JAVA EXCEPTIONS

Exception Definition

- An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions
- When an error occurs within a method, the method creates an object and hands it off to the runtime system
- This object is called an *exception object* and contains information about the error, including its type and the state of the program when the error occurred

Throwing Exceptions 1/2

- When a method encounters an abnormal condition that it can't handle itself, it may *throw* an exception
- Throwing an exception is like throwing an SOS signal to indicate there is a problem that can't be handled where it occurred
- After that you “hope” that this signal will be caught and the problem will be dealt with!..

Throwing Exceptions 2/2

- In general, code you write should throw only exceptions
- In addition to throwing Throwable objects (whose classes are declared in java.lang) you can throw objects of your own design
- To create your own class of throwable objects, you need only to declare it as a subclass of some member of the Throwable family
- In general, however, the throwable classes you define should extend class: Exception

```java
public class Exception1
{
  public static void main(String[] args)
  {
    String str1 = "Hello Unipi World!";
    System.out.println(str1);
    throw new NullPointerException();
  }
}
```

C:\Windows\system32\cmd.exe

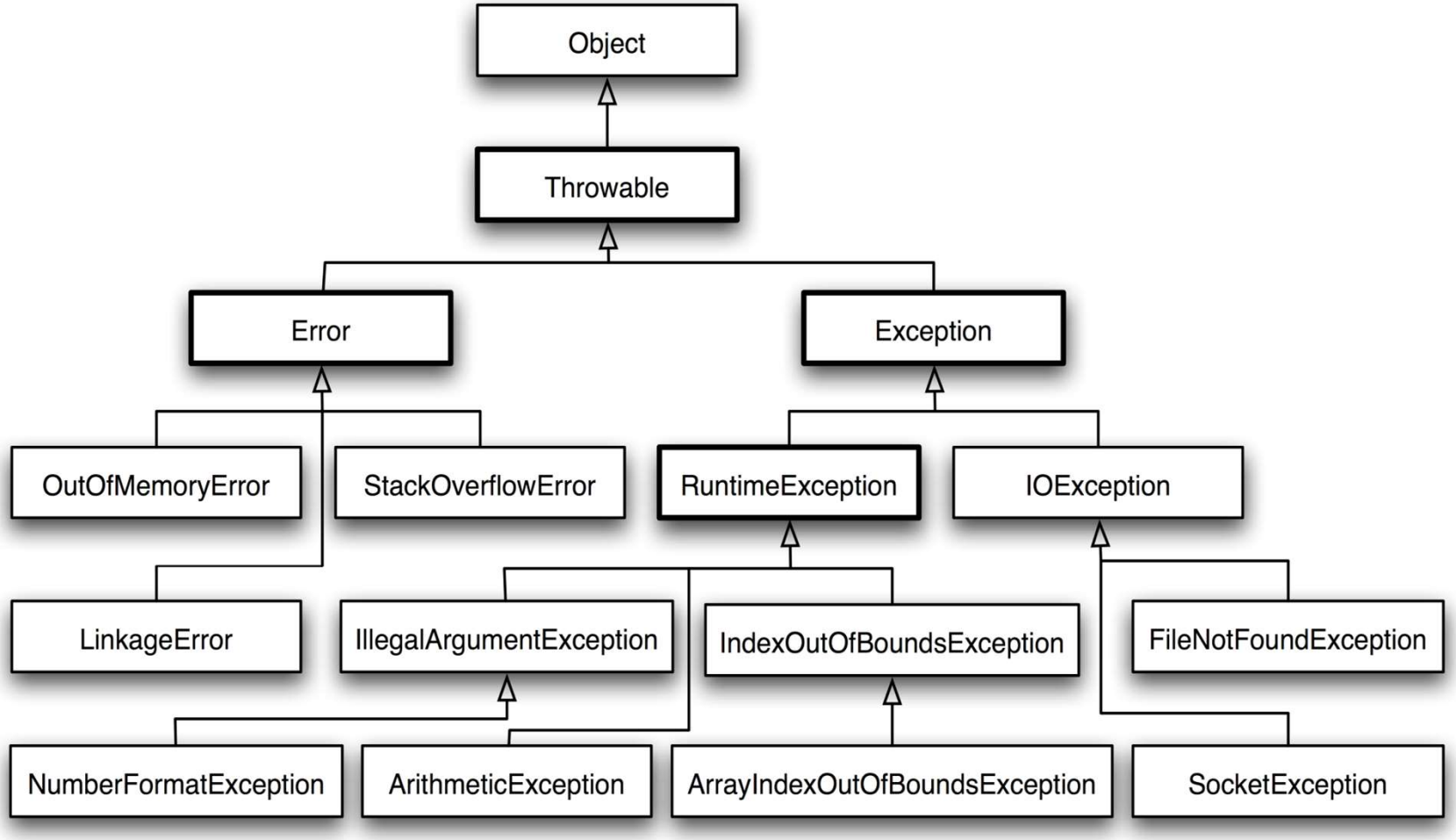
```
E:\myjavaprogs>javac Exception1.java

E:\myjavaprogs>java Exception1
Hello Unipi World!
Exception in thread "main" java.lang.NullPointerException
        at Exception1.main(Exception1.java:7)

E:\myjavaprogs>_
```

