Java Exception Handling

Dealing with errors using Java's exception handling mechanism

Approaches For Dealing With Error Conditions

Use conditional statements and return values

Use Java's exception handling mechanism

Approaches For Dealing With Error Conditions

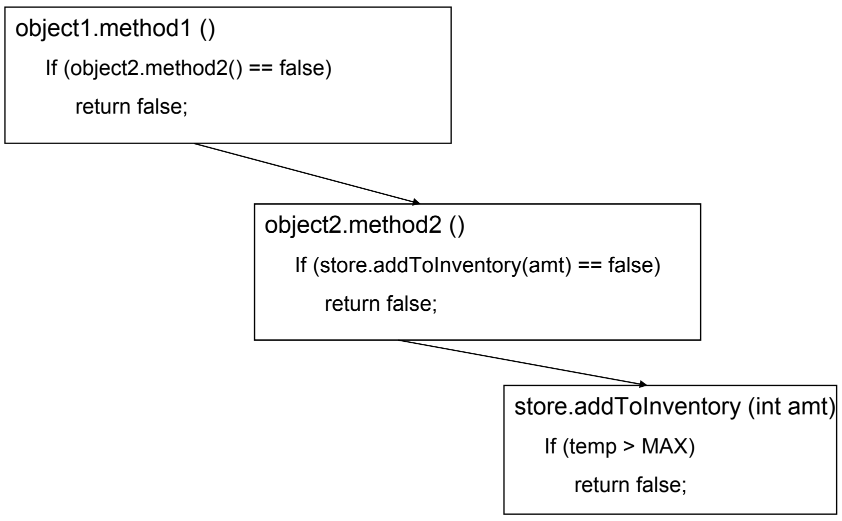
Use conditional statements and return values

Use Java’s exception handling mechanism

Class Inventory: An Earlier Example

```
class Inventory
{
    public  boolean addToInventory (int amount)
    {
       int temp = stockLevel + amount;
       if (temp > MAX)
       {
          System.out.print("Adding " + amount + " item will cause stock ");
          System.out.println("to become greater than " + MAX + " units");
           return false;
       }
       else
       {
          stockLevel = stockLevel + amount;
          return true;
       }
    }
    :
```

Some Hypothetical Method Calls: Condition/Return

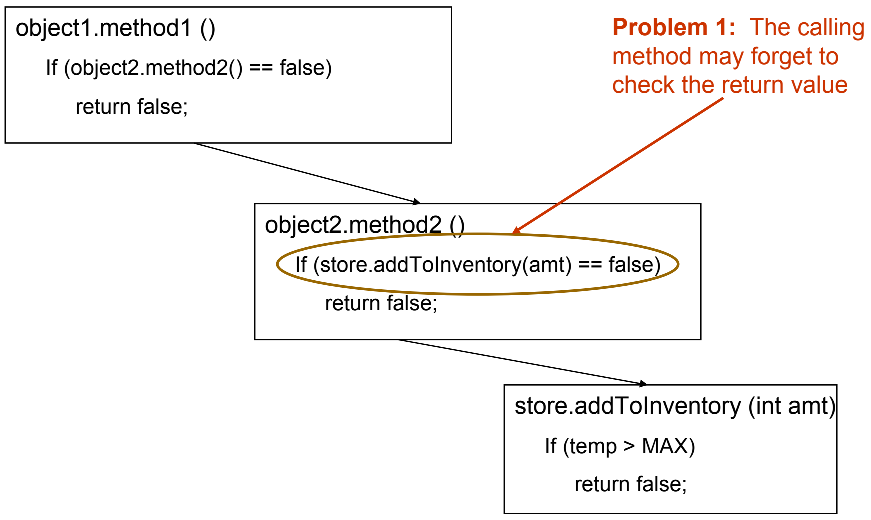


```
object1.method1 ()
    If (object2.method2() == false)
        return false;

object2.method2 ()
    If (store.addToInventory(amt) == false)
        return false;

store.addToInventory (int amt)
    If (temp > MAX)
        return false;
```

