**Homework 1**

**Programming Languages Principles and Implementation**

**JAVA**

**Name: Jason Forrisi**

**Homework 1 will permit you to review Java.**

**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 Classes.
* 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
* Part 2 – 100 pts

**Part 1 – HackerRank 110 pts**

A HackerRank test was sent to you by email. 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: 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 in the management spreadsheet.

**Exercise 2: Lecture notes. 5 pts**

Read the Java lecture notes posted in BlackBoard.

What is the title of slide n/2?

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.

Matthew-> 13-> Slide 6’s title

**(Page 5): Classes, Instances, Instance Variables**

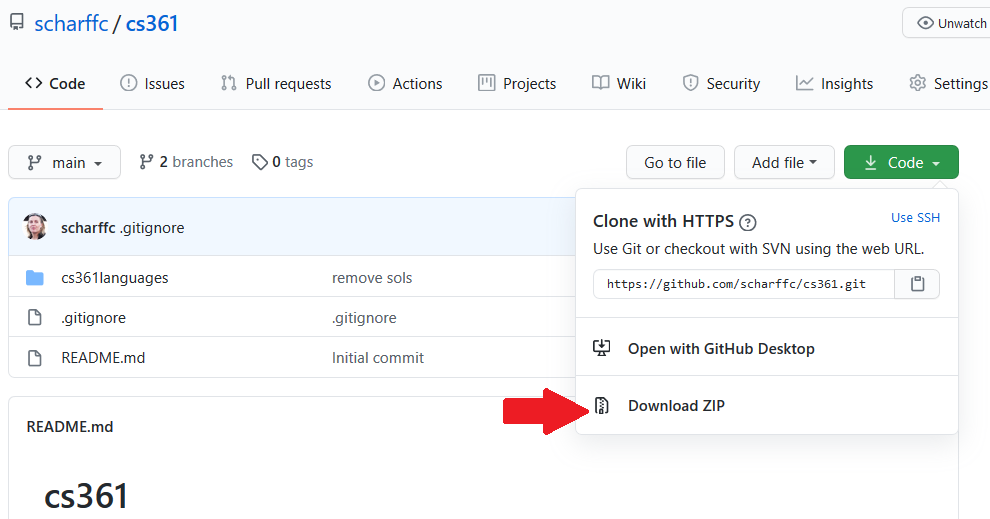
**Exercise 3: Setting up. 0 pts**

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/>

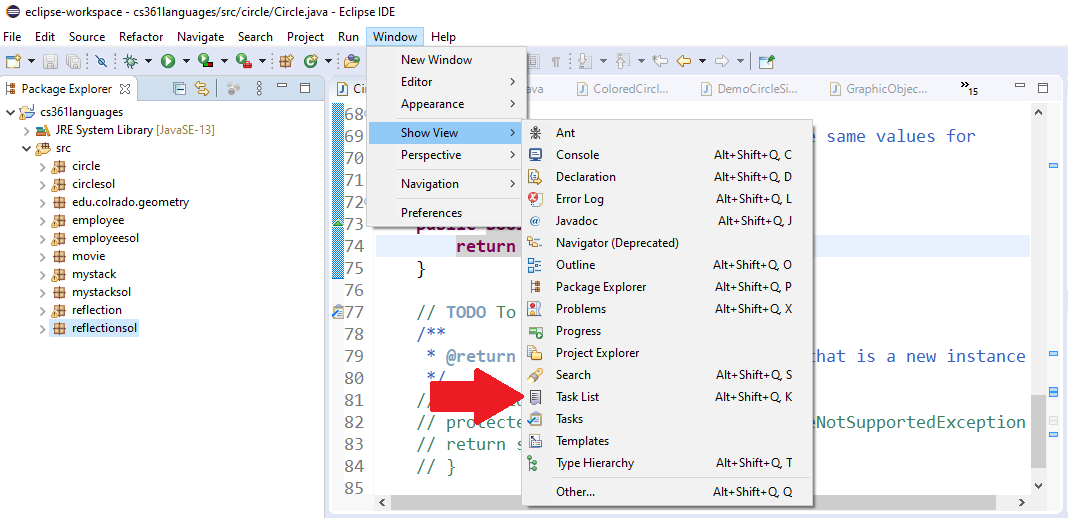
Download the code at <https://github.com/scharffc/cs361>. You can download a zip file.

If you are more comfortable with another IDE it is fine but I may sometimes ask you to have EXACTLY the same organization of files as me for automated grading.

