

Recitation 2

Exception handling

Exceptions

Exceptions make your code crash

```

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

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

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

Exceptions

What could happen without exceptions?

```

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?

Exceptions

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

Throwable@x2

detailMessage "/ by zero" Throwable

Throwable() Throwable(String) getMessage()

Exceptions

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@x2

detailMessage "/ by zero" Throwable

Throwable() Throwable(String) getMessage()

Exception() Exception(String) Exception

Error@x2

detailMessage "/ by zero" Throwable

Throwable() Throwable(String) getMessage()

Error() Error(String) Error

Exceptions

A Throwable instance: ArithmeticException

There are so many different kinds of exceptions we need to **organize** them.

ArithmeticException@x2

detailMessage "/ by zero" Throwable

Exception

RuntimeException

ArithmeticException

```

graph TD
    Throwable --> Exception
    Throwable --> Error
    Exception --> RuntimeException
    RuntimeException --> ArithmeticException
    
```

Exceptions

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)
```

```

1 class Ex {
2     static void main(...) {
3         second();
4     }
5     }
6     static void second() {
7         third();
8     }
9     }
10    static void third() {
11        int c = 5/0;
12    }
13 }
```

AE = ArithmeticException

Exceptions

Decoding the output from an exception

```

1 public static void main(String[] args) {
2     int div= 5/0;
3 }
```

Exception that is thrown message

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Animal.main(Animal.java:2)
```

called method line number

Exceptions

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.

```

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

Exceptions

throw keyword: Forcing a crash

Why might I want to crash the application?

```

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();
        }
    }
}
```

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

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

Exceptions

Exercise 3: Illegal Arguments

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.

Exceptions

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();
    }
}

```

Exceptions

throws clause

```

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

```

Unhandled exception type `IOException`

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

```

public static void second() throws IOException {
    ...
    String line= keyboard.readLine();
}

```

Exceptions

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>

Exceptions

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

Key takeaways

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.

```

CLASS BALL EXTENDS THROWABLE {}
CLASS P1 {
    P TARGET;
    P(P TARGET) {
        THIS.TARGET=TARGET;
    }
    VOID AMP(BALL BALL) {
        TRY {
            THROW BALL;
        }
        CATCH(BALL B) {
            TARGET.AMP(B);
        }
    }
    PUBLIC STATIC VOID MAIN(STRING[] ARGS) {
        P PARENT = NEW P(0);
        P CHILD = NEW P(PARENT);
        PARENT.TARGET = CHILD;
        PARENT.AMP(NEW BALL());
    }
}

```

<http://vkcod.com/1188/>

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