**Homework 1**

**Programming Languages Principles and Implementation**

**JAVA**

**Name: Angela Bonsol**

**Homework 1 is a comprehensive review of Java. It focuses however on programming languages, rather than only Java itself!**

**Instructions**

* This homework assignment is to be done alone or in a group of 2 students (not 3).
* Your written solution must be integrated in this document. Do not change the order and do not remove text. This part will be submitted through BlackBoard.
* The code will be posted in GitHub. We will cover Git / GitHub this week. You can keep it on your computer until we cover GitHub.

All your code must be available on GitHub under the csS361 repository eventually.

* All Java code must be written and tested in an IDE. We recommend at this time the Eclipse IDE (<http://www.eclipse.org>). All my code for hw1 is written in Eclipse!
* Code that does not compile will be graded as 0.
* In case of problems with this homework, post your questions in Piazza.
* Part 1 - HackerRank: 110 points - Due date: **Monday 9/21**
* Part 2 – 100 pts - Due date: **Monday 9/14**

**Part 1 – HackerRank 110 pts**

A HackerRank test was sent to you. This is part of the homework. You have until 9/21 to do it. You have to do it in one sitting of 3 hours. The time suggested by HackerRank is 69 minutes. Please check the instructions in your email.

You should do Part 2 of this homework before Part 1! You will review Java before doing the homework.

**Part 2 – Java - 100 points**

**Exercise 1: Join Piazza. 5 points if done by 9/9**

We will use Piazza for direct communications. Post your questions there. It is will used only for the communications, nothing else. Everything else will be in BB.

Join here ASAP: <https://piazza.com/pace/fall2020/cs361> with code 4040

Directories are created for organized communications.

You can use Piazza from the web site or from the mobile app. Download the mobile app!

Ask your questions now!

This is my first time to use Piazza. I decided to use it rather than Slack because it implements FERPA and privacy.

**Done**

**Exercise 2: GitHub. 5 pts**

The course has a GitHub repository with the code that we will cover in class: <https://github.com/scharffc/cs361>.

Create a GitHub account (if you do not have one). <https://github.com>.

Create a repository called cs361.

During the course, you will submit your work in GitHub. It will need to be organized. It is up to you but I must not have problems to find your work!

You will learn Git and GitHub in different lectures and assignments.

Post your GitHub cs361 link here: <https://bit.ly/3kKdCxU>

**Done**

**Exercise 3: Lecture notes. 5 pts**

Read the Java lecture notes posted in BlackBoard.

What is the title of slide n? n=1, Angela

“The Object Oriented Paradigm and Java”

n is the number corresponding to the first letter of your first name.

For example, for me, n is 3 as my first name is Christelle.

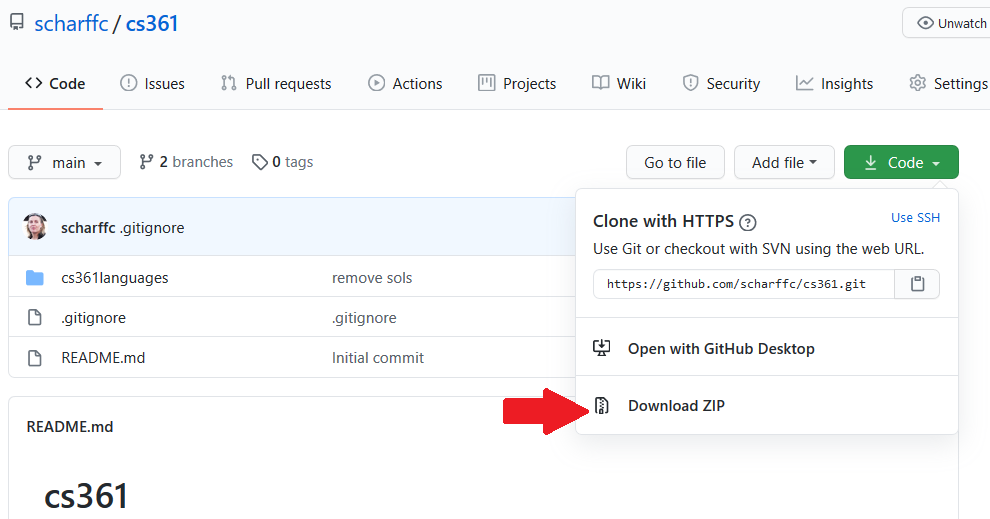
**Done**

**Exercise 4: Setting up. 0 pts**

Download Eclipse for Java Developers (We do not need the Java EE version). Download the latest version or update / upgrade your current version.