Handling Exceptions

- Exceptions are *caught* by handlers positioned along the thread's method invocation stack
- If the calling method isn't prepared to catch the exception, it throws the exception up to *its* calling method, and so on
- When you program in Java, you must position catchers (the exception handlers) strategically, so your program will catch and handle all exceptions from which you want your program to recover


Object
Throwable
Error
Exception
OutOfMemoryError
StackOverflowError
RuntimeException
IOException
LinkageError
IllegalArgumentException
IndexOutOfBoundsException
FileNotFoundException
NumberFormatException
ArithmeticException
ArrayIndexOutOfBoundsException
SocketException

try - catch

```
try {
    // some code that may throw exception
}
catch (Exception e) {
    e.printStackTrace();
}
```

try - finally

```
try {
    // some code that may throw exception
}
finally {
    // some code to clean up
}
```

try – catch – finally

```
try {
    // some code that may throw exception
}
catch (Exception e) {
    e.printStackTrace();
}
finally {
    // some code to clean up
}
```

Multiple exception handlers

```
try {
// Code that might generate exceptions
} catch(Type1 id1)|{
// Handle exceptions of Type1
} catch(Type2 id2) {
// Handle exceptions of Type2
} catch(Type3 id3) {
// Handle exceptions of Type3
}
```

Examples

```java
import java.util.Scanner;

class Exception2 {
  public static void main(String[] args) {
  int a, b, result;
  Scanner input = new Scanner(System.in);
  System.out.println("Input two integers");
  a = input.nextInt();
  b = input.nextInt();
  result = a / b;
  System.out.println("Result = " + result);
  }
}
```

C:\Windows\system32\cmd.exe

```
E:\myjavaprogs>java Exception2
Input two integers
15
3
Result = 5

E:\myjavaprogs>java Exception2
Input two integers
3
t
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at Exception2.main(Exception2.java:12)

E:\myjavaprogs>_
```

C:\Windows\system32\cmd.exe

```
E:\myjavaprogs>java Exception2
Input two integers
8
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Exception2.main(Exception2.java:14)

E:\myjavaprogs>_
```

```java
import java.util.*;
import java.util.Scanner;
class Exception3 {
  public static void main(String[] args) {
  int a, b, result;
  Scanner input = new Scanner(System.in);
  System.out.println("Input two integers");
      try {
        a = input.nextInt();
        b = input.nextInt();
        result  = a / b;
        System.out.println("Result = " + result);
      }
      catch (ArithmeticException e) {
        System.out.println("Exception caught: Division by zero!");
      }
      catch (InputMismatchException e) {
        System.out.println("Exception caught: No integer entered!");
      }
  }
}
```

C:\Windows\system32\cmd.exe

```
E:\myjavaprogs>javac Exception3.java

E:\myjavaprogs>java Exception3
Input two integers
32
John
Exception caught: No integer entered!

E:\myjavaprogs>java Exception3
Input two integers
23
0
Exception caught: Division by zero!

E:\myjavaprogs>_
```

```java
import java.util.*;
import java.util.Scanner;
class Exception4 {
  public static void main(String[] args) {
    dividenumbers();
  }
      public static void dividenumbers(){
      int a, b, result;
      Scanner input = new Scanner(System.in);
      System.out.println("Input two integers");
          try {
            a = input.nextInt();
            b = input.nextInt();
            result  = a / b;
            System.out.println("Result = " + result);
          }
          catch (ArithmeticException e) {
            System.out.println("Exception caught: Division by zero!");
            dividenumbers();
          }
          catch (InputMismatchException e) {
            System.out.println("Exception caught: No integer entered!");
            dividenumbers();
          }
      }
}
```

C:\Windows\system32\cmd.exe

```
E:\myjavaprogs>java Exception4
Input two integers
Maria
Exception caught: No integer entered!
Input two integers
21
0
Exception caught: Division by zero!
Input two integers
100
5
Result = 20

E:\myjavaprogs>_
```