

# Parallel Programming

## Recitation Session 1

Thomas Weibel <weibelt@ethz.ch>

Laboratory for Software Technology,  
Swiss Federal Institute of Technology Zürich

March 4, 2010

# Administration

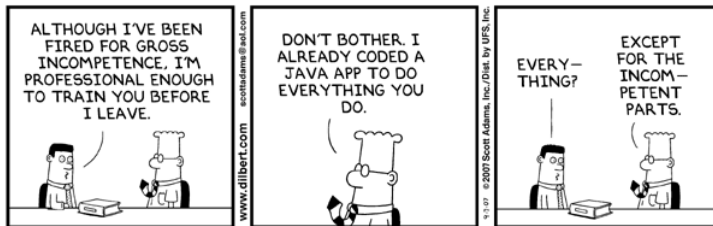
- Contact me if you feel lost: [weibelt@ethz.ch](mailto:weibelt@ethz.ch)
- Get the slides:  
<http://n.ethz.ch/~weibelt/download/parprog/>
- Homework is optional, everybody gets a “testat”
- Past experience: students who do the homework have a good chance to pass the exam



Source: <http://www.asiamex.com>

# Executive Summary

- Solution to the last assignment
- Exceptions in Java
  - Quick overview
  - Quiz
- Hints for the next assignment



© Scott Adams, Inc./Dist. by UFS, Inc.

# Outline

**1 Last Assignment**

2 Exceptions

3 Exceptions Quiz

4 New Assignment

# Solution

```
class Solution {  
    public static void main(String[] args) {  
        int i;  
        int tmp;  
  
        /* iterate through the argument vector */  
        for (i = 0; i < args.length; i++) {  
            /* convert to int and increase */  
            tmp = Integer.parseInt(args[i]) + 1;  
  
            /* print out the result */  
            System.out.println(tmp);  
        }  
    }  
}
```

# Alternative Solution

```
class AlternativeSolution {  
    public static void main(String[] args) {  
        /* iterate through the argument vector */  
        for (String arg : args) {  
            System.out.println(  
                Integer.parseInt(arg) + 1  
            );  
        }  
    }  
}
```

# Outline

1 Last Assignment

**2 Exceptions**

3 Exceptions Quiz

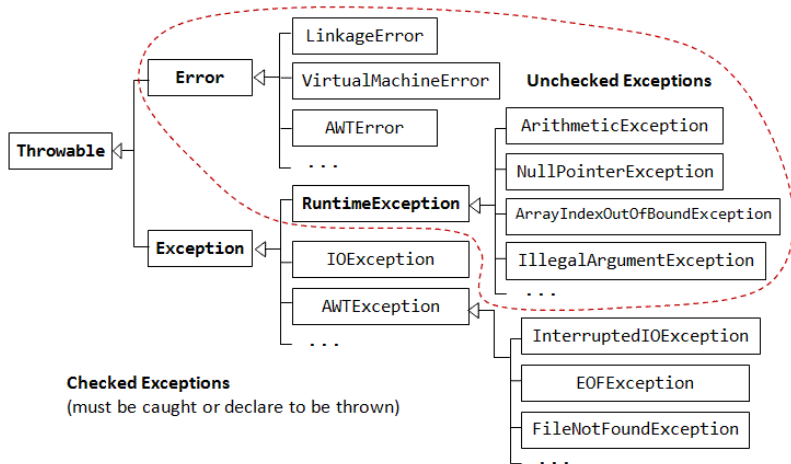
4 New Assignment

# Exceptions

- Exceptional event
- If a method encounters an exception, it creates an “exception object” and hands it off to the runtime system for handling
- Java knows 3 types of exceptions
  - Checked exceptions: Exceptions a program wants to recover from, eg. open a non-existing file (user input error)
  - Errors: Exceptions beyond the program's control, eg. hard disk error (block cannot be read)
  - Runtime exceptions: Violation of logic of a program, eg. null pointer access



# Exception Types



# Throwing an Exception

- Handing off the exception object is called throwing an exception
- Java contains the `throw` clause

```
Exception e = new Exception();  
throw (e);
```

# Try-Catch-Finally

Code that can throw exceptions must be enclosed in a `try` statement:

```
try {  
    // statements  
}  
catch (ExceptionType1 name) {  
    // handler for exceptions of type1  
}  
catch (ExceptionType2 name) {  
    // handler for exceptions of type2  
}  
finally {  
    // statements to always execute  
}
```

Alternative: Announce a throw in the enclosing method (see later)

# Catching an Exception

- To handle an exception it must be “caught”
- Java has a `catch` clause that will catch exceptions of a certain type (class):

```
catch (<ExceptionType> <name>)
```

- If you catch type X you also catch all subtypes of X.

