Tecnología de la Programación

Manejo de errores con excepciones

David Cabrero Souto

Facultad de Informática
Universidade da Coruña

Curso 2007/2008

¿ Por qué ?

- El código falla
- Sino falla, fallará el entorno
- Y sino, fallará el usuario . . .

¿ Qué es el manejo de excepciones ?

- El manejo de excepciones es una *estructura de control* de los *lenguajes de programación* diseñada para manejar condiciones anormales que pueden ser tratadas por el mismo programa que se desarrolla. (*wikipedia*)
- Excepción: evento que interrumpe la ejecución del programa
  - Errores de programación (p.e.: dividir por cero)
  - Errores del sistema (p.e.: fichero no existe)
  - Errores del interprete (p.e.: problemas con librería dinámica)
  - ...

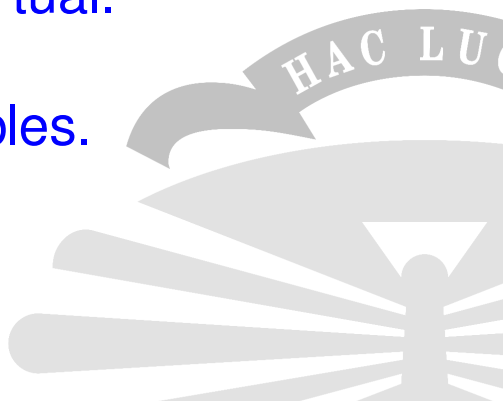
¿ Cómo es el manejo de excepciones ?

- Habitualmente, es una estructura `try/catch`, `try/except`, `begin/exception`, ...

```
try {
        // Código habitual
        // Puede ocurrir una excepción
}
catch {
        // Se ejecuta si ocurre una excepción
}
```

Manejo de excepciones en Java (i)

- Cuando ocurre un error, *se lanza una excepción*
- A cada excepción se le asocia un objeto de tipo `java.lang.Throwable`
- Hay dos subtipos directos de `Throwable`
  - `java.lang.Error`
    Es un error irrecuperable, la ejecución del programa se detiene sin más.
    Ejemplos: problemas en la carga dinámica de clases, problemas de falta de memoria en la máquina virtual.
  - `java.lang.Exception`
    Resto de casos: errores considerados recuperables.

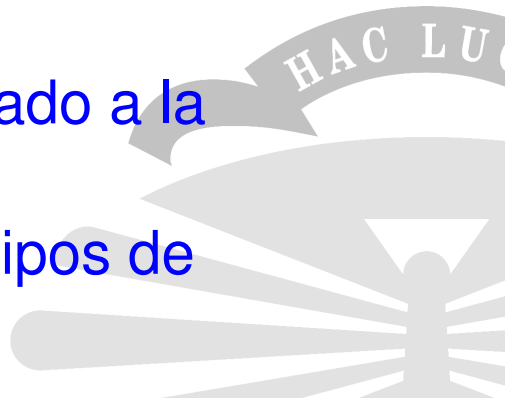


Manejo de excepciones en Java (ii)

- *Capturar la excepción*. Cuando se produce una excepción de tipo `Exception` dentro de un bloque *try*, se ejecuta el bloque *catch* correspondiente, si lo hubiera.

```
try {
        // Código que produce la excepción
}
catch (RuntimeException e) {
        // Código que maneja la excepción
}
catch (IOException e) {
        // Código que maneja la excepción
}
```

- Dentro del bloque *catch* recupero el objeto asociado a la excepción
- Puedo tener varios bloques *catch* para distintos tipos de excepciones



Manejo de excepciones en Java (iii)

- Las excepciones llevan asociado un objeto
  - Son objetos: atributos y métodos
  - La librería (JDK) define varios tipos de excepciones
  - Puedo definir mis subtipos de `Exception`
- Las execpciones *se propagan*
  - Si dentro del método que la provoca no se captura, se examina de forma iterativa la lista de llamadas, bucando un método que la capture, si llegamos al `main`, se interrumpe la ejecución.
- También cláusula `finally`

```
try {
}
catch(Exception e) {
}
finally {
  // Código que se ejecuta siempre
}
```

Manejo de excepciones en Java (iv)

- Puedo lanzar una excepción de forma explícita

```
throw new Exception("Operación no autorizada");
```

- *Checked exceptions*. La lista de excepciones que pueden lanzar un método forman parte de su firma.

```
public void doSomething(...) throws IOException {
   ...
}
```

```
InputFile.java:8: Warning: Exception java.io.FileNotFoundExceptio
must be caught, or it must be declared in throws clause of this m
        fin = new FileInputStream(filename);
              ^
```

Vs. control de errores tradicional (i)

- Separar el código de manejo de errores

```
if (fout = fopen("filename","w") == NULL) {
  // Error
  switch(errno) {
  case ENOENT:
    ...
  default:
    ...
  }
if (fprintf(fout, "...", ...) < 0) {
  // Error
  ...
}
```

Vs. control de errores tradicional (ii)

- Las excepciones se propagan de forma automática

```
void foo() {
  if (bar() < 0) {
    // Error
    ...
  }
}

int bar() {
  if (f() < 0)
    return -1;
  ...
}

int f() {
  if ((fout = fopen("filename","w")) == NULL)
   return -1;
  ...
}
```

Vs. control de errores tradicional (iii)

- Las excepciones son ciudadanos de primera clase: objetos
- Podemos declarar tipos y subtipos