Some Hypothetical Method Calls: Condition/Return



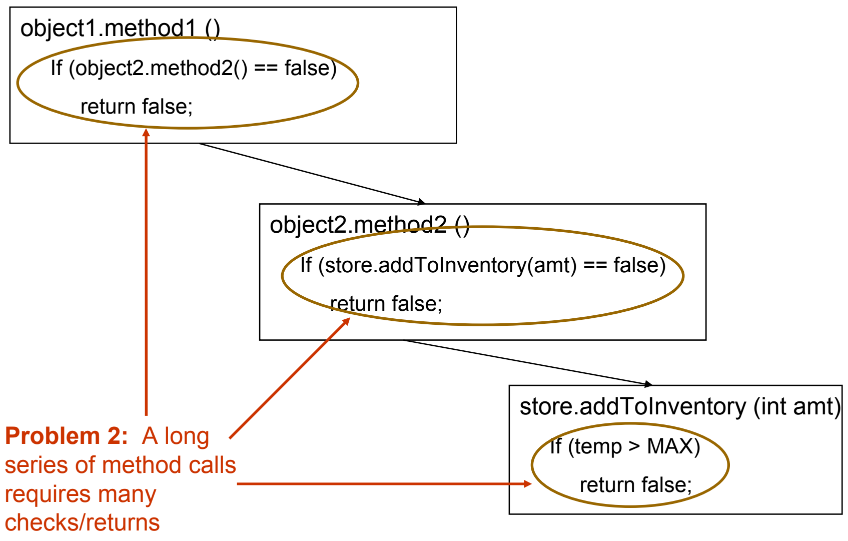
```
object1.method1 ()
    If (object2.method2() == false)
        return false;

object2.method2 ()
    If (store.addToInventory(amt) == false)
        return false;

store.addToInventory (int amt)
    If (temp > MAX)
        return false;
```

Problem 1: The calling method may forget to check the return value

Some Hypothetical Method Calls: Condition/Return



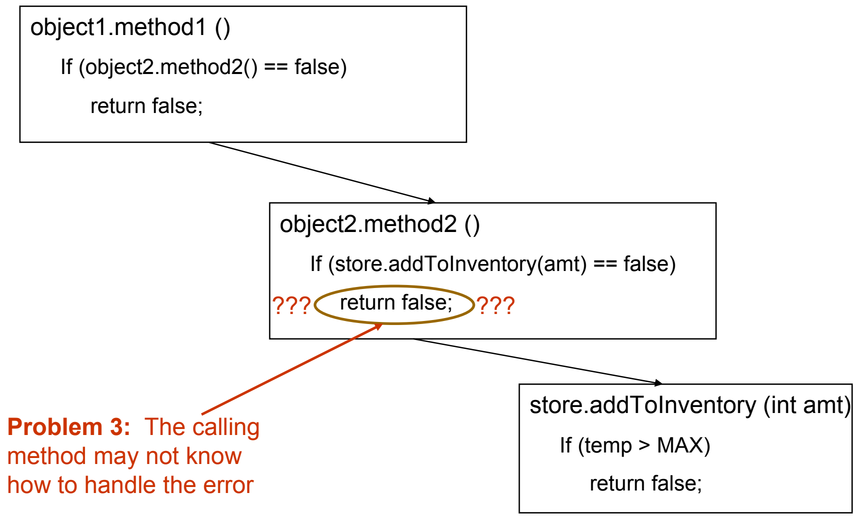
```
object1.method1 ()
    If (object2.method2() == false)
        return false;
```

```
object2.method2 ()
    If (store.addToInventory(amt) == false)
        return false;
```

```
store.addToInventory (int amt)
    If (temp > MAX)
        return false;
```

Problem 2: A long series of method calls requires many checks/returns

Some Hypothetical Method Calls: Condition/Return



```
object1.method1 ()
    If (object2.method2() == false)
        return false;
```

```
object2.method2 ()
    If (store.addToInventory(amt) == false)
???     return false; ???
```

```
store.addToInventory (int amt)
    If (temp > MAX)
        return false;
```