# Finally

- After the `catch` block has finished you may want to “clean” up the state:

```
try {  
    ...  
}  
finally {  
    if (db != null && db.isConnected())  
        db.close();  
    else  
        System.out.println("Not connected");  
}
```

- The `finally` block will always be executed before the try statement completes

# Announcing Throws

```
class Foo {  
    ...  
    void bar() throws FooBarException {  
        ...  
        throw(new FooBarException());  
    }  
}
```

- If a method can throw an exception, you must indicate that there could be throws
- Compiler needs to prepare for possible throw

# Example

```
class Example {  
    public static void main(String[] args) {  
        for (String arg : args) {  
            try {  
                int tmp = Integer.parseInt(arg);  
                if (tmp < 0)  
                    throw(new MyException("< 0"));  
                System.out.println(tmp);  
            }  
            catch (MyException e) {  
                System.out.println(e.getMessage());  
            }  
        }  
    }  
}
```

## Example: Define New Exception Type

New exception types can be defined by extending `Exception`:

```
class MyException extends Exception {  
    MyException(String text) {  
        super(text);  
    }  
}
```



# Outline

1 Last Assignment

2 Exceptions

**3 Exceptions Quiz**

4 New Assignment

## Quiz: Array Index

```
public static void main(String[] args) {  
    int[] array = new int[4];  
    for (int i = 0; i < 5; i++) {  
        try {  
            array[i] = i;  
            System.out.println(array[i]);  
        }  
        catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Invalid index "  
                               + i + "\n");  
        }  
    }  
}
```

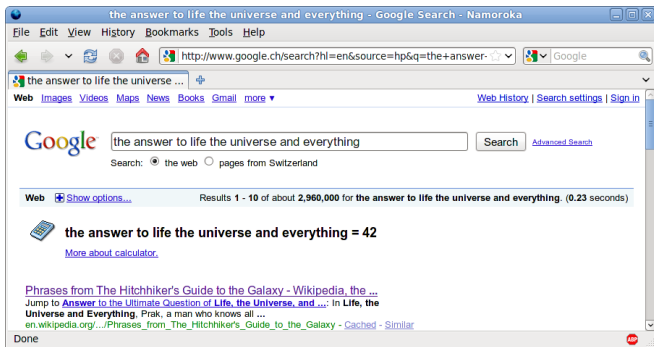
# Quiz: Catch Order

```
public static void main(String[] args) {  
    try {  
        throw new  
            InputMismatchException("Foobar!");  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    } catch (InputMismatchException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

# Quiz: Finally

```
public static void main(String[] args)
    throws Exception {
    try {
        throw new
            Exception("I can has exception!");
    }
    catch (Exception e) {
        System.out.println(e.getMessage());
        throw new Exception("Oh noes!");
    }
    finally {
        System.out.println("Finally!");
    }
}
```

# Quiz: Division by Zero



```
public static void main(String[] args) {  
    int theAnswer = 42;  
    System.out.println(theAnswer / 0);  
}
```

# Outline

1 Last Assignment

2 Exceptions

3 Exceptions Quiz

**4 New Assignment**

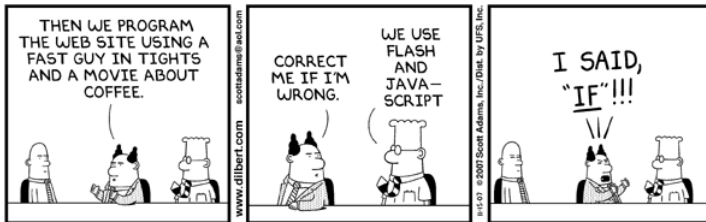
# Hints for the New Assignment

- Separate process method to convert and increment
- Define a new exception class
- Throw exception in case of a negative number
- Catch exception and print the message

```
class NegValException extends Exception {  
    public NegativeValueException() {  
        super("Negative value -- bailing out.");  
    }  
}
```

# Summary

- If a method encounters an exception, it creates an “exception object” and hands it off to the runtime system for handling
- Java knows 3 kind of exceptions: checked exceptions, errors, and runtime exceptions
- Use `throw` to throw exceptions
- `try/catch/finally` can be used to handle exceptions



© Scott Adams, Inc./Dist. by UFS, Inc.