Exceptions:
When something goes wrong

Image from Wikipedia

Conditions that cause exceptions

> Error internal to the Java Virtual Machine

> “Standard” exceptions:
    Divide by zero
    Array index out of bounds
    Etc.

> Manually generated exceptions - using a throw statement

Exception handling fundamentals

> Exceptions are represented by classes

> All exceptions are derived from a class named Throwable

> Two direct subclasses of Throwable:
    Error – exceptions from the Java Virtual Machine
    Exception – handles errors from program activity
        Subclasses of exception handle various errors

> Standard package java.lang defines several exception sub-classes

> Custom exception classes are frequently created

Exception related keywords

Five keywords:

try

catch

throw

throws

finally

Using try and catch

try and catch go together

```
try  {
      // block of code to monitor for errors
}

catch (ExceptType1 exOb) {
      // handler for ExceptType1
}

catch (ExceptType2 exOb) {
      // handler for ExceptType2
}

.
.
.
```

Demonstrate exception handling.

```
class ExcDemo1 {
    public static void main(String args[]) {
    int nums[] = new int[4];

    try {
        System.out.println("Before exception is generated.");
        // Generate an index out-of-bounds exception.
        nums[7] = 10;
        System.out.println("this won't be displayed");
    }

    catch (ArrayIndexOutOfBoundsException exc) {
        // catch the exception
        System.out.println("Index out-of-bounds!");
    }

    System.out.println("After catch statement.");
  }
}
```

```
class ExcDemo1 {
    public static void main(String args[]) {
    int nums[] = new int[4];

    try {
        System.out.println("Before exception is generated.");
        // Generate an index out-of-bounds exception.
        nums[7] = 10;
        System.out.println("this won't be displayed");
    }

    catch (ArrayIndexOutOfBoundsException exc) {
        // catch the exception
        System.out.println("Index out-of-bounds!");
    }

    System.out.println("After catch statement.");
  }
}
```

OUTPUT

```
Before exception is generated.
Index out-of-bounds!
After catch statement
```

```
/* An exception can be generated by one
   method and caught by another. */

class ExcTest {
      // Generate an exception.
    static void genException() {
        int nums[] = new int[4];
        System.out.println("Before exception is generated.");

         // generate an index out-of-bounds exception
        nums[7] = 10;
        System.out.println("this won't be displayed");
  }
}

class ExcDemo2 {
    public static void main(String args[]) {

    try {
          ExcTest.genException();
    }
    catch (ArrayIndexOutOfBoundsException exc) {
          // catch the exception
          System.out.println("Index out-of-bounds!");
    }
    System.out.println("After catch statement.");
  }
}
```

What if an exception is uncaught?

(by your program)

What if an exception is uncaught?
(by your program)

The JVM default handler will catch it.

Then what?

Another example

```
class ExcTypeMismatch {
  public static void main(String args[]) {
      int nums[] = new int[4];

      try {
         System.out.println("Before exception is generated.");

           // generate an index out-of-bounds exception
         nums[7] = 10;
         System.out.println("this won't be displayed");
       }

       catch (ArithmeticException exc) {
            // catch the exception
         System.out.println("Index out-of-bounds!");
       }

      System.out.println("After catch statement.");
  }
}
```

(what happens when this executes?)

Graceful error handling

```
class ExcDemo3 {
  public static void main(String args[]) {
      int numer[] = { 4, 8, 16, 32, 64, 128 };
      int denom[] = { 2, 0, 4, 4, 0, 8 };

      for(int i=0; i<numer.length; i++) {
          try {
             System.out.println(numer[i] + " / " +
                        denom[i] + " is " +
                        numer[i]/denom[i]);
          }
          catch (ArithmeticException exc) {
              // catch the exception
          System.out.println("Can't divide by zero!");
          }
      }
  }
}
```

Output

```
4 / 2 is 2
Can’t divide by zero!
16 / 4 is 4
32 / 4 is 8
Can’t divide by zero!
128 / 8 is 16
```

