Homework 2

Jan Schoepp

Pace University

CS611 - Principles of Programming languages

**Homework 2**

**Programming Languages**

**Instructions:**

* Due date: 10/10
* This homework assignment is to be done alone or in a group of 2 students.
* Problems must be done in order.
* You need to fill out this document with your answers. Homeworks with answers only will not be accepted.
* All Java code must be written and tested in the Eclipse IDE (<http://www.eclipse.org>) (or similar).
* Code must be provided in annex and printed directly from Eclipse.
* Code that does not compile will be graded as 0.

All your code must be submitted in Blackboard in a zip file containing all your code for the homework. The name of the file should be: hw1-lastnamefirstname1ststudent-lastnamefirstname2ndstudent.zip.

* Your homework must be well presented and have a cover page. 10 points will be reduced from your grade if you do not do have a cover page.
* The presentation of the hard copy of your homework assignment must contain your name(s).
* In case of problems with this homework, contact me by email [cscharff@pace.edu](mailto:cscharff@pace.edu).
* Grade: 100 points

**Question 1: History of programming languages**

Put the following programming languages on a chronological timeline:

* Fortran – 1957 Johan Backus, IBM, United States
* Lisp – 1958 John McCarthy, Massachusetts Institute of Technology (MIT), United States
* Cobol -1959 Howard Bromberg and others, CODASYL, United States
* ISETL – 1969 Jacob T. Schwartz, New York University, United States
* PASCAL – 1970 Niklaus Wirth, Zürich
* Prolog – 1972 Alain Colmerauer, France
* C – 1972 Dennis Ritchie, Bell Labs, United States
* SML – 1973 (?) Robin Milner, University of Edinburgh, Scotland
* ADA – 1980 Jean Ichbiah, CII Honeywell Bull, United States
* C++ - 1983 Bjarne Stoustrup, Bell Labs, United States
* EIFFEL – 1986 Bertrand Meyer, Eiffel Software, United States
* Perl – 1987 Larry Wall, Unisys, United States
* Python – 1991 Guido van Rossum, Python Software Foundation, United States
* Java – 1995 James Gosling, Sun Microsystems (Oracle Corporation), United States
* Ruby – 1995 Yukihiro Matsumoto, Japan

**On the same timeline,** indicate the name of the designer of the programming language, where it was created (company, national lab, higher education institution etc.), and the country.

**Question 2:**

Consider the following code. Each *draw* method has a number.

public class Circle{

public double center\_x, center\_y;

public double radius;

public void draw() {

// **(1)** method to draw circle on the screen

}

public void draw(Color color) {

// **(2)** method to draw circle on the screen with a

// given color

}

}

public class ColoredCircle extends Circle{

public int color;

public void draw() {

// **(3)** method to draw the colored circle

}

}

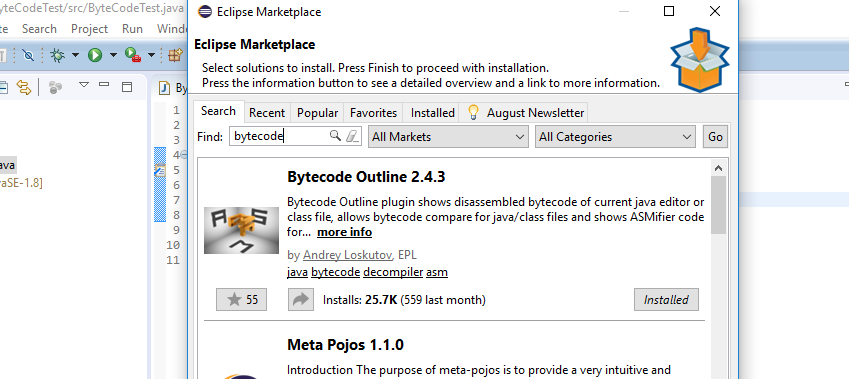
1. Explain polymorphism on the code above.
   * We are defining a new class ColoredCircle, which has all the properties as Circle, but we have the option to add extra functionality to it. In the class we have overridden the draw() method, and we have the option to add something else to this method. The method draw(Color color) can still be called from both classes.
2. c is of type Circle and d is of type ColoredCircle. Can we write c = d;? Why?
   * Yes, because d (ColoredCircle) is an extension of c (Circle), and this will give the features of d to c. c will be able to call (2) and (3).
3. c is of type Circle and d is of type ColoredCircle. Can we write d = c;? Why?
   * Yes, because d (ColoredCircle) is an extension of c (Circle), and this will revert d back to having the features of c. c will be able to call (1) and (2).
4. c is of type Circle and d is of type ColoredCircle. Can we write c = d;? Why? What happens if we execute the code below? What method called *draw* is called? Why?
   * Yes, because d (ColoredCircle) is an extension of c (Circle), and this will give the features of d to c. c will become of type ColoredCircle, and therefore the method draw() number 3 will be called.

c = d;

c.draw();

**Question 3:**

Install the following Eclipse Bytecode Outline plugin from: <http://asm.objectweb.org/eclipse/index.html> or from the Eclipse MarketPlace.