Problem 3: The calling method may not know how to handle the error

Approaches For Dealing With Error Conditions

Use conditional statements and return values

Use Java's exception handling mechanism

Handling Exceptions

Format:

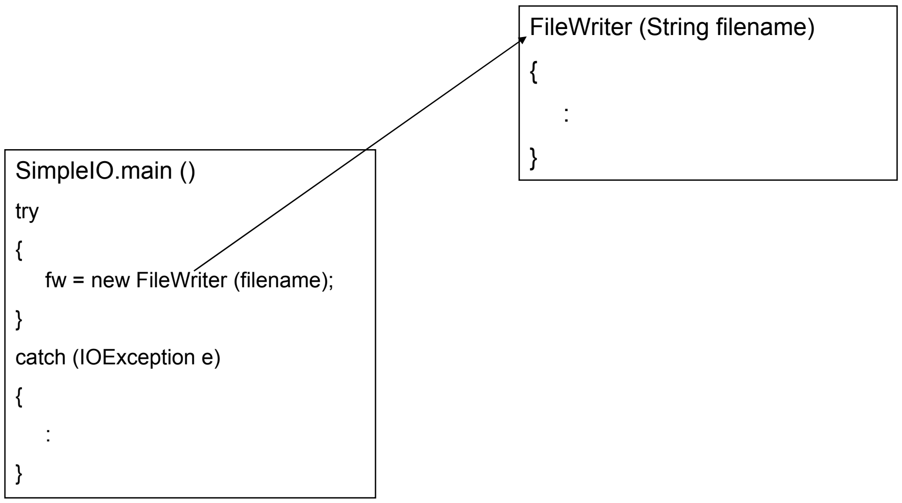
```
try
{
    // Code that may cause an exception to occur
}
catch (ExceptionType identifier)
{
    // Code to handle the exception
}
```

Handling Exceptions: An Example Revisited

The complete program can be found in the directory:
/home/profs/tamj/233/examples/exceptions/handlingExceptions/firstExample

```
class SimpleIO
{
   public static void main (String [] argv)
   {
       :
       try
       {
          fw = new FileWriter (filename);
              :
       }
       catch (IOException e)
       {
              :
       }
```

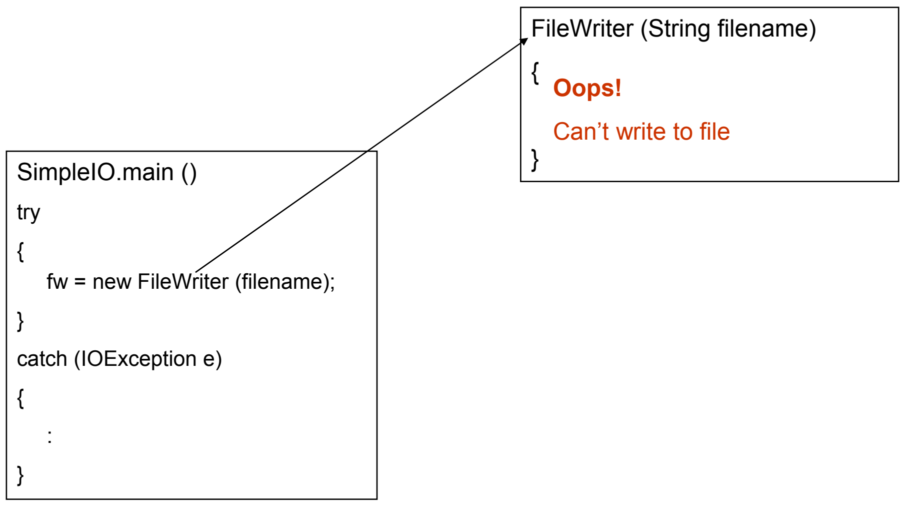
Handling Exceptions: An Example Revisited



```
SimpleIO.main ()
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    :
}
```

```
FileWriter (String filename)
{
    :
}
```

