COMP 250

Lecture 4

Java Programming Overview
Compiler, JRE, JDK, IDE
Debugging
Java documentation (API)
Packages

Fri. Jan. 14, 2022
TODO (Done?): learn basic Java syntax

https://www.w3schools.com/java/default.asp

By today, you should have covered all the basics.

I am not expecting you to have mastered these topics by now. That will come with practice...
Yesterday’s Tutorial

weeks 1 & 2 (Sasha)

This folder contains various materials put together by T.A. Sasha to get you started with coding in Java. It includes:

- Installation Guide for JDK and IDE videos
  - For Windows  [https://youtu.be/X6b5-RjbjXKE](https://youtu.be/X6b5-RjbjXKE)
  - For Mac    [https://youtu.be/X6b5-RjbjXKE](https://youtu.be/X6b5-RjbjXKE)
- Sasha’s tutorial from Thursday Jan. 13 (zoom recording) in which he covered:
  - IDE + Tech Setup
  - Expectations for assignments, quizzes, exam
  - How to download and start assignments with Ed / MyCourses / IntelliJ
  - How run tests in your IDE
Brief History of **Java**


- Like C++, Java is object oriented.  **ASIDE:** Java is *easier to use* than C++: Java does not require memory management, and pointers are hidden.

- Java was used in early web browsers to run “Java applets”. Modern web browsers now use JavaScript (unrelated to Java), but Java remains a commonly used language in industry.

- Java was written [James Gosling](#) (“Dr Java”) at Sun Microsystems. Sun was acquired by Oracle (2010). Oracle now maintains/improves Java.
Java Programs

A Java program ("application") is a Java class that has a main method. I am not expecting you to understand the code below yet. We will start talking about classes next week.

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

• High Level (e.g. C, C++, Java, Python,... and hundreds more)

• Assembly language (human readable version of machine code)

• Machine code (binary code that controls the circuits of a computer)

In COMP 273 and ECSE 324, you will learn MIPS assembly language and machine code.
ASIDE  Example:  MIPS Assembly Language

The code below is part of a MIPS assembly language program in COMP 273. Each of the instructions is encoded in 32 bits which provide data and control information for a MIPS CPU (hardware).

```
addi $sp,$sp,-8                    # else, make space for 2 items on the stack
sw $ra, 4($sp)                    # store return address on stack
sw $a0, 0($sp)                    # store argument n on stack
       # (will need it to calculate returned value)
addi $a0, $a0, -1                 # compute argument for next call: n = n-1
jal sumton                        # jump and link to sumton (recursive)
lw $ra, 4($sp)                    # load the return address
lw $a0, 0($sp)                    # load n from the stack
addi $sp, $sp, 8                  # change the stack pointer
add $v0, $a0, $v0                 # add current argument n to $v0
jr $ra                             # return to parent
```
A compiler is a program that translates a higher level language into a lower level language.
Running a C Program (or Fortran, ...)

With C, there is an intermediate step of “object files” which I am leaving out for simplicity.

The physical computer only understands the machine code.
Java Compiler (source $\rightarrow$ byte code)

A Java compiler is a program that translates a Java source file into a Java class file. The latter is byte code.

Java source file: Example.java

class Example{
  ...
}

Java compiler

Java byte code file Example.class

see next slide
Source Code

class Example {
    public static int sumToN(int n){
        int sum = 0;
        for (int k=0; k < n; k++){
            sum = sum + k;
        }
        return sum;
    }
}

Byte Code

Example.class

0: iconst_0
1: istore_1
2: iconst_0
3: istore_2
4: iload_2
5: iload_0
6: if_icmpge 19
9: iload_1
10: iload_2
11: iadd
12: istore_1
13: iinc 2, 1
16: goto 4
19: iload_1
20: ireturn

ASIDE: this slide is for your interest only!

What I’m showing here is like assembly language which you’ll learn in COMP 273. It is a human readable version of Java byte code. The actual byte code is a sequence of coded bytes.

The numbers on left are byte indices where instruction starts (like a line number).

iconst_n, istore_n, iload_n are standard instructions that each have own code word. (Each code word is a byte.)

See here for the Java byte codes (the “instruction set”).
Java has strict rules for syntax. If a program has invalid syntax, then the compiler reports an error.