Multiple catch statements

```
class ExcDemo4 {
  public static void main(String args[]) {
    // Here, numer is longer than denom.
    int numer[] = { 4, 8, 16, 32, 64, 128, 256, 512 };
    int denom[] = { 2, 0, 4, 4, 0, 8 };

    for(int i=0; i<numer.length; i++) {
      try {
        System.out.println(numer[i] + " / " +
                           denom[i] + " is " +
                           numer[i]/denom[i]);
      }
      catch (ArithmeticException exc) {
        // catch the exception
        System.out.println("Can't divide by Zero!");
      }
      catch (ArrayIndexOutOfBoundsException exc) {
        // catch the exception
        System.out.println("No matching element found.");
      }
    }
  }
}
```

Output

```
4 / 2 is 2
Can’t divide by zero!
16 / 4 is 4
32 / 4 is 8
Can’t divide by zero!
128 / 8 is 16
No matching element found.
No matching element found.
```

Subclasses must precede superclasses in catch statements.

```
class ExcDemo5 {
  public static void main(String args[]) {
    // Here, numer is longer than denom.
    int numer[] = { 4, 8, 16, 32, 64, 128, 256, 512 };
    int denom[] = { 2, 0, 4, 4, 0, 8 };

    for(int i=0; i<numer.length; i++) {
      try {
        System.out.println(numer[i] + " / " +
                           denom[i] + " is " +
                           numer[i]/denom[i]);
      }
      catch (ArrayIndexOutOfBoundsException exc) {
        // catch the exception
        System.out.println("No matching element found.");
      }
      catch (Throwable exc) {
        System.out.println("Some exception occurred.");
      }
    }
  }
}
```

```
// Use a nested try block.
class NestTrys {
  public static void main(String args[]) {
    // Here, numer is longer than denom.
    int numer[] = { 4, 8, 16, 32, 64, 128, 256, 512 };
    int denom[] = { 2, 0, 4, 4, 0, 8 };

    try { // outer try
      for(int i=0; i<numer.length; i++) {
        try { // nested try
          System.out.println(numer[i] + " / " +
                             denom[i] + " is " +
                             numer[i]/denom[i]);
        }
        catch (ArithmeticException exc) {
          // catch the exception
          System.out.println("Can't divide by Zero!");
        }
      }
    }
    catch (ArrayIndexOutOfBoundsException exc) {
      // catch the exception
      System.out.println("No matching element found.");
      System.out.println("Fatal error -- program terminated.");
    }
  }
}
```

Manually throw an exception.

```
class ThrowDemo {
  public static void main(String args[]) {
    try {
      System.out.println("Before throw.");
      throw new ArithmeticException();
    }
    catch (ArithmeticException exc) {
      // catch the exception
      System.out.println("Exception caught.");
    }
    System.out.println("After try/catch block.");
  }
}
```

Create an exception.

```
class NonIntResultException extends Exception {
  int n;
  int d;

  NonIntResultException(int i, int j) {
    n = i;
    d = j;
  }

  public String toString() {
    return "Result of " + n + " / " + d +
           " is non-integer.";
  }
}
```

```
class CustomExceptDemo {
  public static void main(String args[]) {

    // Here, numer contains some odd values.
    int numer[] = { 4, 8, 15, 32, 64, 127, 256, 512 };
    int denom[] = { 2, 0, 4, 4, 0, 8 };
    for(int i=0; i<numer.length; i++) {
      try {
        if((numer[i]%2) != 0)
          throw new
            NonIntResultException(numer[i], denom[i]);

        System.out.println(numer[i] + " / " +
                           denom[i] + " is " +
                           numer[i]/denom[i]);
      }
      catch (ArithmeticException exc) {
        // catch the exception
        System.out.println("Can't divide by Zero!");
      }
      catch (ArrayIndexOutOfBoundsException exc) {
        // catch the exception
        System.out.println("No matching element found.");
      }
      catch (NonIntResultException exc) {
        System.out.println(exc);
      }
    }
  }
```