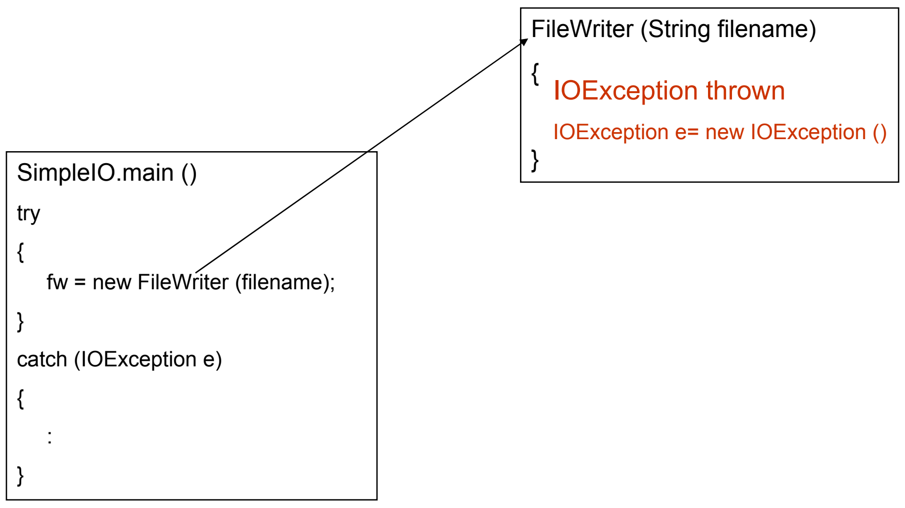
Handling Exceptions: An Example Revisited



```
SimpleIO.main ()
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    :
}
```

```
FileWriter (String filename)
{  Oops!
   Can't write to file
}
```

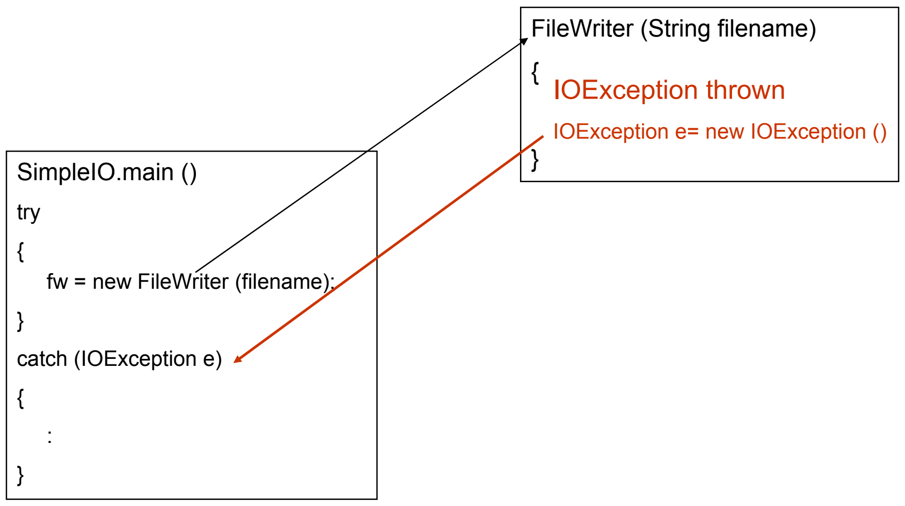
Handling Exceptions: An Example Revisited



```
SimpleIO.main ()
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    :
}
```

```
FileWriter (String filename)
{  IOException thrown
   IOException e= new IOException ()
}
```

