

Week 11, Lecture 1

Exceptions

Java



Announcements

- **Assignment 3** is due at the end of **this week**; deadline is **4pm Friday 15th October**.
- **Labs 8, 9 and 10** are due **next week**.
- The **Prac Exams** will be held the **week after** on **Thurs 28th** and **Fri 29th October**. Make sure you know the details of your exam session. See the course web page for details.

Week 11

Lecture 1: Exceptions

Lecture 2: File I/O

Java Genesis:

–Ch11: Exceptions and files

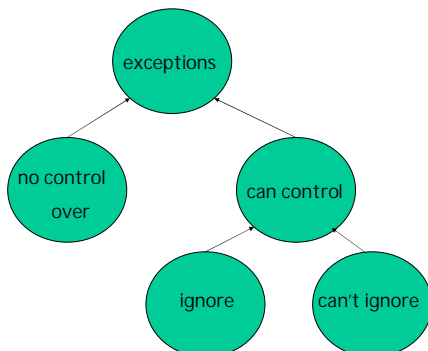
Assignment 3 (deadline 4pm Oct 15th)

Exceptional circumstances

are a fact of life that we can pre-plan for (it's too late when trouble strikes). Consider:

- life boats
- parachutes
- fire escapes
- spare tyres
- first-aid kits
- etc., etc.

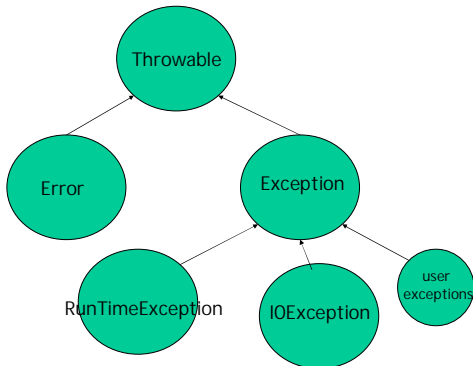
Exception Categories



Programming Exceptions

- user's input is incorrect
- file cannot be found
- run out of disk space
- run out of memory
- arithmetic error (number out of range, divide by zero ...)
- array index out of range
- and so on ...

Java's Emergency Classes



JAVA'S HELP: LANGUAGE CONSTRUCTS - 1

```
try {  
    ...code...  
} catch (ExceptionClass1 c) {  
    what to do with it  
} catch (ExceptionClass2 d) {  
    what to do with it  
}
```

Throwing zone

Specialist catching zone

This style is a contract to deal with exception here, now!!

LANGUAGE CONSTRUCTS - 2

```
public type methodName (argsIfAny)  
    throws ExceptionClass1, ExceptionClass2 {  
    ...  
    code for method  
    ...  
}
```

Throwing zone

This style passes the handling of exceptions to the caller! The catching zone is the caller (or the caller of the caller...). In any case, the caller must deal with it.

Summary

For exceptions we must handle, or for other exceptions we want to handle, use

`try...catch ... statement`

For exceptions that **must** be handled which we elect not to catch locally (or if we want to indicate an exception can occur in some method), use

`resultType methodId (possibleArgs) throws ExceptionList` as part of the method declaration. In this case the onus is placed on the caller to deal with any of these exceptions that occur.

The next 4 slides give the original code for the integer-division calculator

```
import genesis.*;  
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
  
public class DivisionFrame extends JFrame {  
  
    // instance variables  
    private JTextField numerator, denominator, result;  
    private String task = "Enter integers and press '=' to  
                           perform division.";  
  
    private JLabel taskMsg = new JLabel(task, JLabel.CENTER);  
    private String welcome = "Welcome to the deluxe  
                             integer-division calculator";  
    private JLabel statusMsg =  
        new JLabel(welcome, JLabel.CENTER);
```

```
// constructor
public DivisionFrame ( ) {
    setTitle("Performing Integer Division");
    setBounds(50, 100, 750, 200);
    Font f = new Font("SansSerif", Font.BOLD, 30);
    taskMsg.setFont(f);
    taskMsg.setForeground(Color.blue);
    statusMsg.setFont(f);
    statusMsg.setForeground(Color.blue);
    numerator = new JTextField("0", 7);
    numerator.setFont(f);
    denominator = new JTextField("1", 7);
    denominator.setFont(f);
    result = new JTextField("0", 7);
    result.setFont(f);
}
```

```
JLabel divSign = new JLabel("/", JLabel.CENTER);
divSign.setFont(new Font("SansSerif", Font.BOLD, 50));
divSign.setForeground(Color.black);
JButton equalBut = new JButton("=");
equalBut.setFont(f);
equalBut.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        int numValue =
            Integer.parseInt(numerator.getText());
        int denomValue =
            Integer.parseInt(denominator.getText());
        int divValue = numValue/denomValue;
        result.setText(""+divValue);
    }
});
```

```
JPanel p1 = new JPanel ();
p1.add(numerator);
p1.add(divSign);
p1.add(denominator);
p1.add(equalBut);
p1.add(result);
JPanel p2 = new JPanel ();
p2.add(taskMsg);
JPanel p3 = new JPanel ();
p3.add(statusMsg);
Container c = getContentPane();
c.add(p1, "Center");
c.add(p2, "South");
c.add(p3, "North");
}
}
```

The last 2 slides give the exception-handling modifications made to the original code.

```
equalBut.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        try {
            performDivision();
        } catch (NumberFormatException e) {
            statusMsg.setForeground(Color.red);
            statusMsg.setText("numbers must both
                be integers: try again");
            taskMsg.setText(" ");
        } catch (ArithmeticException e) {
            statusMsg.setForeground(Color.red);
            statusMsg.setText("cannot divide by
                zero: try again");
            taskMsg.setText(" ");
        }
    }
});
```

```
public void performDivision ( ) throws
    NumberFormatException,
    ArithmeticException {
    int numValue =
        Integer.parseInt(numerator.getText());
    int denomValue =
        Integer.parseInt(denominator.getText());
    int divValue = numValue/denomValue;
    result.setText(""+divValue);
    statusMsg.setForeground(Color.blue);
    statusMsg.setText("division performed
        without error");
    taskMsg.setText(task);
}
```