- did the programmer misspell or forget to declare a variable? ("cannot resolve symbol")
- did the programmer forget a semicolon or bracket? ("expecting ;")
- are the types compatible?
  - e.g. `int j = 1.0;`

If there is a compiler error (a.k.a. “syntax error”) then the compiler does not produce a class file.
Compiler Errors

If there is a **compiler/syntax error**, then the compiler does not produce a .class file. You need to edit your code.
ASIDE: Compilers, Grace Hopper, WICS

High level languages and hence compilers were invented very earlier in CS. One of the pioneers (1950s) was Grace Hopper.

Annual events for women in computer science:

• Grace Hopper Celebration of Women in Computing
• ACM Canadian Celebration of Women in Computing

Many women McGill CS students go to latter conference: see McGill Women in CS (WICS)
Running a Java Program

Java programs (.java files) → Java compiler → Java byte code (.class files) → Java Virtual Machine (JVM) → output of the program

The JVM is a program that runs on your computer. The JVM simulates a specialized Java computer (i.e. “virtual machine”).
Running a Java Program

Java programs (.java files) → Java compiler → Java byte code (.class files) → Java Virtual Machine (JVM)

The JVM interprets the byte code, translating it into machine code instructions that run on the physical computer.

Think of this translation as happening while the program is running (although some of the translation can be done in advance too).

The JVM itself is a program which consists of machine code that runs on your computer.

output of the program
Java Runtime Environment (JRE)

The JRE includes the implementation of the JVM and precompiled standard Java library files.

It will be specific to your operating system & computer hardware.
Unlike C programs which must be recompiled in order to run on different operating systems & hardware, the class files are “portable”: they can run on any computer. This is a key reason why Java has been so popular.
Java Development Kit (JDK)

The JDK includes the Java compiler and debugging tools needed for developing your own code, and the JRE.

You can run programs on your computer with only the JRE. But if you want to write (software development) your own Java programs, you need the JDK.

You can compile from command line with

> javac Example.java
> java Example
Integrated Development Environment (IDE)

Java programs (.java files) → Java compiler → Java byte code (.class files) → Java Virtual Machine (JVM)

Eclipse, IntelliJ IDEA, NetBeans, BlueJay, DrJava, etc IDE’s are applications that are built on top of a JDK. They provide a graphical user interface (GUI) and offer many useful tools for software development.
COMP 250

Lecture 4

Java Programming Overview
Compiler, JRE, JDK, IDE
Debugging
Java documentation (API)
Packages

Fri. Jan. 14, 2022
Debugging: 3 kinds of error

1) compiler/syntax error (discussed earlier)

Java programs (.java files) → Java compiler → Java byte code (.class files) → output of the program

2) Runtime error: the program does not finish executing. i.e. It “crashes”. The IDE will give some information about what happened (what “exception” occurred).

3) Logic error: the program finishes executing but the result is incorrect.
Runtime errors: Java “Exceptions”

Examples

• ArrayIndexOutOfBoundsException

```
double[] x = {7.0, 2.3, 5.0};

System.out.println( x[3] );
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3

• NullPointerException

We will discuss this next week. We need reference types next week.
Debug Mode  (Eclipse demo)

• set breakpoint
• execute a single statement at a time
• step over/into methods
• display variable values
• modify variables
• ...

COMP 250

Lecture 4

Java Programming Overview

Compiler, JRE, JDK, IDE
Debugging
Java documentation (API)
Packages

Fri. Jan. 14, 2022
The Java Application Programming Interface (API) defines all classes in the standard library. You can find the complete list via here. e.g. Math library

```java
public final class Math
extends Object

The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

Unlike some of the numeric methods of class StrictMath, all implementations of the equivalent functions of class Math are not defined to return the bit-for-bit same results. This relaxation permits better-performing implementations where strict reproducibility is not required.

