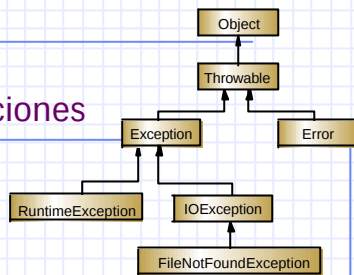


Excepciones



Java

Excepciones

1

Excepciones

- Situaciones anómalas que aparecen durante la ejecución de un programa
 - Cuando aparece una condición excepcional se crea un objeto *Throwable* que se envía al método que la ha generado

Java

Excepciones

2

Excepciones

- Su gestión permite la detección y corrección de errores en ejecución
 - Simplifican los programas ya que se diferencia el código normal del código de tratamiento de errores
 - Se crean programas mas robustos ya que en muchos casos si no se trata la excepción el programa no compila
 - Sólo se deben usar cuando no se puede resolver la situación anómala directamente en ese contexto
 - Se tiene que seguir haciendo el control de errores habitual

Java

Excepciones

3

Dos tipos de situaciones excepcionales

- Excepciones
 - Situaciones más o menos habituales que impiden completar la ejecución correcta del código
 - Generalmente el programador debe proporcionar el código que las trate o gestione
 - Ejemplos
 - Error en el código o en los datos
 - Uso inadecuado de un método
- Ejemplo, la excepción *ArrayIndexOutOfBoundsException* no debería lanzarse nunca si los índices de acceso a un vector no se salen de los límites.

Java

Excepciones

4

Dos tipos de situaciones excepcionales

- Errores
 - Representan situaciones de error normalmente no recuperables
 - El programador normalmente no tiene que proporcionar un tratamiento para ellas
 - Ejemplos
 - No se puede localizar y cargar una clase, Se agota la memoria

Java

Excepciones

5

Tipos de excepciones

- Predefinidas en el sistema
 - Se lanzan automáticamente cuando se realiza alguna operación no válida
 - acceso a un objeto que no existe,
 - acceso a una posición de un array que no existe,
 - división por cero
- Generadas por el programador
 - El programa explícitamente genera una excepción al detectar una situación de error que no se puede resolver en ese contexto
 - Útil en situaciones de prueba y depuración

Java

Excepciones

6

Ejemplo

```
public class HolaMundo {
    public static void main (String
    args[]){
        int i = 0;
        String vectorS [] = {
            "Hola mundo1",
            "Hola mundo 2",
            "Hola mundo 3" };
        while (i < 4 ) {
            System.out.println(vectorS[i]);
            i++; }
        }
    }
```

Java

Excepciones

7

Ejemplo

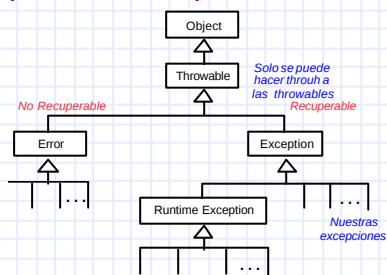
```
c:\...\>java HolaMundo
HolaMundo 1
HolaMundo 2
HolaMundo 3
java.lang.ArrayIndexOutOfBoundsException : 3
    at HolaMundo.main(HolaMundo.java:12)
```

Java

Excepciones

8

Tipos de excepciones



Java

Excepciones

9

Gestión de excepciones

Para gestionar excepciones hay que insertar el código donde se puede dar dentro de un bloque try.

Se crea una lista de bloques catch adyacentes, uno por cada posible excepción.

Java

Excepciones

10

Gestión de excepciones

• Sintaxis

```
try {
    // bloque de código donde puede producirse una excepción
} catch( TipoExcepción1 e ) {
    // gestor de excepciones para TipoExcepción1
    // se ejecuta si se produce una excepción de tipo TipoExcepción1
} catch( TipoExcepción2 e ) {
    // gestor de excepciones para TipoExcepción2
    throw(e); // se puede volver a lanzar la excepción propagar
} finally {
    // bloque de código que se ejecuta siempre, haya o no excepción
}
```

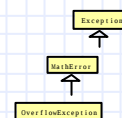
Java

Excepciones

11

Gestión de excepciones

```
try { /*...*/ }
catch (OverflowException e) { /*...*/ }
catch (MathError e) { /*...*/ }
catch (Exception e) { /*...*/ }
finally { /*...*/ }
```



Java

Excepciones

12

Gestión de excepciones

finally

- Es el bloque de código que se ejecuta siempre, haya o no excepción

```
try {
    inicioProceso ();
    gestionProceso();
}
finally {
    finProceso ();
}
```

- La única situación en la que la sentencia finally no se ejecuta es si se llama al método System.exit (), que termina la ejecución del programa.

