

# Exceptions

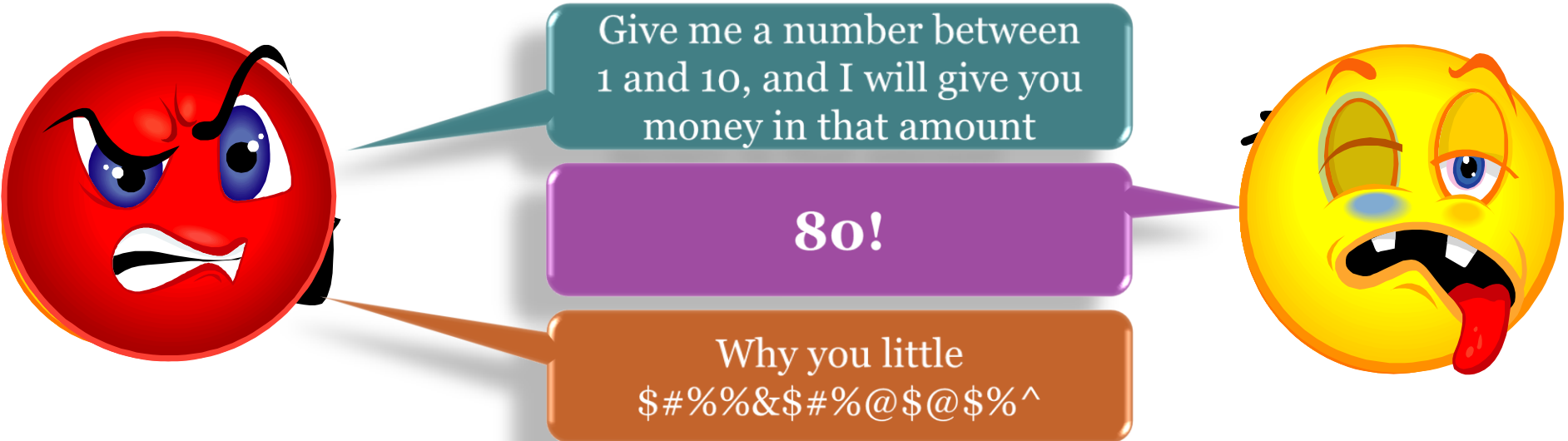
Author: Boaz Kantor

The Interdisciplinary Center, Herzliya

Introduction to Computer Science

Winter 2009-10 Semester

# Why Exceptions?



"An exception is an event that occurs during the execution of a program, that disrupts the normal flow of instructions"

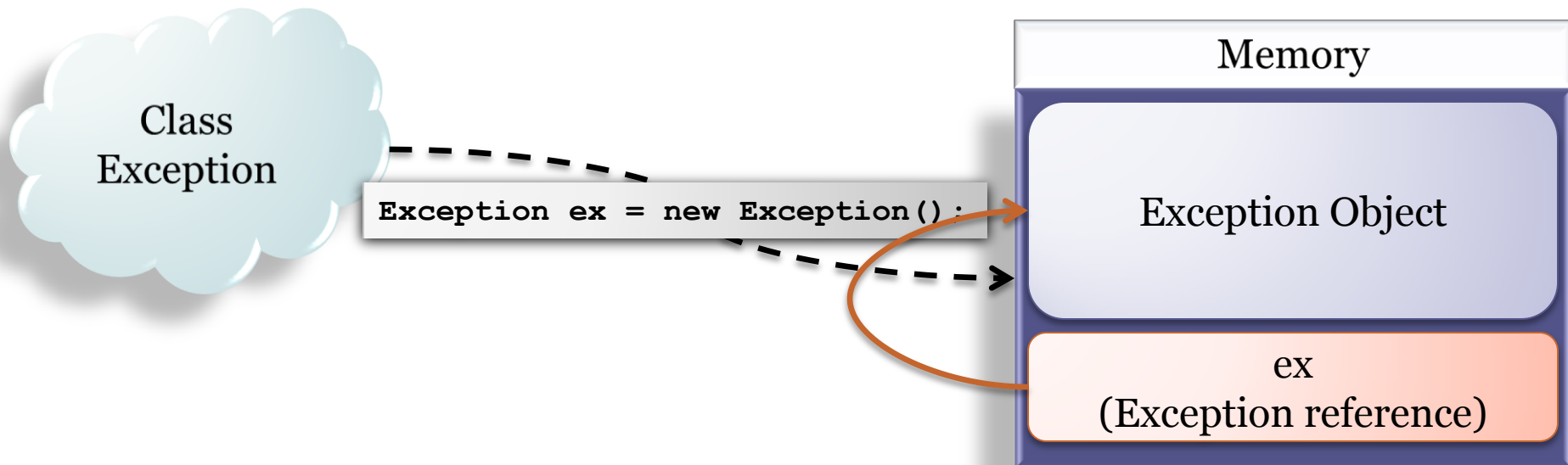
# Why Not Return Errors?

- Return values are **not meant** for that
- Requires program-wide standardization of ERROR variables
- May be ignored
- Do not contain much of information
- Not neat, not systematic!

```
public static final int ERROR_NEGATIVE_NUMBER = -1;
public double squareRoot(int number) {
    if (number < 0) {
        return (double)ERROR_NEGATIVE_NUMBER;
    }
    return Math.sqrt(number);
}
```

# What Is an Exception?

- A class:
  - Can be instantiated
  - Has fields and methods
  - You can write your own exceptions!