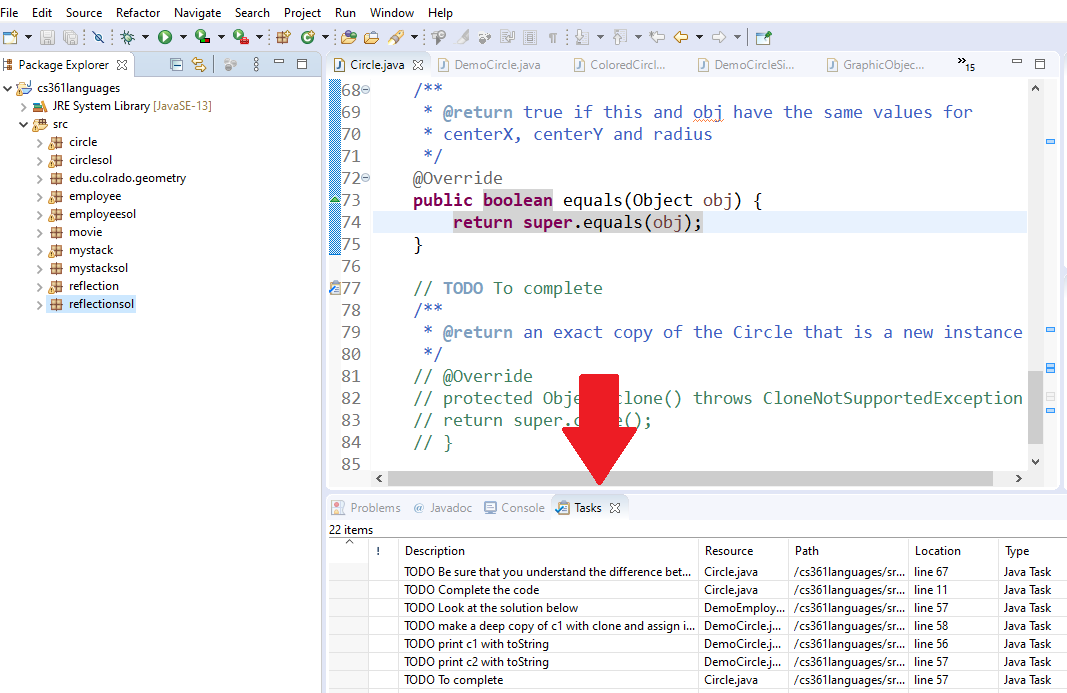


Open the file in Eclipse. The code in cs36languages was written with Eclipse.

In Eclipse, add the View Tasks.



You should now see 22 TODO tasks.



**Exercise 4: 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 th e 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!

Delivery this homework by pushing your code to your GitHub (nothing else is accepted).

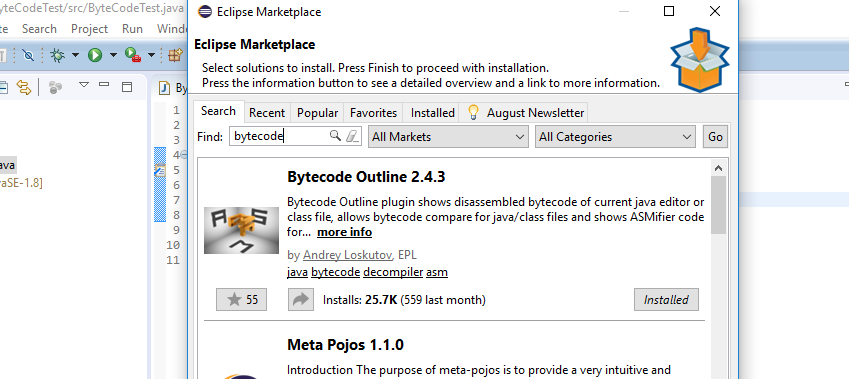
* Circle package
  + This package will make you review polymorphism and crucial Object methods: toString, clone and equals.
* Employee package
  + This package will make you review polymorphism.
* Stack package
  + This package will make you review generics.
* Reflection package
  + This package will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

**Exercise 5: Java Bytecode – 24 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

Explain the following bytecode commands:

* ILOAD
* IINC
* GOTO

ILOAD loads a variable and pushes it to the stack. For example line 18 loads the loads the local variable 2 previous stored in line 9. Then line 19 loads local variable 1. These are both on top of the stack so when IADD runs on line 20, these numbers are popped and added together.

IINC increases the value of a variable. When IINC is called in line 24, it takes local variable 1 and adds 1 to it.

The GOTO command jumps to another point in the code. For example GOTO on 14 jumps to L3 on line 25.

Explain how assignments, loops etc. are implemented in bytecode.

Assignments are implemented in bytecode with the ISTORE and ILOAD commands. ISTORE to store an assignment and ILOAD to call the variable.

Loops are implemented with the GOTO command and conditionals. The GOTO command can jump back to the beginning of the code block if the condition is not met.

Conditionals are implemented using commands such as IF\_ICMPLE, IFNULL or IFNE.

Did you discover anything about the Java language by looking at the bytecode?

You can see the procedural steps looking at the bytecode and learn how the JVM reads the code.

**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)