Handling Exceptions

Atul Prakash

Reference: Sun's Java tutorials:

http://java.sun.com/docs/books/tutorial/essential/exceptions/

Exceptions

- Exceptions are errors encountered during execution. What to do?
  - Let the program die, or
  - Raise an alert and handle the alert to get around the exception
- Java provides a mechanism to do the latter

Exceptions: try catch block

- Called a try block because you “try” method calls

```
try {
  // Application code goes here…
}
catch(ExceptionType1 e1) {
  // Handle exceptions for Type1
} catch(ExceptionType2 e2) {
  // Handle exceptions for Type2
} catch(ExceptionType3 e3) {
  // Handle exceptions for Type3
}
```

Exception Syntax

```
try {
   // Your code goes here…
 }
 catch (exceptiontype arg) {
   // Your error handling code goes here…
 }
```

```
try {
   int result = x/0;
   System.out.println("result: " + result);
}
catch (ArithmeticException e) {

  System.out.println("caught an exception: " + e);
  e.printStackTrace();   // to print out the stack

}
```

http://java.sun.com/j2se/1.4.2/docs/api/java/lang/ArithmeticException.html

Common Standard Exceptions

- Assertion violations: AssertionError
- Arithmetic problems: ArithmeticException
- File I/O:
  - FileNotFoundException
  - IOException

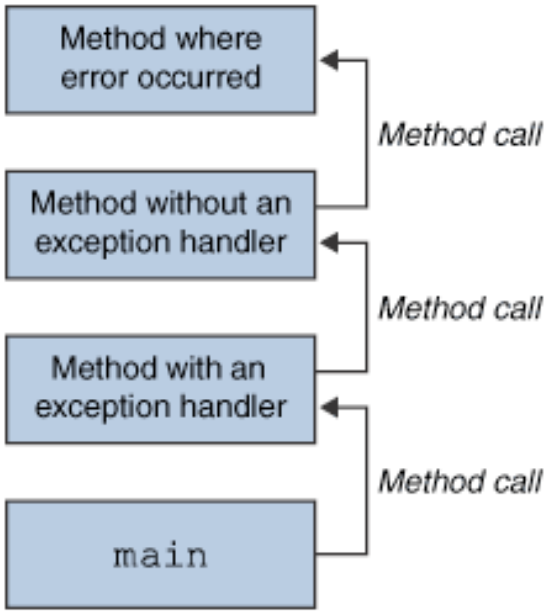
Exception and Error are two catch-all exception types, if you don't know the specific exception

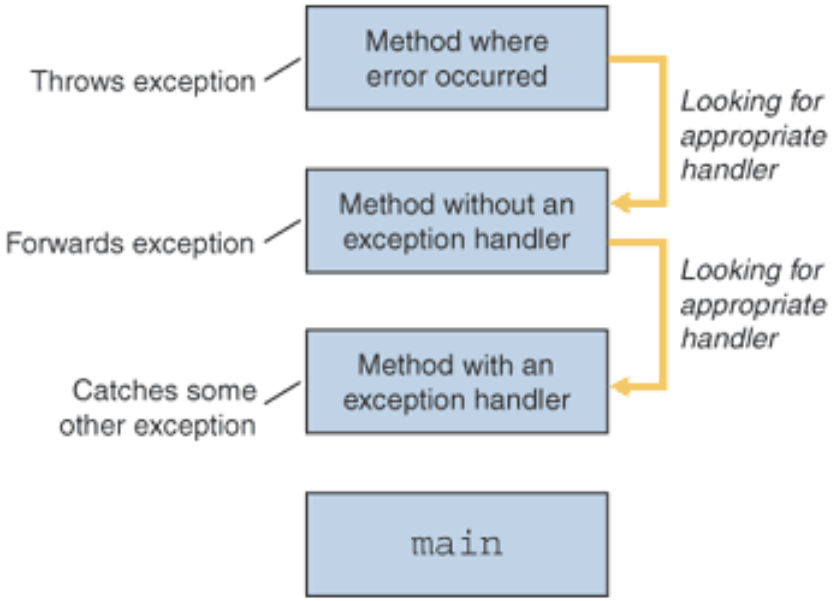
```
// OK
  try {
      ... read y ....
      int result = x/y;

       System.out.println("result: " + result);
   }
   catch (Exception e) {

      e.printStackTrace();

   }


// Preferred: identifies the cause better.
   try {
          ... read y ...
         int result = x/y;

          System.out.println("result: " + result);
   }
   catch (IOException e) {

       System.out.println("trouble reading y: " + e);
        e.printStackTrace();

    }

   catch (ArithmeticException e) {

      System.out.println("trouble with division: " + e);
      e.printStackTrace();

   }
```

Uncaught Exceptions

- If an exception is not caught in a function:
  - The function terminates and the exception passes to the caller
  - This keeps happening till the exception is caught in a caller.
  - If no one catches, the program terminates.

