

# COS 212 - Java Crash Course

Evert Geldenhuys

University of Pretoria

12 February 2019

# Table of Contents

- 1 Hello World
- 2 Makefile
- 3 Data Types
- 4 Operators
- 5 Decision Statements
- 6 Loops
- 7 Exception Handling
- 8 Generic Types

# Hello World

```
/* HelloWorld.java */  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

## Compile:

```
javac *.java
```

## Run:

```
java HelloWorld
```

## Outputs:

```
Hello, World!
```

```
default:
    javac *.java
run:
    java Main
clean:
    rm *.class
```

## Compile:

make

## Run:

make run

# Primitive Data Types

Type	Size	Range
boolean	depends	true, false
char	2 bytes	Unicode characters
byte	1 byte	[-128, 127]
short	2 bytes	[-32768, 32767]
int	4 bytes	[-2147483648, 2147483647]
long	8 bytes	[-9223372036854775808, 9223372036854775807]
float	4 bytes	[-3.4E38, 3.4E38]
double	8 bytes	[-1.7E308, 1.7E308]

1

- Primitive Data Types are not objects, they only contain primitive values

---

<sup>1</sup>Adam Drozdek. *Data structures and algorithms in Java*. CENAGE Learning, 2013.

# Reference Data Types

Primitive type	Wrapper class
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

2

- Reference Data types are Object wrappers around the Primitive data types

---

<sup>2</sup>URL:

<https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html>.

# Reference Data Types

- The Java compiler will automatically convert between primitive and reference types

```
public class ReferenceDataTypes {
    public static void main(String[] args) {
        int five = 5;
        Integer six = 6;
        String result = sum(five, six);

        System.out.println(result); // 11
    }

    public static String sum(Integer a, int b) {
        Integer result = a + b;
        return result.toString();
    }
}
```

```
// String creation using string literal
String hello = "Hello";
String world = "World";

// String concatenation
String helloWorld = hello + ", " + world + "!";

// Outputs: Hello, World!
System.out.println(helloWorld);
3
```

---

<sup>3</sup>URL: <https://docs.oracle.com/javase/tutorial/java/data/strings.html>.



Operators	Precedence
postfix	expr++ expr-
unary	++expr -expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
relational	< > <= >= instanceof
equality	== !=
logical AND	&&
logical OR	
ternary	? :
assignment	= += -= *= /= %=

<sup>4</sup> (Excluding bitwise operators)

---

<sup>4</sup>URL: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>.

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>.

# Decision Statements

```
if (condition) {  
    // do something  
} else { // Optional else  
    // do something else  
}
```

```
// Ternary Operator  
condition ? do-if-true : do-if-false
```

```
switch (integerExpression) {  
    case: value1: block1; break;  
    ...  
    case: valueN: blockN; break;  
    default: defaultBlock;  
}
```

5

---

<sup>5</sup>Adam Drozdek. *Data structures and algorithms in Java*. CENAGE Learning, 2013.

# Loops

```
while (condition) {  
    // do something  
}
```

```
do {  
    // do something  
} while (condition);
```

```
for (initialization; condition; increment) {  
    // do something  
}
```

6

---

<sup>6</sup>Adam Drozdek. *Data structures and algorithms in Java*. CENAGE Learning, 2013.

# Exception Handling

```
Integer x = null; // x is a reference so it can be null

try {
    // Without the try-catch the program would crash here
    x = Integer.parseInt("nope");
} catch (NumberFormatException exception) {
    System.out.println("Could not convert string to int");
}

// Check if the string was successfully parsed
if (x != null) {
    System.out.println(x);
}
```

# Generic Types

- Often you want to work with data in a generic way
- For example code that will work for Integers and Doubles
- Can take advantage of the fact that everything in Java is an Object

```
public class Box {  
    private Object object;  
  
    public void set(Object object) { this.object = object; }  
    public Object get() { return object; }  
}
```

7

---

<sup>7</sup>URL:

<https://docs.oracle.com/javase/tutorial/java/generics/types.html>.

# Generic Types

```
public class Box {  
    private Object object;  
  
    public void set(Object object) { this.object = object; }  
    public Object get() { return object; }  
}
```

8

- However, cannot determine at compile time if the Box class is used correctly
- Code can be written to expect an Integer and a String at a different place which may cause an error

---

<sup>8</sup>URL:

<https://docs.oracle.com/javase/tutorial/java/generics/types.html>.

# Generic Types

```
/**
 * Generic version of the Box class.
 * @param <T> the type of the value being boxed
 */
public class Box<T> {
    // T stands for "Type"
    private T t;

    public void set(T t) { this.t = t; }
    public T get() { return t; }
}
```

9

---

<sup>9</sup>URL:

<https://docs.oracle.com/javase/tutorial/java/generics/types.html>.

# Generic Types

```
public class Box<T> {...}
```

```
// Usage:
```

```
Box<Integer> integerBox = new Box<Integer>();
```

```
// Or compiler can infer the type for Box
```

```
Box<Integer> integerBox = new Box<>();
```

```
// Will only accept type Integer
```

```
integerBox.set(5);
```

```
10
```

---

<sup>10</sup>URL:

<https://docs.oracle.com/javase/tutorial/java/generics/types.html>.








These slides and additional guides will be available at:  
<https://cos212.evert.io/>

Areas not covered:

- Arrays
- ArrayList (Dynamic Arrays)

# References I

-  Adam Drozdek. *Data structures and algorithms in Java*. CENAGE Learning, 2013.
-  URL: <https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html>.
-  URL: <https://docs.oracle.com/javase/tutorial/java/data/strings.html>.
-  URL: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>.
-  URL: <https://docs.oracle.com/javase/tutorial/java/generics/types.html>.