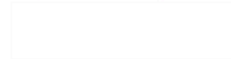


# Week 11, Lecture 1

## Exceptions

EXPLORING  
**Java**



# Week 11

**Lecture: Exceptions & File I/O**

*Java Genesis:*

–**Ch11: Exceptions and files**

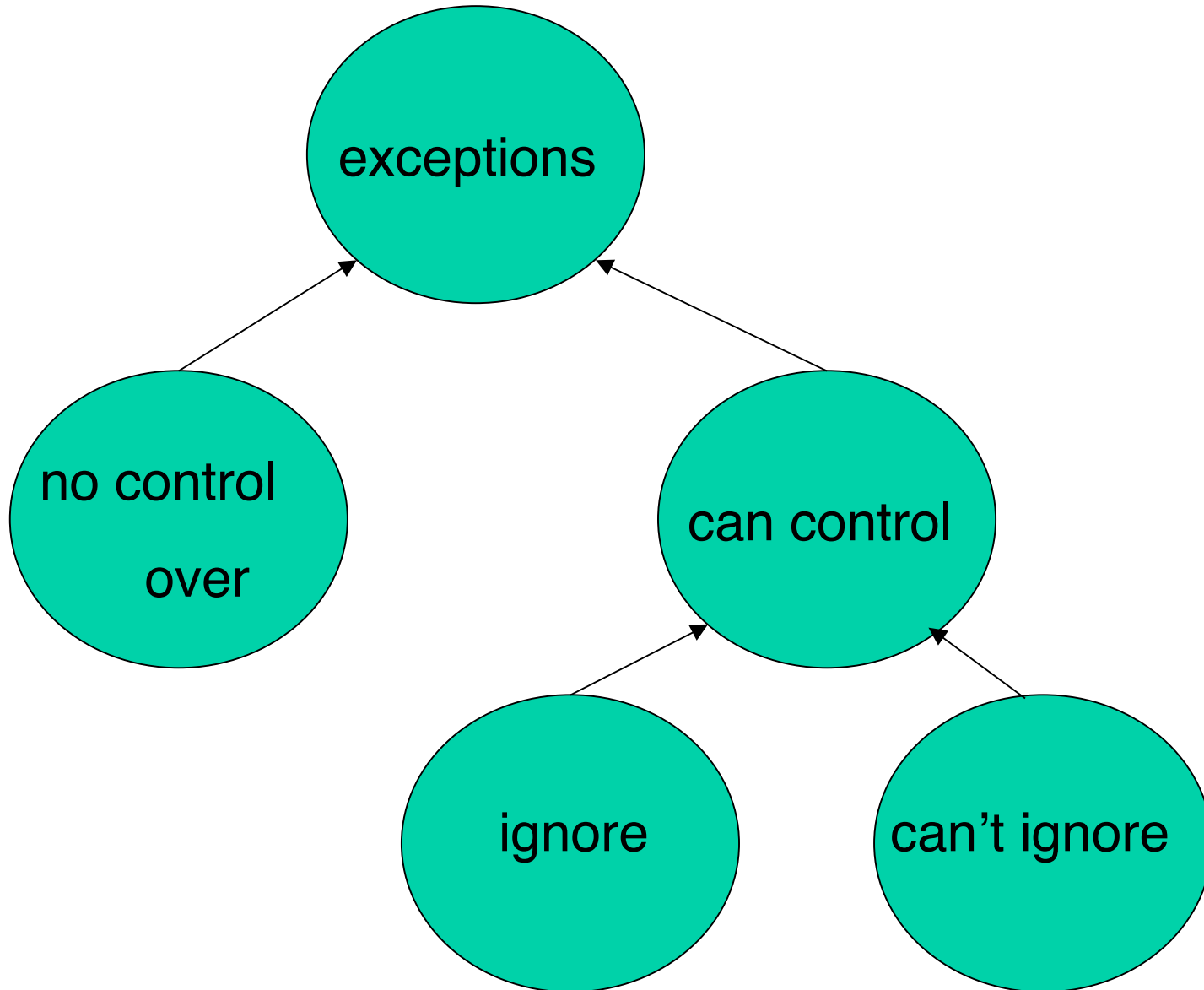
**Assignment 3**

# 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.

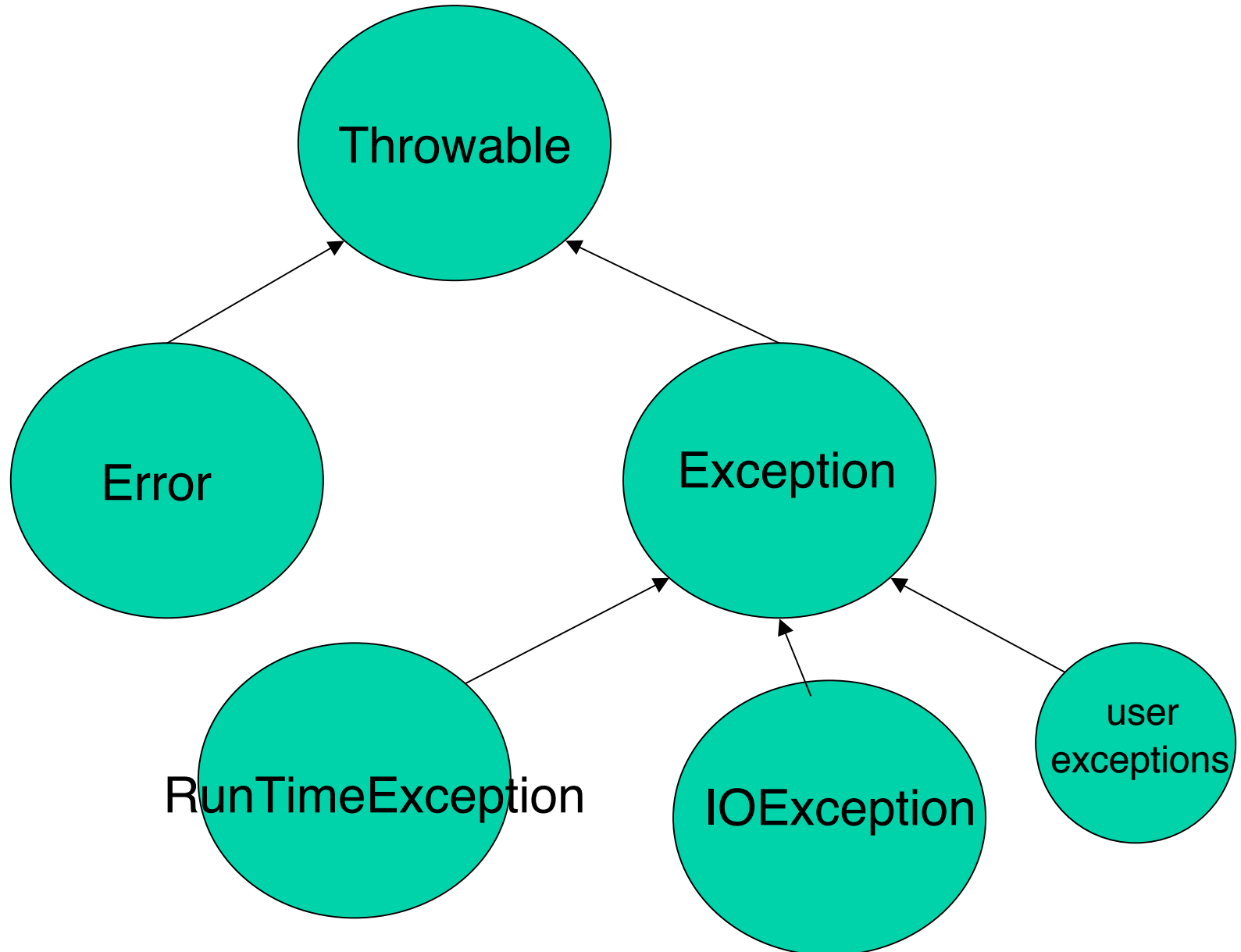
# Exception Categories



# 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
```

```
}
```

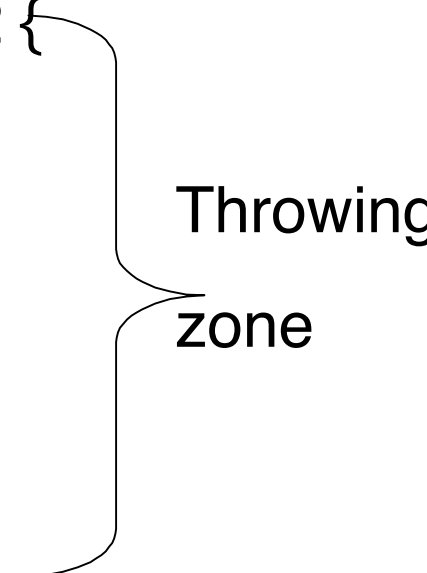
This style is a contract to deal with exception here, now!!

Throwing zone

Specialist  
catching zone

## LANGUAGE CONSTRUCTS - 2

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



The diagram illustrates the 'Throwing zone' of a method signature. A large right-facing curly bracket groups the `throws ExceptionClass1, ExceptionClass2` clause and the entire method body (including the `code for method` and the ellipsis `...`). The text 'Throwing zone' is positioned to the right of the bracket, indicating that the caller is responsible for handling any exceptions thrown within this 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



```
// 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 );
}
```

# Week 11, Part 2

## File I/O

EXPLORING  
**Java**



```
import java.io.*;
import genesis.*;
```

```
public class ReadPoem {
```

```
    public static void main (String [ ] args)
                                throws IOException {
```

```
        FileReader fr = new FileReader("poem.txt");
```

```