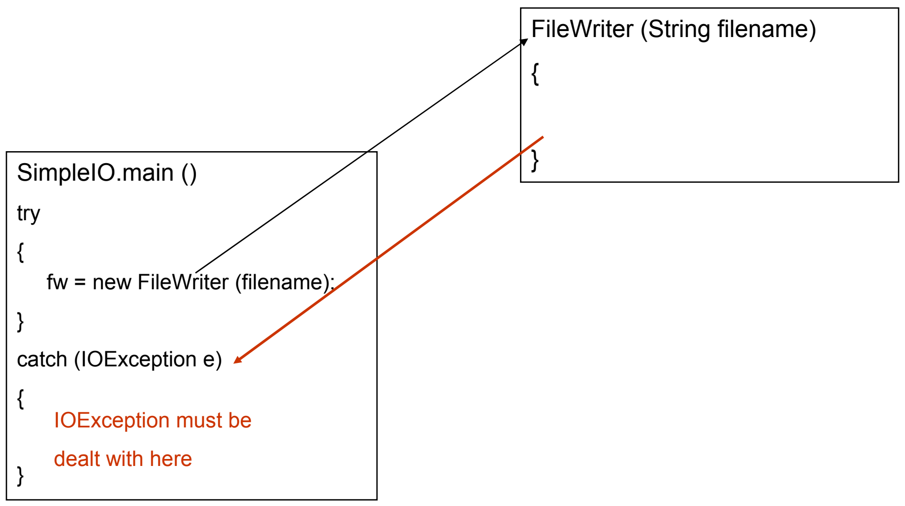
Handling Exceptions: An Example Revisited



```
FileWriter (String filename)
{
    IOException thrown
    IOException e= new IOException ()
}
```

```
SimpleIO.main ()
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    :
}
```

Handling Exceptions: An Example Revisited



```
FileWriter (String filename)
{

}
```

```
SimpleIO.main ()
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    IOException must be
    dealt with here
}
```

Handling Exceptions: An Example Revisited

```
try
{
   fw = new FileWriter (filename);
   pw = new PrintWriter (fw);

   pw.println(iw1.getNum());
   pw.close();
   fr = new FileReader(filename);
   br = new BufferedReader(fr);

  System.out.println("Read from file: " + br.readLine());

}
```

Handling Exceptions: An Example Revisited

```
try
{
   fw = new FileWriter (filename);        <- Exception 1
   pw = new PrintWriter (fw);

   pw.println(iw1.getNum());
   pw.close();
   fr = new FileReader(filename);
   br = new BufferedReader(fr);

  System.out.println("Read from file: " + br.readLine());

}
```

Where The Exceptions Occur In Class FileWriter

For online documentation for this class go to:

http://java.sun.com/j2se/1.4.1/docs/api/java/io/FileWriter.html

```
Class FileWriter
{
   public FileWriter (String fileName) throws IOException;
   public FileWriter (String fileName, boolean append) throws IOException;
        :



}
```

Handling Exceptions: An Example Revisited

```
try
{
   fw = new FileWriter (filename);
   pw = new PrintWriter (fw);

   pw.println(iw1.getNum());
   pw.close();
   fr = new FileReader(filename);        Exception 2
   br = new BufferedReader(fr);

  System.out.println("Read from file: " + br.readLine());

}
```

Where The Exceptions Occur In Class FileReader

For online documentation for this class go to:

http://java.sun.com/j2se/1.4.1/docs/api/java/io/FileReader.html

```
Class FileReader
{
   public FileReader (String fileName) throws FileNotFoundException;
   public FileReader (File file) throws FileNotFoundException;
       :
}
```

Handling Exceptions: An Example Revisited

```
try
{
   fw = new FileWriter (filename);
   pw = new PrintWriter (fw);

   pw.println(iw1.getNum());
   pw.close();
   fr = new FileReader(filename);
   br = new BufferedReader(fr);

   System.out.println("Read from file: " + br.readLine());

}
```

Exception 3

Where The Exceptions Occur In Class BufferedReader

For online documentation for this class go to:

http://java.sun.com/j2se/1.4.1/docs/api/java/io/BufferedReader.html

```
Class BufferedReader
{
        public BufferedReader (Reader in);
        public BufferedReader (Reader in, int sz);
        public String readLine () throws IOException;
                :
}
```