By default many of the Math methods simply call the equivalent method in StrictMath for their implementation. Code generators are encouraged to use platform-specific native libraries or microprocessor instructions, where available, to provide higher-performance implementations of Math methods. Such higher-performance implementations still must conform to the specification for Math.
```
<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static double</td>
<td>abs(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the absolute value of a double value.</td>
</tr>
<tr>
<td>static float</td>
<td>abs(float a)</td>
</tr>
<tr>
<td></td>
<td>Returns the absolute value of a float value.</td>
</tr>
<tr>
<td>static int</td>
<td>abs(int a)</td>
</tr>
<tr>
<td></td>
<td>Returns the absolute value of an int value.</td>
</tr>
<tr>
<td>static long</td>
<td>abs(long a)</td>
</tr>
<tr>
<td></td>
<td>Returns the absolute value of a long value.</td>
</tr>
<tr>
<td>static double</td>
<td>acos(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the arc cosine of a value; the returned angle is in the range 0.0 through π.</td>
</tr>
<tr>
<td>static double</td>
<td>asin(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the arc sine of a value; the returned angle is in the range -π/2 through π/2.</td>
</tr>
<tr>
<td>static double</td>
<td>atan(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the arc tangent of a value; the returned angle is in the range -π/2 through π/2.</td>
</tr>
<tr>
<td>static double</td>
<td>atan2(double y, double x)</td>
</tr>
<tr>
<td></td>
<td>Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).</td>
</tr>
<tr>
<td>static double</td>
<td>cbrt(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the cube root of a double value.</td>
</tr>
<tr>
<td>static double</td>
<td>ceil(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.</td>
</tr>
<tr>
<td>static double</td>
<td>copySign(double magnitude, double sign)</td>
</tr>
<tr>
<td></td>
<td>Returns the first floating-point argument with the sign of the second floating-point argument.</td>
</tr>
<tr>
<td>static float</td>
<td>copySign(float magnitude, float sign)</td>
</tr>
<tr>
<td></td>
<td>Returns the first floating-point argument with the sign of the second floating-point argument.</td>
</tr>
<tr>
<td>static double</td>
<td>cos(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the trigonometric cosine of an angle.</td>
</tr>
<tr>
<td>static double</td>
<td>cosh(double x)</td>
</tr>
<tr>
<td></td>
<td>Returns the hyperbolic cosine of a double value.</td>
</tr>
<tr>
<td>static double</td>
<td>exp(double a)</td>
</tr>
<tr>
<td></td>
<td>Returns the exponential value of a double.</td>
</tr>
</tbody>
</table>
Sample entry

static double abs(double a)

Returns the absolute value of a double value.

It specifies the **name** of the method and the **parameters (number and type)**. The name + types of each parameter define the *method’s signature*.

The return type and other modifiers *(static)* are also specified.

There is also a detailed description of what the method does.
If you are doing *software development* as part of a team, then you may be asked to make documentation for the Java classes that you write and that will be used by others.

Javadoc is a tool for creating a class API in form of an html file. The API is nicely formatted when displayed in a browser.

- In Eclipse, **File > Export**
- Expand **Java**, select **Javadoc**. Then click **Next**.

*Javadoc is not necessary for COMP 250.*
COMP 250

Lecture 4

Java Programming Overview
Compiler, JRE, JDK, IDE
Debugging
Java documentation (API)
Packages

Fri. Jan. 14, 2022
A package is a set of classes. The two on left are examples from the standard Java library. The two on right are examples of my own packages.
We put a `package` statement at the first line of our class definition file. This says which package the class belongs to.

[DEMO THIS IN ECLIPSE]

```
Point2D.java

package demos;

class Point2D{
    :
}
```
Example (Eclipse)

```java
package demos;

class Point2D {
    int x;
    int y;
    static int numberOfPoint2D;

    Point2D(int x, int y) {
        this.x = x;
        this.y = y;
        numberOfPoint2D += 1; // increment
    }
}
```
Example (Eclipse)
Packages and File folders

Packages are organized as folders on your computer file system.

In this Eclipse example, there is a project name ("250") and the .java files are in the directory: 
C:\Users\MichaelLanger\Dropbox\Eclipse\250\src\demos\Point2D.java

```java
package demos;

class Point2D {
    int x;
    int y;
    static int numberOfPoint2D;
}
```
You can have *packages within packages*, corresponding to file folders within file folders. e.g. `C:\Users\YourName\Dropbox\Eclipse\250\src\demos\recursion\TestFactorial.java`

The package name matches the folder path. On the file system, subfolders are indicated with a "slash". *In the package name, a “dot” is used.*
Packages and File Folders

250/src/ contains source code e.g. TestArithmetic.java

250/bin/ contains Java byte code e.g. TestArithmetic.class

For more information on packages and file folders, see here.
Coming up...

Lectures

Mon. Jan 17    arrays

Wed. Jan. 19   objects & classes 1:
               (wrapper classes, strings)

Fri. Jan. 21   objects & classes 2

Homework (TODO)

• w3schools Tutorial   (done)

• Install either Eclipse or IntelliJ.   (done)

• Simple coding exercises   (posted soon)

Assignment 1 to be posted Fri. Jan. 28 (2 weeks).