

## Lecture 14 Summary

- Exceptions vs. Errors
- Exceptions vs. RuntimeExceptions
- `try...catch...finally`
- `throw` and `throws`

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

1

By the end of this lecture, you will be able to differentiate between errors, exceptions, and runtime exceptions.

You will also be able to describe the class model for exceptions in Java.

You will also be able to create, throw and catch your own exceptions.

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

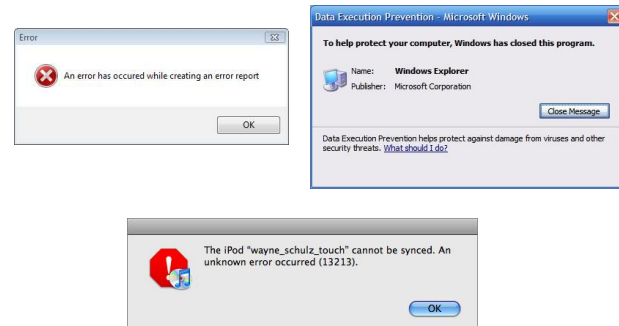
2

What's the worst/funniest error message you've seen on your computer?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

3



March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

4

How have you handled errors in *your* programs so far?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

5

## Error Codes

- Did you run into any difficulties using `atof`?
  - what happens when you enter 0?
  - what happens when you enter “hello”?
- `strtod`: sets the value of a global variable called `errno` to something other than 0 (depending on the type of error).

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

6

If you considered all of the possible stuff someone could type into your program and handled it “nicely”, what would the code look like in C?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

7

## Reporting Errors (GNOME)

- GNOME Human Interface Guidelines 2.2
  - 11.2 Warning and Error Messages:

“A good warning or error message contains two elements:

  1. A brief description of the problem.
  2. A list of ways the user can remedy the problem.

Both of these elements should be presented in non-technical, jargon-free language, unless your target audience is particularly technically-minded.”

March 9, 2009

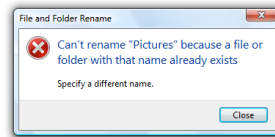
Slides by Mark Hancock  
(adapted from notes by Craig Schock)

8

## Reporting Errors (Vista)

- Windows User Experience Interaction Guidelines (Windows Vista)

“Effective error messages inform users that a problem occurred, explain why it happened, and provide a solution so users can fix the problem.”



March 9, 2009

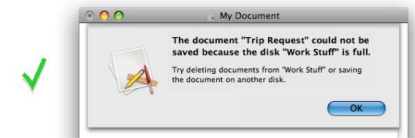
Slides by Mark Hancock  
(adapted from notes by Craig Schock)

9

## Reporting Errors (Apple)

- Apple Human Interface Guidelines

“Provide useful error messages to users when something does go wrong. An error message should clearly convey what happened, why it happened, and the options for proceeding.”

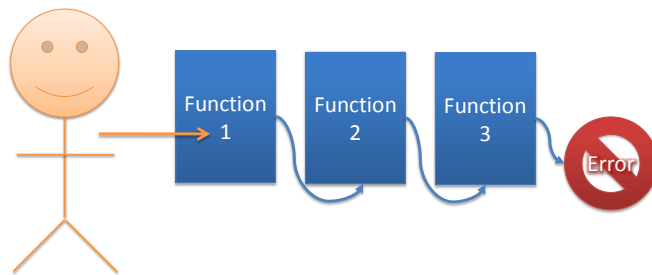


March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

10

## Dealing with Errors



Where would you put the code to handle the error?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

11

## Possibilities

- Handled in Function 3:
  - Advantage: know specific reason for error
  - Disadvantage: don't know context
- Handled in Function 1:
  - Advantage: know about context
  - Disadvantage: need code added to Functions 1 & 2 to pass back info about reason for error

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

12

## Exceptions

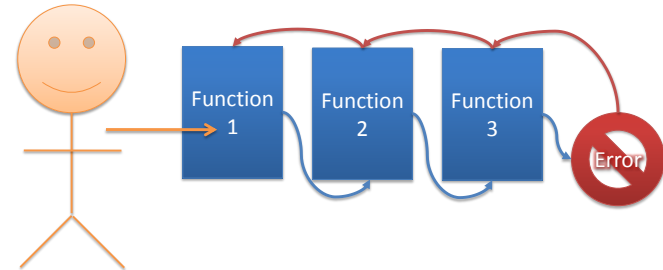
- Basic idea:
  - Program flows according to usual constructs (sequential, conditional, iterative execution, etc.)
  - Something *exceptional* happens
  - Interrupt normal behaviour by *throwing* an exception (an object with error information)
  - Somewhere (anywhere) up the call stack, this exception can be *caught* and handled
  - Transfer control to the catching method (removing everything below it from the call stack)

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

13

## Throwing an Exception



March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

14

## Java Exception Handling Keywords

```
try
catch
finally
```

```
throw
throws
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

15

## Catching an Exception

```
try
{
    // ... some code which might throw an exception
}
catch (Exception x)
{
    // ... code which handles the exception
}
finally
{
    // ... code which is executed no matter what
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

16

## Example

```
import java.util.ArrayList;

public class ExceptionExample
{
    public static void main(String[] args)
    {
        ArrayList<String> strs = new ArrayList<String>();
        String s = strs.get(100);
        System.out.println(s);
    }
}
```

