

# Exceptions and File I/O

---

COP3330 - Fall 2006  
University of Central Florida

H. Schwartz

---

## Outline

---

- An Exception
- throws
- try - catch
- finally
- Creating Exceptions
- File output

# An Exception

---

- runtime: "abnormal event"
  - Examples:
    - array index out of bounds
    - dividing by zero
    - trying to open a nonexistent file

---

COP 3330 - Fall 2006

# An Exception

---

```
...
File file = new File("example.txt");
//Scanner can be used with files as well
Scanner fIn = new Scanner(file);
    //could cause FileNotFoundException if file doesn't exist
...
fIn.nextInt();//could cause IOException if there is not an int next
```

- How to handle:
  - throw: allow the exception
  - try - catch: handle gracefully

---

COP 3330 - Fall 2006

# throws

```
public static void main (String [] args) throws FileNotFoundException {
    File file = new File("example.txt");

    Scanner fIn = new Scanner(file);
    //could cause FileNotFoundException if file doesn't exist
    ...
}
```

- Allows a method to throw the exception
  - A warning to others that the method may do this - leaves handling to others

COP 3330 - Fall 2006

# try-catch

- Handle the exception without causing a runtime error.
  - Continue running
  - Program exits due to code

```
...
File file = new File(stdin.next());
//Scanner can be used with files as well
try {
    Scanner fIn = new Scanner(file);
}
catch (FileNotFoundException e){
    System.err.println(file + " could not be found");
    System.exit(0); //terminates running the program
}
...
```



**Tangent:**  
Standard Error  
Stream

COP 3330 - Fall 2006

# try-catch

- You may have multiple catch blocks for one try block.

```
...
int a = 0;
try {
    int n = fIn.nextInt();
    int d = fIn.nextInt();
    a = n / d;
}
catch (InputMismatchException e){
    //the input was the wrong type ...
}
catch (NoSuchElementException e){
    //there was no more input to get ...
}
catch (ArithmeticException e){
    //d was probably 0 and the division caused an error ...
}
}
```

COP 3330 - Fall 2006

# finally

- always run after a try-catch

```
...
//from example on previous page
...
catch (ArithmeticException e){
    //d was probably 0 and the division caused an error ...
}
finally {
    a = 0;//reset a to 0
}
}
```

COP 3330 - Fall 2006

# Creating Exceptions

---

- An Exception is actually a class
  - must extend Exception

```
public class WeirdException extends Exception {  
  
    private String secondaryMessage;  
    public WeirdException(String str){  
        //constructor for WeirdException, takes in a string  
        super(str); //passes the string to Exception  
        this.secon...;  
    }  
}
```

---

COP 3330 - Fall 2006

# Creating Exceptions

---

- To throw at any given point..
  - throw is a keyword
  - follow with an Object, which is an Exception

```
throw new WeirdException("something weird happened");
```

---

COP 3330 - Fall 2006

# Writing to a file

---

- Many options
- One of the easiest: `FileWriter`
  - Inherits `Writer`, and methods:
    - `write(String str, int off, int len)`
    - `write(String str)`
    - `append(CharSequence csq)`
    - `close()`

COP 3330 - Fall 2006

# Writing to a file

---

- Constructors:
  - `FileWriter(String fName)` //nonappend mode
  - `FileWriter(String file, boolean b)` //b indicate wheter to append

```
//open in append mode
FileWriter fWrt = new FileWriter(fileName, true);

fWrt.write("Hello World!");

fWrt.close();
```

\* write throws `IOException`

COP 3330 - Fall 2006