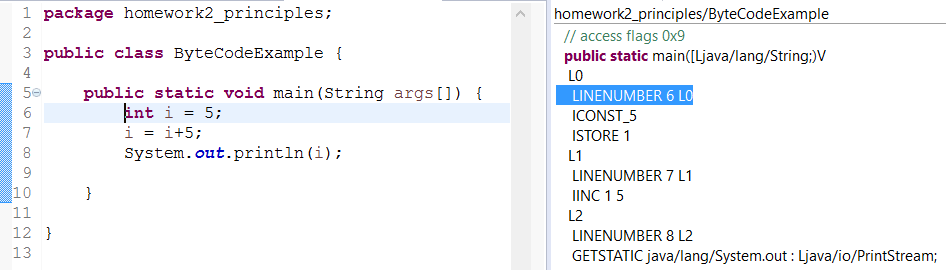
*[Dr. Scharff tested with the new version of Eclipse (Neon) and with Eclipse Marketplace Byte Outline 2.4.3 plugin and it works! ]*

1. What Eclipse version are you using?
   * Eclipse Version: Mars.2 Release (4.5.2)
2. What Java version are you using?
   * Java 8 update 71
3. What is the Bytecode generated by the following statements?

int i = 5;

i = i+5;

Explain the syntax of the Bytecode. Provide a screenshot to support your work.



* The bytecode, as illustrated on the right, is a step by step breakdown of what is going on.   
  **L0** – *Refers to the first action in the main method.*  
  **LINENUMBER 6 L0** – *Refers to line number 6 in the code, and the first action in the code.*  
  **ICONST\_5** – *declares a new constant 5 to be stored into a variable.*  
  **ISTORE 1** – *Declares that the constant declared in the line above is to be stored in the variable 1.***L1** – *Refers to the second action in the main method.***LINENUMBER 7 L1** - *Refers to line number 7 in the code, and the second action in the code.***IINC 1 5** – *Declares that the variable 1 should be incremented with a constant of 5.*

1. Compare the Bytecode generated by the 2 functions below and write down your conclusions.

Provide screenshots to support your work.

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

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

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

sum += i;

}

**return** sum;

}

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

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

**while** (i <= n) {

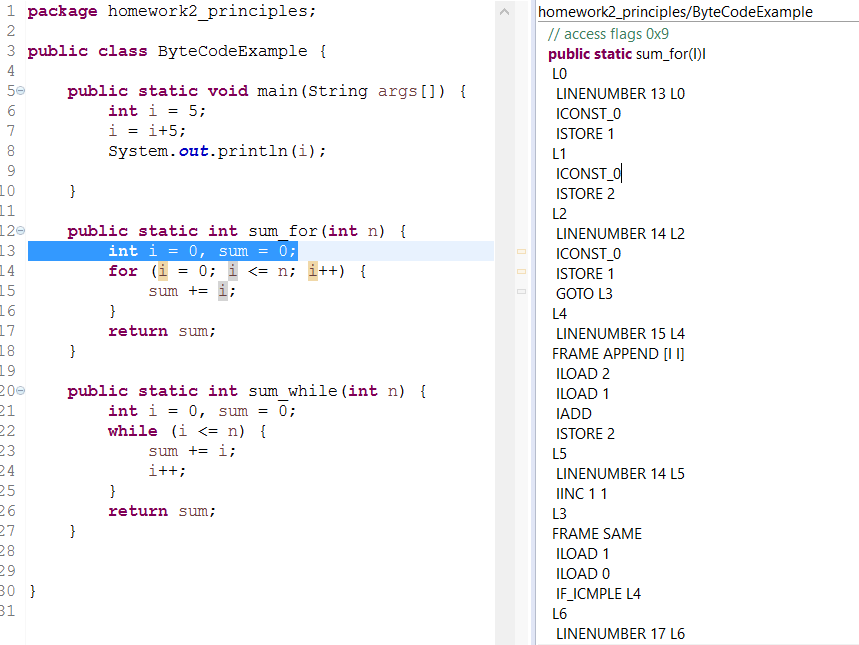
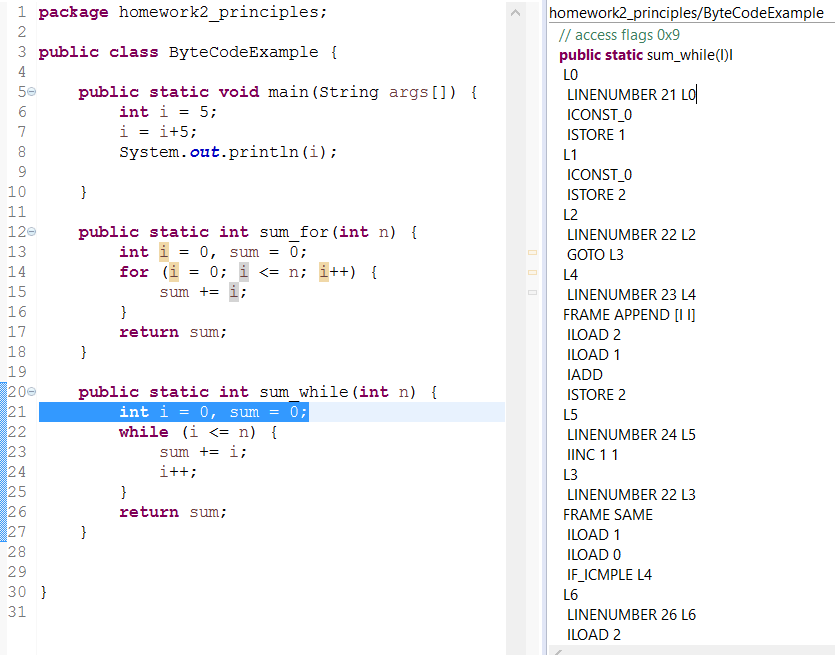
sum += i;