Handling Exceptions: An Example Revisited

```
catch (IOException e)
{
   System.out.println("File IO error: Exception thrown");
   System.out.println(e);

   System.out.println();
   e.printStackTrace();
}
```

Handling Exceptions: An Example Revisited

```
catch (IOException e)
{
   System.out.println("File IO error: Exception thrown");
   System.out.println(e);

   System.out.println();
   e.printStackTrace();
}
```

```
java.io.FileNotFoundException: data
(No such file or directory)
```

```
java.io.FileNotFoundException: data (No such file or directory)
     at java.io.FileInputStream.open(Native Method)
     at java.io.FileInputStream.<init>(FileInputStream.java:103)
     at java.io.FileInputStream.<init>(FileInputStream.java:66)
     at java.io.FileReader.<init>(FileReader.java:41)
     at SimpleIO.main(SimpleIO.java:35)
```

Common Exceptions

NullPointerException
ArrayIndexOutOfBoundsException
ArithmeticException

Common Exceptions: An Example

```
int [] arr = null;
arr[0] = 1;            ← NullPointerException

arr = new int [4];
int i;
for (i = 0; i <= 4; i++)
    arr[i] = i;

arr[i-1] = arr[i-1] / 0;
```

Common Exceptions: An Example

```
int [] arr = null;
arr[0] = 1;

arr = new int [4];
int i;
for (i = 0; i <= 4; i++)
    arr[i] = i;        ← ArrayIndexOutOfBoundsException
                         (when i = 4)

arr[i-1] = arr[i-1] / 0;
```

Common Exceptions: An Example

```
int [] arr = null;
arr[0] = 1;

arr = new int [4];
int i;
for (i = 0; i <= 4; i++)
    arr[i] = i;

arr[i-1] = arr[i-1] / 0;
```

ArithmeticException

(Division by zero)

Categories Of Exceptions

Unchecked exceptions

Checked exception

Unchecked Exceptions

- The compiler doesn't require you to handle them if they are thrown.
- They can occur at any time in the program (not just for a specific method)
- Typically they are fatal runtime errors that are beyond your control
  - Use conditional statements rather than the exception handling model.
- Examples: NullPointerException,IndexOutOfBoundsException, ArithmeticException…

Checked Exceptions

Must be handled if they are ever thrown

- Use a try-catch block

Deal with problems that occur in a specific place

- When a particular method invoked

Example: IOException

Avoid Squelching Your Exceptions

```
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    // Do nothing here.  Just set up the try-catch block to bypass those pesky
    // syntax errors.
}
```

Avoid Squelching Your Exceptions

```
try
{
    fw = new FileWriter (filename);
}
catch (IOException e)
{
    // Do nothing here.  Just set up the try-catch block to bypass those pesky
    // syntax errors.
}
```

NO!

The Finally Clause

Part of Java's exception handling model (try-catch-finally)
Used to enclose statements that must always be executed.