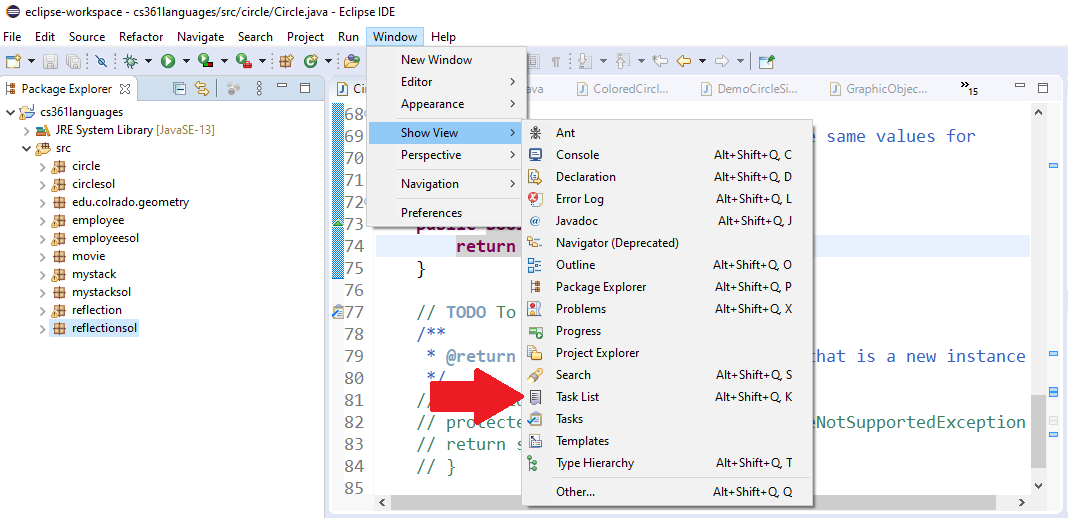
<https://www.eclipse.org/downloads/packages/>

Download the code at <https://github.com/scharffc/cs361>. You can download a zip file.

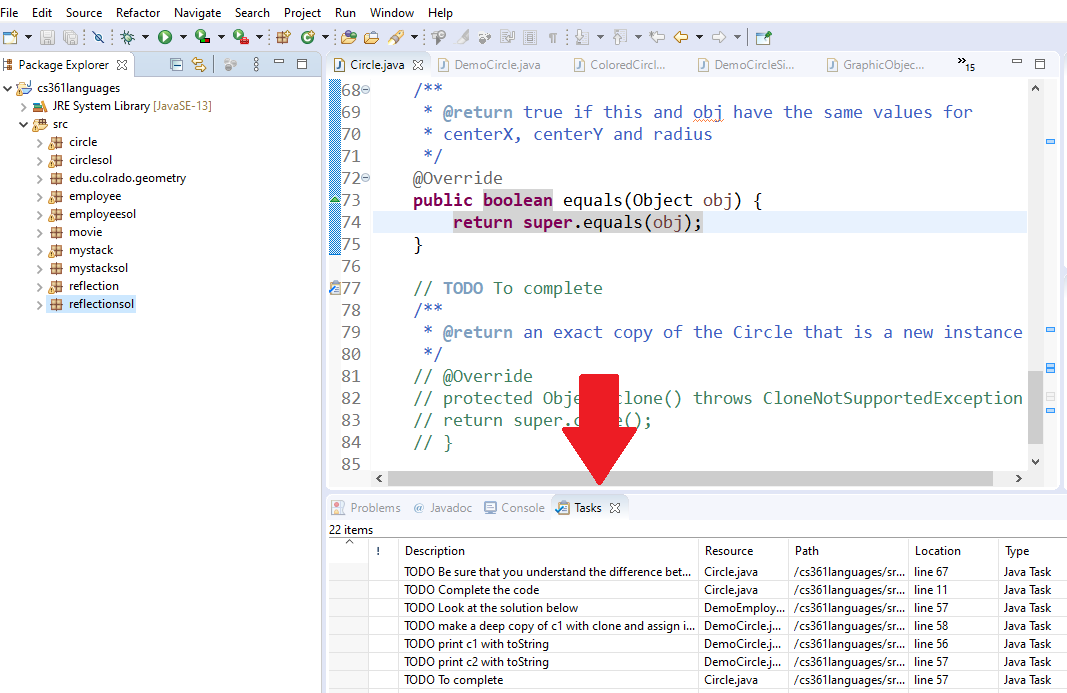


Open the file in Eclipse. The code in cs36languages was written with Eclipse.

In Eclipse, add the View Tasks.



You should now see 22 TODO tasks.



**Done**

**Exercise 5: Do the 22 TODO tasks. 66 pts**

Do not remove the TODO such that I can go through them when correcting your work.

Code is provided to you. In real life, you often have to go through code written by someone else. You will have to read all the code in each package to do the required work.

