Exception handling in Java

Reading for this lecture:
Weiss, Section 2.5 (exception handling), p. 47.
ProgramLive, chapter 10.

I need to know whether the YOU would like a lecture that reviews classes, subclasses, inheritance, overriding, constructors, apparent and real types, using some example. Please write your opinion on a blank piece of paper and pass it forward, now.

A computer lets you make more mistakes faster than any other invention in history -- with the possible exception of handguns and tequila. Mitch Ratcliffe

The essence of immorality is the tendency to make an exception of myself. Jane Addams.

Exception handling in Java

An exception: something bad or unexpected that happens in a program, e.g.
- divide by zero
- subscript out of bounds
- i/o error
- null pointer exception
- no more memory
- etc.

Big problem: where the exception occurs is not always the best place to handle it (see next slide).

Java has special mechanisms for dealing with exceptions:
- define your own exception
- throw an exception
- catch and handle an exception

Throwing exceptions

Divide by zero? Java “throws” an exception:
```
throw new ArithmeticException("Divide by zero");
```

IO error? Java “throws” an exception:
```
throw new java.io.IOException("…");
```

Integer.parseInt detects that its arg doesn’t contain an int? It “throws” an exception:
```
throw new NumberFormatException("…");
```

You, too, can throw exceptions in your program --and you can define your own classes whose instances are exceptions.

What is an exception?
- Who can throw it?
- What happens when it is thrown?
- Who catches it?

Example from GUI JLiveWindow

// Return the int in integer field f, or 0 if either
// f is invalid or the field doesn’t contain an int
public static int getField(int f) {
    try {
        intFields[f].getTextField();
    } catch (ArrayIndexOutOfBoundsException ex) {
        return 0;
    } catch (NumberFormatException ex2) {
        String t = intFields[f].getText();
        intFields[f].setText( t + "not an integer; use 0");
    }
    return null;
}

execute this field f did not contain an int
execute this if f is out of bounds

A case where detection of an error is not the place to handle it

Class Integer has this method:
```
int v = Integer.parseInt("234X4");
```

Method parseInt doesn’t know how to handle the error; so it should let the caller handle it. But how can it do this without complicating the handling of non-exceptional cases? That’s the question. The answer is:

Use Java’s exception-handling mechanism.
Try statement

```
try <block>
catch ( <class-name> <identifier ) <block>
catch ( <class-name> <identifier ) <block>
...catch ( <class-name> <identifier ) <block>finally <block> [we don’t discuss this]
```

- Need at least one catch clause or the finally clause.
- To execute, execute the try <block>; if no exception is thrown, that’s it. If a throw statement is executed:

  ```
  throw new ...;
  ```

  the exception is thrown. That’s the end of the try block! Who catches the thrown exception is described on next slide.

Class Error

```
Errors that should not be caught!!!
```

Let the system handle them

```
public class Error extends Throwable {  
    public Error(String m) { ... }  
}
```

```
public class Exception extends Throwable {  
    public Exception() { ... }  
    public Exception(String m) {... }  
    public NumberFormatError () {... }  
}
```

```
public class NoClassDefFoundError extends Error {  
    public NoClassDefFoundError (String m) {... }  
}
```

```
public class IOException extends Exception {  
    public IOException() { ... }  
    public IOException(String m) {... }  
    public IOException() {... }  
    public IOException(String m) {... }  
}
```

```
public class NoClassDefFoundError extends Exception {  
    public NoClassDefFoundError (String m) {... }  
    public NoClassDefFoundError () {... }  
}
```

Propagation of a thrown exception

```
public class C {  
    public static void main(String [] pars {  
        try { second(); }  
        catch (XXException ae) {}  
        public static void first() { second(); }  
        public static void second() { ... throw new XXException(); ... }  
    }  
}
```

The Throwable Hierarchy

```
Throwable
   Error (not checked -- explain later)
      Internal Error
      NoClassDefFoundError
      OutOfMemoryError
      AWTError
      StackOverflowError
Exception
   RuntimeException (not checked -- explain later)
      NullPointerException
      IllegalArgument Exception
      NumberFormatException
      ArrayIndexOutOfBoundsException
      ...IOException
      FileNotFoundException
      NumberFormatException
      ...StackOverflowError
      AWTError
      OutOfMemoryError
      Internal
      Error       (not checked -- explain later)
```

Over 25 Errors and 120 Exceptions come with Java.
You can define your own subclasses of Throwable.
The non-boldfaced ones are not checked (explain latter)
The boldfaced ones are checked (explain later)

Example

```
// Read a line from the keyboard and return the integer
// on it. If the line does not contain an integer, keeping
// asking the user for one until they types one
public static int readLineInt() {  
    String s;  
    Read the next line into s;  
    s.trim();  
    // invariant: input contains the last line read; previous  
    // lines did not contain a recognizable integer
    while (true) {  
        try {  
            return Integer.parseInt(s);  
        }  
        catch (NumberFormatException ex) {  
            System.out.println("That wasn’t an integer.");  
            // Type an integer,");
                Read the next line into s;  
                s.trim();  
        }  
    }
```
The throws clause

class public static void main(String[] pars) {
    first();
}

class public static void first() {
    throw new Exception();
}

Illegal program; won’t compile:
    Error: Exception java.lang.Exception must be caught
         or it must be declared in the throws clause of this
         method (method first).

To fix the problem, change method first to:

class public static void first() throws Exception{
    throw new Exception();
}

But then method main is illegal and must be changed to

class public static void main(String[] pars) throws Exception{
    first();
}

Rule: All checked exceptions (see slide 6) must be caught
       or mentioned in a throws-clause. Unchecked exceptions
       need not be mentioned.

You are responsible for knowing:

• About classes Throwable, Error, and Exception.
• How to write your own subclass of Exception.
• How to use a try-statement.
• How an exception propagates up the call chain
   until it is caught by a catch-clause
• How to throw an exception using the throw-
   statement.
• How to insert a throws-clause.

In any culture, subculture, or family in which belief is valued above thought, self-surrender is valued above self-expression, and conformity is valued above integrity, those who preserve their self-esteem are likely to be heroic exceptions.

Nathaniel Branden