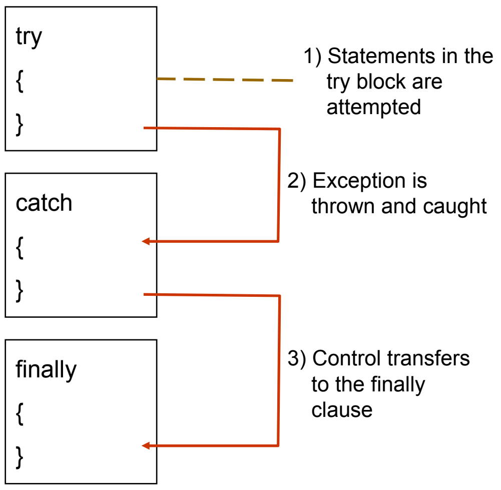
The Finally Clause

```
try
{
}
```

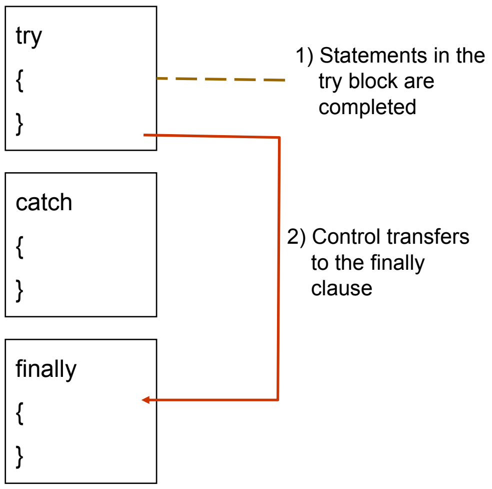
```
catch
{
}
```

```
finally
{
}
```

The Finally Clause: Exception Thrown


try
{
}
catch
{
}
finally
{
}
1) Statements in the try block are attempted
2) Exception is thrown and caught
3) Control transfers to the finally clause

The Finally Clause: No Exception Occurs


try
{
}
catch
{
}
finally
{
}
1) Statements in the try block are completed
2) Control transfers to the finally clause

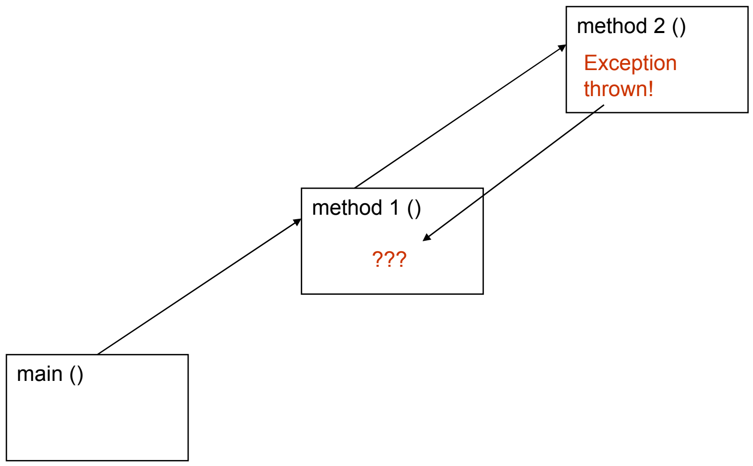
Try-Catch-Finally: An Example

The complete program can be found in the directory:
/home/profs/tamj/233/examples/exceptions/handlingExceptions/secondExample

Try-Catch-Finally: An Example (2)

```
try
{
   BufferedReader br = new BufferedReader(new FileReader("phil"));
   String s = br.readLine();
   while (s != null)
        s = br.readLine();
    return;
}
catch (IOException e)
{
   e.printStackTrace();
    return;
}
finally
{
    System.out.println("<<<Finished reading>>>");
    return;
}
```

When The Caller Can’t Handle The Exceptions



When The Caller Can’t Handle The Exceptions: An Example

The complete program can be found in the directory:
/home/profs/tamj/233/examples/exceptions/handlingExceptions/thirdExample

```
import java.io.*;

class IntermediateIO
{
   public static void main (String [] argv)
   {
      method1 ();
   }
```

When The Caller Can't Handle The Exceptions: An Example (2)

```
public static void method1 ()
{
   try
   {
      method2 ();
      return;
   }
   catch (IOException e)
   {
      System.out.println("IOException thrown while reading input file");
      e.printStackTrace();
      return;
   }
}
```

When The Caller Can't Handle The Exceptions: An Example (3)

```
public static void method2 () throws IOException
{
   BufferedReader br = null;
   String s;

   br = new BufferedReader(new FileReader("phil"));
   s = br.readLine();
   while (s != null)
   {
      System.out.println(s);
      s = br.readLine();
   }
   return;
}
```

Summary

Handling exceptions with the try-catch block

Checked vs. unchecked exceptions

Using the finally clause to guarantee the execution of clean-up statements regardless of whether an exception occurs or not.