Example



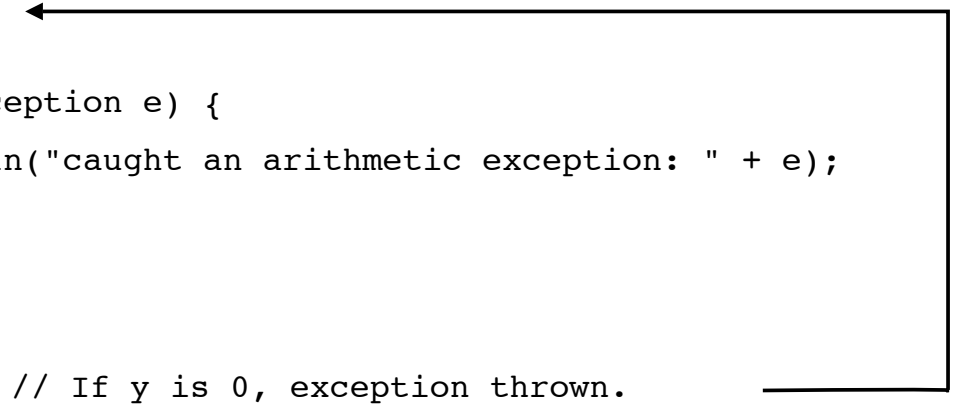


Acknowledgement for the figures: Sun's Java tutorials

Exception Handling

```
void h() {
       int x = 10;
       int y = 0;
       f(x, y);
}

void f(int x, int y) {
        try {
           g(x, y);
        }
        catch (ArithmeticException e) {
           System.out.println("caught an arithmetic exception: " + e);
        }
     }

     void g(int x, int y) {
         int result = x/y;  // If y is 0, exception thrown.
         System.out.println("result: " + result);
      }
```

4. Exception raised in g(x,y).
5. There is a catch clause for the exception.
6. Exception caught and handled



1. Exception raised on x/y.
2. Not caught in g
3. Passed on to the calling function f

Defining your own Exceptions

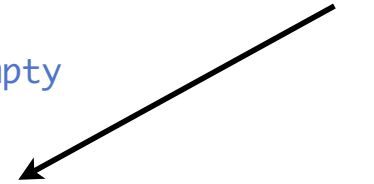
- Define an exception, say EmptyListException, as follows

```
// File: EmptyListException.java

public class EmptyListException extends Exception {

}
```

Don't worry about the extends syntax. For now, just create

Throwing Exceptions

```
public class Largest {

  /**
   * Return the largest element in a list.
   *
   * @param list A list of integers
   * @return The largest number in the given list
 * @throws EmptyListException when the list is empty
   */

  public static int largest(int[] list) throws EmptyListException {
    int index, max=Integer.MIN_VALUE;

    if (list.length == 0) {
      throw new EmptyListException("Empty list");
    }
    for (index = 0; index < list.length; index++) {
      if (list[index] > max) {
        max = list[index];
      }
    }
    return max;
  }
}
```



Handling Exceptions

- Most exceptions, except those that Java calls runtime exceptions, must either be
  - caught by the function using try...catch.
  - or declared at the top of the function using “throws”

Example of catching

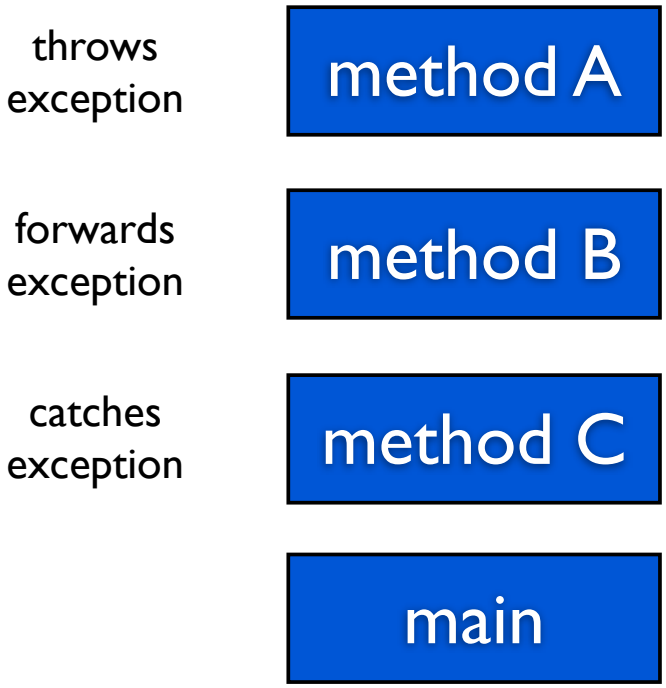
```
public void testEmpty() {
  try {
    Largest.largest(new int[] {});
  } catch (EmptyListException e) {
    ... do something here like print a message ...
  }
```

Example of Not Catching

```
public void testOrder2() throws EmptyListException
{
    int[] arr = new int[3];
    arr[0] = 8;
    arr[1] = 9;
    arr[2] = 7;
    assertEquals(9, Largest.largest(arr));
  }
```

The call to "largest" could result in an exception, as far as compiler is concerned. The exception must be either caught or the function must say that it can throw the exception

Example



Which of the methods must declare in the function header that an exception is thrown?

Answer

- A and B