        BufferedReader br = new BufferedReader(fr);
```

```
        String line = br.readLine( );
```

```
        while (line != null) {
```

```
            Transcript.println(line);
```

```
            line = br.readLine( );
```

```
        }
```

```
        br.close( );
```

```
    }
```

```
}
```

```
import java.io.*;
import genesis.*;
public class SumNumbers {

    public static void main (String [ ] args)
                                throws IOException {

        FileReader fr =
            new FileReader("numbers.txt");
        BufferedReader br = new BufferedReader(fr);
        int sum = 0;
        String line = br.readLine( );
        while (line != null) {
            Transcript.println(line);
            sum = sum + Integer.parseInt(line);
            line = br.readLine( );
        }
        Transcript.println("Sum is "+sum);
        br.close( ); } }
```

```
import java.io.*;
import genesis.*;
import java.util.*;
public class SumNumbersAgain {

    public static void main (String [ ] args)
                                throws IOException {

        FileReader fr =
            new FileReader("numbers again.txt");
        BufferedReader br = new BufferedReader(fr);
        int sum = 0;
        String line = br.readLine( );
```

```
while (line != null) {  
    Transcript.println(line);  
    StringTokenizer st =  
        new StringTokenizer(line);  
    while (st.hasMoreTokens( )) {  
        sum = sum +  
            Integer.parseInt(st.nextToken( ));  
    }  
    line = br.readLine( );  
}  
Transcript.println("Sum is "+sum);  
