Recitation 2

Exception handling
Exceptions make your code crash

```java
public static void main(String[] args) {
    System.out.println(args[0]);
}
```

```java
public static void main(String[] args) {
    System.out.println(8 / 0);
}
```

```java
public static void main(String[] args) {
    System.out.println(null.toString());
}
```
What could happen without exceptions?

```java
public static double getAverage(double[] b) {
    double sum = 0;
    for (int i = 0; i < b.length; i++) {
        sum += b[i];
    }
    return sum / b.length;
}
```

If `b.length` is 0, what should be returned?
- Infinity
- “special” int - `Integer.MAX_VALUE`
- 2110
- 0
Superclass of exceptions: Throwable

When some sort of exception occurs, an object of class `java.lang.Throwable` (or one of its subclasses) is created and “thrown” --we explain later what “throw” means.

The object has
1. Field to contain an error message
2. Two constructors
3. Function to get the message in the field
Superclass of exceptions: Throwable

Two subclasses of Throwable exist:
Error: For errors from which one can’t recover – don’t “catch” them
Exception: For errors from which a program could potentially recover – it’s ok to “catch” them

```
Exception
  detailMessage
  Throwable
  Throwable(String)
  getMessage
  Exception
  Exception(String)
```

```
Error
  detailMessage
  Throwable
  Throwable(String)
  getMessage
  Error
  Error(String)
```
A Throwable instance: ArithmeticException

There are so many different kinds of exceptions we need to organize them.
Throwing an exception

When an exception is thrown, it is thrown to the place of call, which throws it out further to where that method was called. The code that called main will “catch” the exception and print the error message.

**Method call:** main(new String[] {});

**Console:**

java.lang.AE: / by zero
at Ex.third(Ex.java:11)
at Ex.second(Ex.java:7)
at Ex.main(Ex.java:3)

AE = ArithmeticException
Decoding the output from an exception

```java
public static void main(String[] args) {
    int div = 5/0;
}
```

Exception in thread "main" java.lang.ArithmeticException: / by zero
at Animal.main(Animal.java:2)
Try statement: catching a thrown exception

To execute the try statement:

Execute the try-block. If it finishes without throwing an exception, fine.

If the try-block throws a `MyException` object, catch it (execute the catch block); else throw it out further.

If the exception was caught, execution proceeds to the code `S` following the try-statement.

```java
try {
    code (this is the try-block)
}
catch (MyException ae) {
    code (this is the catch-block)
}
S; (code following the try statement)
```

`ae` is like a parameter. When the catch-block catches a thrown object, `ae` contains the object.
throw keyword: Forcing a crash

Why might I want to crash the application?

parseInt("42") -> 42
parseInt("Sid") -> ???

class Integer {
    /** Parse s as a signed decimal integer.
     * Throw a NumberFormatException
     * if not possible */
    public static int parseInt(String s) {
        if (can’t convert to int){
            throw new NumberFormatException();
        }
    }
}
Demo 1: Read an Integer

- Ask the user to input an `int`
- Try to convert user input to an `int`
- If an exception is thrown, catch it and ask for more input
Create `class Person` with two fields, `name` and `age`. Throw an `IllegalArgumentException` instead of having preconditions when given a `null` name or a non-positive age.
How to write an exception class

/** An instance is an exception */
public class OurException extends Exception {

    /** Constructor: an instance with message m*/
    public OurException(String m) {
        super(m);
    }

    /** Constructor: an instance with no message */
    public OurException() {
        super();
    }
}
**throws clause**

```java
public static void second() {
    ...
    String line = keyboard.readLine();
    ...
}
```

You may get an error message like the yellow one above. In that case, insert a throws clause as shown below.

```java
public static void second() throws IOException {
    ...
    String line = keyboard.readLine();
}
```
throws clause for checked exceptions

/** Class to illustrate exception handling */
public class Ex {

    public static void main() {
        try { second(); } catch (OurException e) {} }

    public static void second() throws OurException { third(); }

    public static void third() throws OurException { throw new OurException("mine"); }

}

If you’re interested in the “controversy”,
http://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html
Demo 2: **Pythagorean Solver**

- Given $a$ and $b$: solve for $c$ in $a^2 + b^2 = c^2$
- Reads in input from keyboard
- Handles any exceptions
Thrown exceptions bubble up the call stack until they are handled by a try-catch block. In the system, the call of method main is in a try-catch statement, and its catch block prints out information about the thrown exception.

Alt-Text: I'm trying to build character but Eclipse is really confusing.