What will happen?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

17

## Example

```
import java.util.ArrayList;

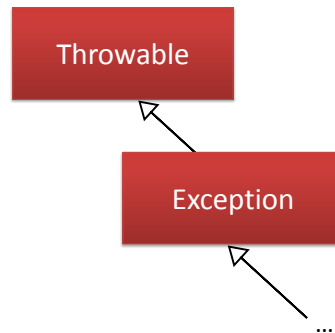
public class ExceptionExample
{
    public static void main(String[] args)
    {
        try
        {
            ArrayList<String> strs = new ArrayList<String>();
            String s = strs.get(100);
            System.out.println(s);
        }
        catch (Exception x)
        {
            System.err.println("An error occurred");
            System.err.println(x.getMessage());
            x.printStackTrace(System.err);
        }
    }
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

18

## Exceptions are Objects



March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

19

## Example: Out of Bounds

```
import java.util.ArrayList;

public class ExceptionExample
{
    public static void main(String[] args)
    {
        try
        {
            ArrayList<String> strs = new ArrayList<String>();
            String s = strs.get(100);
            System.out.println(s);
        }
        catch (ArrayIndexOutOfBoundsException x)
        {
            System.err.println("An error occurred");
            System.err.println(x.getMessage());
            x.printStackTrace(System.err);
        }
    }
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

20

## Example: Parsing Integers

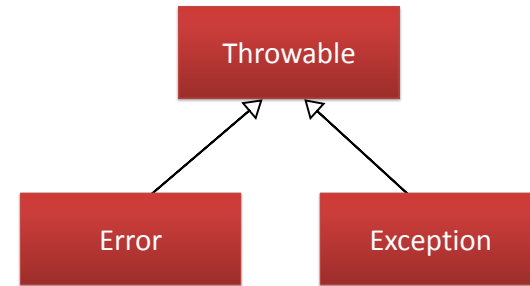
```
public class Number
{
    public static void main(String[] args)
    {
        int count = 0;
        while(count < args.length)
        {
            try
            {
                int x = Integer.parseInt(args[count]);
                System.out.println("Args[" + count + "] = " + x);
            }
            catch (NumberFormatException x)
            {
                System.err.println("Invalid parameter:" + count);
                System.err.println(x.getMessage());
                System.err.println("Stack trace:");
                x.printStackTrace(System.err);
            }
            count++;
        }
    }
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

21

## Errors vs. Exceptions



March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

22

## Errors vs. Exceptions

- Errors
  - Something the programmer **should not** reasonably be expected to recover from.
- Exceptions
  - Something the programmer **should** reasonably be expected to recover from

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

23

## Exercise: Error or Exception

- Read data outside array boundary
- Heap is full (out of memory)
- Null pointer
- File does not exist
- Hard disk failure

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

24

## Catching Multiple Exceptions

```
try
{
    // ... some code which might throw an exception
}
catch (ExceptionOne x)
{
    // ... code which handles ExceptionOne's
}
catch (ExceptionTwo x)
{
    // ... code which handles ExceptionTwo's
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

25

```
import java.io.*;

public class FileExample
{
    public static void main(String[] args)
    {
        try
        {
            FileReader reader = new FileReader(args[0]);

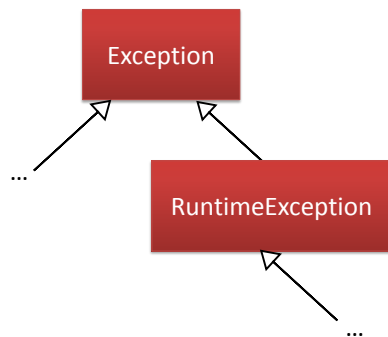
            int c = reader.read();
            reader.close();
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            System.err("Must specify an argument");
        }
        catch (FileNotFoundException e)
        {
            System.err("File not found: " + args[0]);
        }
        catch (IOException e)
        {
            System.err("Error reading file: " + args[0]);
            e.printStackTrace();
        }
    }
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

26

## Checked vs. Unchecked Exceptions



March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

27

## Checked vs. Unchecked Exceptions

- Checked
  - Subclasses of Exception that **do not** inherit from RuntimeException
  - Compiler will complain if uncaught
- Unchecked
  - Subclasses of RuntimeException
  - Can be ignored by the programmer

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

28

## Exercise

- Lookup `ClassCastException` in Java API
  - Is it checked or unchecked?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

29

## Creating Exceptions

```
public class DriverMismatchException
    extends Exception
{
    public DriverMismatchException(String message)
    {
        super(message);
    }
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

30

## Throwing Exceptions

```
public class Driver
{
    private Car car;

    public Driver(Car car)
        throws DriverMismatchException
    {
        if (car.getDriver() != this)
        {
            throw new DriverMismatchException(
                "Car does not match driver");
        }

        this.car = car;
    }
}
```

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

31

## Creating & Throwing Exceptions

- Creating Exceptions
  - Inherit from `Exception`
  - Can inherit from subclass of `Exception`
    - e.g., `IllegalArgumentException`, `RuntimeException`
- Throwing Exceptions
  - Use **`throw`** command
  - If checked, requires a **`throws`** declaration

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

32



## Example: Constructors

- Has no return value
- If an invalid state is reached, exceptions are a nice way to handle the error
- Is there another way?

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

33

## Exercise

- Look up one of each of the following:
  - an Error
  - an unchecked Exception
  - a checked Exception
- Draw the class model starting at the Throwable class.

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

34

## Lecture 14 Summary

- Exceptions vs. Errors
- Exceptions vs. RuntimeExceptions
- `try...catch...finally`
- `throw` and `throws`

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

35

## Next Class

- Collections

March 9, 2009

Slides by Mark Hancock  
(adapted from notes by Craig Schock)

36