br.close( );  
}  
}
```

```
import java.io.*;

public class WritePoem {

    public static void main (String [ ] args)
        throws IOException {
        FileWriter fw =
            new FileWriter("another poem.txt");
        PrintWriter pw = new PrintWriter(fw);
        pw.println("-----");
        pw.println("-----");
        pw.close( );
    }
}
```

```
import java.io.*;

public class AppendPoem {

    public static void main (String [ ] args)
        throws IOException {

        FileWriter fw =
            new FileWriter("another poem.txt", true);
        PrintWriter pw = new PrintWriter(fw);
        pw.println("-----");
        pw.println("-----");
        pw.close( );
    }
}
```

```
import java.io.*;
import java.util.*;
public class ProcessMarks {

    public static void main (String [ ] args)
                                throws IOException {
        FileReader fr = new FileReader("marks.txt");
        BufferedReader br = new BufferedReader(fr);
        FileWriter fw =
            new FileWriter("results.txt");
        PrintWriter pw = new PrintWriter(fw);
        int sum;
        String line = br.readLine( );
```

```
while (line != null) {
    pw.print(line);
    sum = 0;
    StringTokenizer st =
        new StringTokenizer(line);
    while (st.hasMoreTokens( )) {
        try {
            sum = sum +
                Integer.parseInt(st.nextToken( ));
        } catch (NumberFormatException e) { }
    }
    pw.println(" sum: "+sum);
    line = br.readLine( );
}
br.close( );
pw.close( );
}
}
```