando de almacenar, el tipo de índice erróneo, de datos en un arreglo.
- ***FileNotFoundException:*** Un intento de acceder a un archivo que no existe.
- ***IOException:*** Fallas de entrada/salida, tal como la inhabilidad de leer desde un archivo.

Excepciones comunes

- ***NullPointerException:*** Referencia a un objeto NULL
- ***NumberFormatException:*** Una conversión fallida entre Strings y números
- ***OutOfMemoryException:*** Muy poca memoria para instanciar un objeto nuevo (new)
- ***SecurityException:*** Un applet tratando de realizar una acción no permitida por la configuración de seguridad del browser

Excepciones comunes

- ***StackOverflowException:*** El sistema corriendo fuera de espacio en Stack (stack space)