# Two Players

- **Calling method**

- Catches it
- Handles it
- Ignores it

“Calculate  
squareRoot(-2)  
please”

- **Throwing method**

- Declares/documents it
- Throws it

Are you nuts? Catch!

Calling Method

```
public static void main(String[] args)
```

(Exception)



Throwing Method

```
public double squareRoot(int)
```

# Two Types

- Runtime exceptions
  - `RuntimeException`
  - Does not need to be declared in throwing method
  - Does not need to be caught in calling method
  - Due to that, used too often
- Non-runtime exceptions
  - `InstantiationException`, `ParseException`, `PrintException`, ..
  - **Must be declared** in throwing method
  - **Must be caught or declared** in calling method

# Throwing It (by throwing method)

- Similar to 'return' (same same but different)
- Throwing:
  1. Instantiate an Exception  
`Exception ex = new Exception();`
  2. Add information to the object (usually in constructor)  
`Exception ex = new Exception(number + " is negative");`
  3. Throw it using 'throw'  
`throw ex;`
- Immediately returns to the calling method
  - (except when 'finally' block exists)

```
public double squareRoot(int number) {  
    if (number < 0) {  
        throw new Exception(number + " is negative!");  
    }  
}
```

# Declaring It (by throwing method)

- Similar to 'return' documentation
- Either:
  - Document (javadoc)
  - Declare
  - (or both)
- If non-runtime exception = **MUST** declare!

```
/**  
 * @throws Exception  
 */  
public double squareRoot(int number) {
```

- Or:

```
public double squareRoot(int number) throws Exception {
```

# Catching It (by calling method)

- Similar to assigning return value
- How:
  1. Surround your method with 'try' block
  2. Catch in a 'catch' block

```
public static void main(String[] args) {  
    try {  
        double root = squareRoot(-2);  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
}
```

# Ignoring It (by calling method)

- Similar to assigning a returned value:
  - Code may be not surrounded with a `'try..catch'` block
- Will be re-thrown
  - As if the calling method was throwing it
- If non-runtime exception:
  - Can't ignore!
  - Either:
    - Surround with `try..catch`
    - Declare it

# Exceptions vs. Returning Errors, throwing method

## Exceptions

```
public double squareRoot(int number) throws Exception {  
    if (number < 0) {  
        throw new Exception(number + " is negative!");  
    }  
}
```

## Returning Errors

```
public static final int ERROR_NEGATIVE_NUMBER = -1;  
public double squareRoot(int number) {  
    if (number < 0) {  
        return (double)ERROR_NEGATIVE_NUMBER;  
    }  
}
```

# Exceptions vs. Returning Errors, calling method

## Exceptions

```
try {  
    double root = squareRoot(-2);  
} catch (Exception e) {  
    System.out.println(e.getMessage());  
}
```

## Returning Errors

```
double root = squareRoot(-2);  
if (root == ERROR_NEGATIVE_NUMBER) {  
    System.out.println("Error: negative number");  
}
```

# Catching Several Exceptions

- A method may throw several types of exceptions
- The calling method can handle each differently

```
try {  
    // call the throwing method  
} catch (ExceptionType1 e1) {  
    // handle this  
} catch (ExceptionType2 e2) {  
    // handle that  
} finally {  
    // finalize stuff  
}
```

# Finally!

- A block
- **Always** executes when the 'try' block exits

```
try {  
    // call the throwing method  
} catch (ExceptionType1 e1) {  
    // handle this  
} catch (ExceptionType2 e2) {  
    // handle that  
} finally {  
    // finalize stuff  
}
```

# Summarizing Comparison, throwing method

## Returning Errors

```
public static final int ERROR_NEGATIVE = -1;
public static final int ERROR_OUTRANGE = -2;
public static final int MAX_NUMBER = 99;
public double squareRoot(int number) {
    if (number < 0) {
        return ERROR_NEGATIVE;
    }
    if (number > MAX_NUMBER) {
        return ERROR_OUTRANGE;
    }
    return Math.sqrt(number);
}
```

## Exceptions

```
public static final int MAX_NUMBER = 99;
public double squareRoot(int number) {
    if (number < 0) {
        throw new NegativeException();
    }
    if (number > MAX_NUMBER) {
        throw new OutOfRangeException();
    }
    return Math.sqrt(number);
}
```

# Summarizing Comparison, calling method

## Returning Errors

```
public static void main(String[] args) {
    double root = squareRoot(-2);
    double lastResult;
    switch (root) {
        case ERROR_NEGATIVE:
            // handle this
            lastResult = root;
            break;
        case ERROR_OUTRANGE:
            // handle that
            lastResult = root;
            break;
        default:
            print("root = " + root);
            lastResult = root;
            break;
    }
}
```

## Exceptions

```
public static void main(String[] args) {
    double root;
    try {
        root = squareRoot(-2);
    } catch (NegativeException ne) {
        // handle this
    } catch (OutOfRangeException oore) {
        // handle that
    } finally {
        lastResult = root;
    }
    print("root = " + root);
}
```

# Finally!

<http://java.sun.com/docs/books/tutorial/essential/exceptions>

## Exceptions

Author: Boaz Kantor

The Interdisciplinary Center, Herzliya

Introduction to Computer Science

Winter 2009-10 Semester