The code corresponds to the lecture notes. There are explanations in the PDF.

22 TODO tasks may sound a lot but some of them are short!

* Circle package
  + This package will make you review polymorphism and crucial Object methods: toString, clone and equals.
* Employee package
  + This package will make you review polymorphism.
* Stack package
  + This package will make you review generics.
* Reflection package
  + This package will make you practice with Reflection.

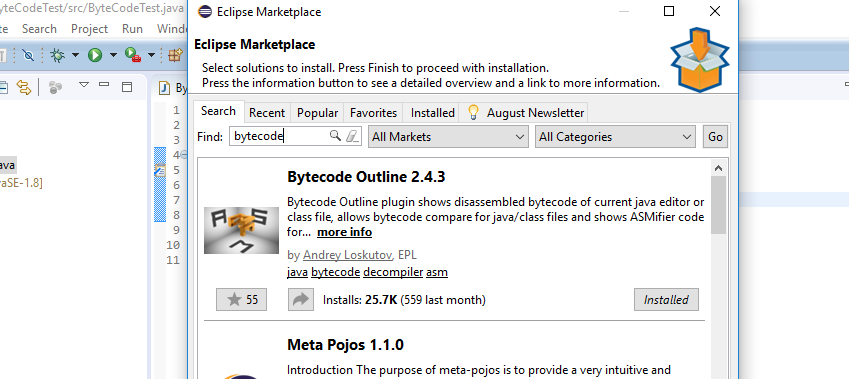
**Done**

**Exercise 6: Java Bytecode – 19 pts**

In the past, I was using an Eclipse plugin such that we look at Java bytecode. The plugin does not work for latest version of Eclipse.

The infos about the plugin are here:

<http://andrei.gmxhome.de/bytecode/index.html>



You may want to install it but it is not required…

You will find below the Java bytecode generated for the following Java code below.

**public** **static** **int** sum\_for(**int** n) {

**int** i = 0, sum = 0;

**for** (i = 0; i <= n; i++) {

sum += i;

}

**return** sum;

}

1 to 39 are the line numbers that you can use to explain the code.

// access flags 0x9 1

public static sum\_for(I)I 2

L0 3

LINENUMBER 4 L0 4

ICONST\_0 5

ISTORE 1 6

L1 7

ICONST\_0 8

ISTORE 2 9

L2 10

LINENUMBER 5 L2 11

ICONST\_0 12

ISTORE 1 13

GOTO L3 14

L4 15

LINENUMBER 6 L4 16

FRAME APPEND [I I] 17

ILOAD 2 18

ILOAD 1 19

IADD 20

ISTORE 2 21

L5 22

LINENUMBER 5 L5 23

IINC 1 1 24

L3 25

FRAME SAME 26

ILOAD 1 27

ILOAD 0 28

IF\_ICMPLE L4 29

L6 30

LINENUMBER 8 L6 31

ILOAD 2 32

IRETURN 33

L7 34

LOCALVARIABLE n I L0 L7 0 35

LOCALVARIABLE i I L1 L7 1 36

LOCALVARIABLE sum I L2 L7 2 37

MAXSTACK = 2 38

MAXLOCALS = 3 39

Use the references to explain the following commands:

* **ILOAD** – loads an int value from a local variable at an unassigned index, then that value is pushed to the operand stack.
* **IINC**- increments the local variable at an unassigned index by a constant.
* **GOTO** – conditional jumps that instructs the system to jump from one method/instruction to another; works similar to a while or for loop in Java.

Explain how this bytecode corresponds to the Java code. In particular, explain how assignments, loops etc. are implemented.

This bytecode very much corresponds to the Java code due to the commands that are used and its similarities for both. The IINC command works similar to Java’s increment(++), in which a variable increments by a constant value. The GOTO command is very similar to the while or loop conditions in Java that makes the system go to a specific place(address) in the instructions or repeat itself when a certain condition is met. IRETURN is similar to the return statement in Java where a value in returned from method, in Bytecode an integer is specifically returned. It can be noticed that if statements are used in Bytecode as conditional statements, similar to java’s if statement declarations. The encapsulation “public” and modifier “static” is also found to be similar in Java, having the same purpose of making a code public and accessible without having to declare any objects in the class yet. Local variables are also used in both to refer to addresses of data that are stored.

**Done**

**References**

* The Java Virtual Machine Specification <https://docs.oracle.com/javase/specs/jvms/se8/html> (Java 8 SE) (2020)
* Java Bytecode Basics <http://www.javaworld.com/javaworld/jw-09-1996/jw-09-bytecodes.html> (1996)
* <http://www.beyondjava.net/blog/java-programmers-guide-java-byte-code/> (2015)