Java

Excepciones

13

Gestión de excepciones

throw

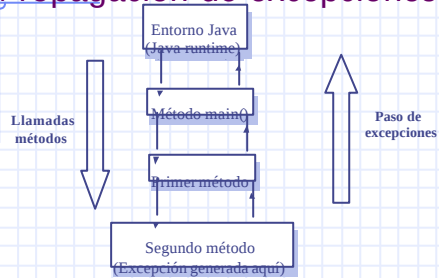
- Permite lanzar de forma explícita una excepción
- Se puede utilizar para propagar una excepción tratada parcialmente

Java

Excepciones

14

Propagación de excepciones

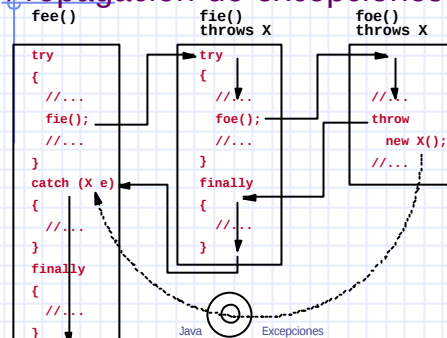


Java

Excepciones

15

Propagación de excepciones



Java

Excepciones

16

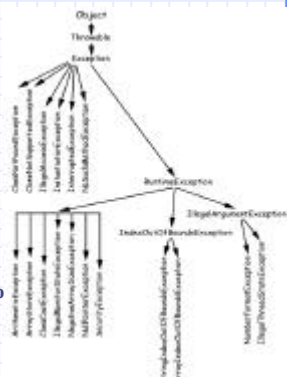
Excepciones

RuntimeExceptions

- ArrayIndexOutOfBoundsException
- ArithmeticException
- NullPointerException
- ClassCastException
- IllegalArgumentException

- No es obligatorio incluir código de tratamiento de este tipo de excepciones

- aunque se puede incluir dicho código



Java

Excepciones

17

Ejemplo simple excepciones

```
public class PruebaExcepciones {
    public static void main(String args[]) {
        int valor=5, cero=0;
        int[] array = {1, 2, 3};
        try {
            valor = valor/cero; //división por cero
            array[4]= 5; //acceso a una posición no disponible
        }
        catch( ArithmeticException e ) {
            System.out.println( "Division por cero" );
        }
        catch( Exception e ) {
            System.out.println( "Se ha producido un error" );
        }
    }
}
```

Java

Excepciones

18

Propagación de excepciones

```
public class PruebaExcepcion {
    public static void main(String args[]) {
        int valor=5, cero=0;
        int[] array = {1, 2, 3};
        try {
            try {
                valor = valor/cero; //división por cero
                array[4]= 5; //acceso a una posición no disponible
            } catch( ArithmeticException e ) {
                System.out.println( "Division por cero" );
                throw e;
            } catch( Exception e ) {
                System.out.println( "Se ha producido un error" );
            }
        } catch( Exception e ){
            System.out.println(e.getMessage());
        }
    }
}
```

throws

Indica que el código producirá una excepción, que no se tratará dentro de él y se pasará al método superior, utilizando la cláusula throws.

```
public void ejemploExcep () throws
IOException
```

A continuación de la palabra reservada throws aparece una lista de todas las excepciones que se pueden dar dentro del método y no serán gestionadas.

Java

Excepciones

20

Gestión incompleta de excepciones

TipoDevuelto nombreMetodo(argumentos) throws listaExcepciones
{ /* cuerpo del método */ }

- Si un método no gestiona o captura todas las excepciones que puede generar (excepto del tipo *Error* o *RuntimeException*) debe especificarlo mediante *throws*

TipoDevuelto nombreMetodo(argumentos) throws listaExcepciones
{ /* cuerpo del método */ }

Java

Excepciones

21

Gestión incompleta de excepciones

```
import java.io.*;
public class PruebaExcepciones {
    public static char leer() throws IOException {
        return (char) System.in.read();
    }
    public static void main(String args[]) {
        try {
            char car=leer();
            System.out.println("Caracter: "+car);
        } catch (IOException e) {
            System.out.println("Error de entrada de datos");
        }
    }
}
```

Java

Excepciones

22

Lanzamiento de excepciones

```
public class Alumno{
    public Alumno(String nombre, String apellidos)
    throws Exception{
        if(nombre == null || apellidos == null)
            throw new Exception("Argumentos no válidos");
        //si el constructor lanza la excepción el objeto
        //no se crea
    }
    public static void main(String args[]) {
        try{
            Alumno alum = new Alumno(null, "hola");
        } catch (Exception e){
            System.out.println("Excepcion: "+
                               e.getMessage());
        }
    }
}
```

Excepciones definidas por el programador

- El programador puede definir sus propias clases de excepciones

- Se define una clase que herede de *Throwable* o más normalmente de *Exception*

```
public class EdadFueraDeRangoException extends Exception {
    public EdadFueraDeRangoException (String texto) {
        super(texto);
    }
}
```

Java

Excepciones

24

Excepciones definidas por el usuario

```
public class FileInputStream
    extends InputStream{
    public FileInputStream(String aFileName)
        throws IOException {
        if (...) {
            throw new IOException("No Such
            File");
        }
        ...
    }
    ...
}
```

Java

Excepciones

25

Excepciones definidas por el usuario

- Gestiona la excepción, incluyendo en el código los bloques try-catch-finally
- Se considera que una excepción está tratada incluso si el bloque catch está vacío.
- Indica que el código producirá una excepción, que no se tratará dentro de él y se pasará al método superior, utilizando la cláusula throws.

Java

Excepciones

26

Excepciones definidas por el usuario

- `public void ejemploExcep () throws IOException`
- A continuación de la palabra reservada `throws` aparece una lista de todas las excepciones que se pueden dar dentro del método y no serán gestionadas.

Java

Excepciones

27

Ejemplo excepción definida por el usuario

```
public class Persona{
    int edad;
    ...
    public void ponEdad(int ed) throws EdadFueraDeRangoException {
        if ((ed < 0) || (ed > 130))
            throw new EdadFueraDeRangoException("Demasiado joven o demasiado viejo");
        edad = ed;
    }
}

try {
    alguien.ponEdad(150);
} catch (EdadFueraDeRangoException e){
    System.out.println("se ha producido la excepción");
    e.printStackTrace();
    System.out.println(e.getMessage());
}
```



```
EdadFueraDeRangoException: Demasiado joven o demasiado viejo
at Persona.ponEdad(Persona.java)
at Persona.main(Persona.java)
```

Java

Excepciones

28

Ejemplo

```
public void connectMe (String serverName)
    throws ServerTimedOutException {
    int success;
    int portToConnect = 80;
    success = open (serverName,
    portToConnect);
    if (success == -1) {
        throw new
        ServerTimedOutException("Imposible
        conectarse", 80);
    }
}
```

Java

Excepciones

29

Ejemplo

```
public void findServer(){
    ...
    try {
        connectMe(defaultServer);
    } catch (ServerTimedOutException e) {
        System.out.println ("Server timed out,
        trying alternate");
    }
    try {
        connectMe(alternateServer);
    } catch (ServerTimedOutException e1) {
        System.out.println("No server
        currently aviable");
    }
    ...
}
```

Java

Excepciones

30

Excepciones más frecuentes

ArithmeticException

```
int i = 12 / 0;
```

NullPointerException

```
Date d = null;
```

```
System.out.println (d.toString());
```

NegativeArraySizeException

Intento de creación de un vector con un número negativo de elementos

Java

Excepciones

31

Excepciones más frecuentes

ArrayIndexOutOfBoundsException

Intento de acceso a un elemento de un vector fuera de rango

SecurityException

Error de seguridad en los navegadores.

Java

Excepciones

32

Sobreescritura de métodos

```
public void test() throws ExceptionA
```

sobre
escribe

```
public void test() throws ExceptionB
```

extends

ExceptionA

ExceptionB

Java

Excepciones

33

Sobreescritura de métodos

```
public void test() throws ExceptionA
```

sobre
escribe

```
public void test() throws ExceptionC
```

extends

Exception

ExceptionA

ExceptionC

Java

Excepciones

34