i++;

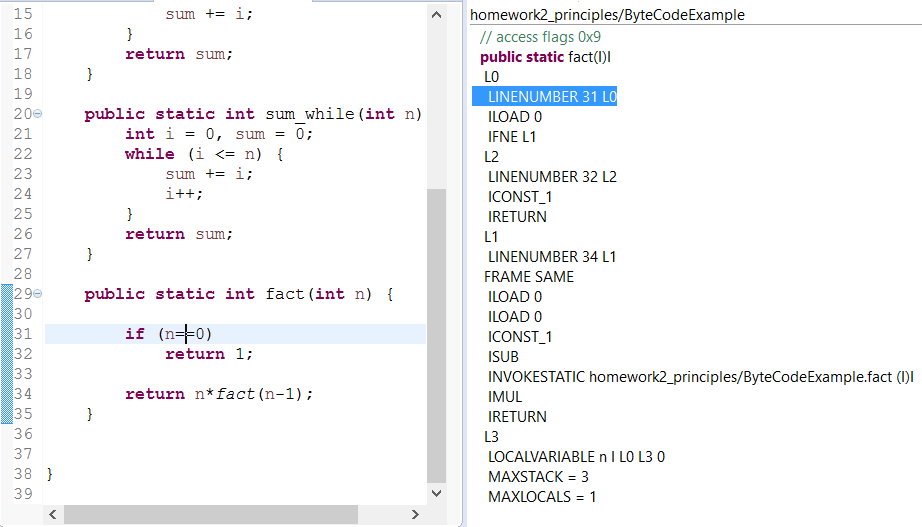
}

**return** sum;

}

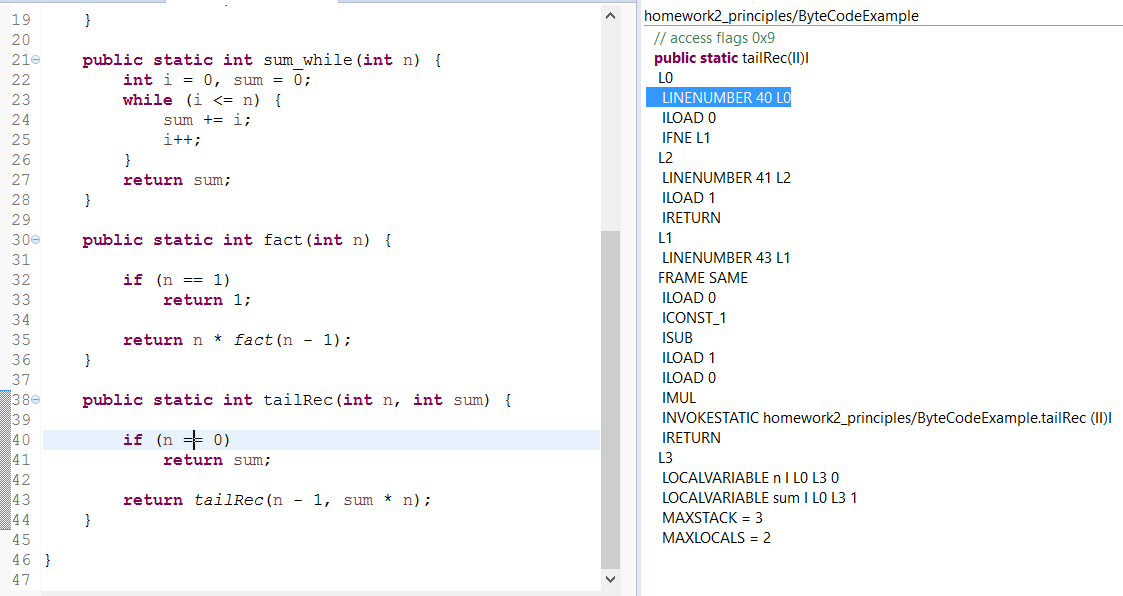
* The methods are doing practically the same thing, in a slightly different way. The only difference in the methods are that i is incremented in different places in the code, but in the bytecode i is always incremented in L5.   
  The for-loop also initializes i again in the loop, unlike the while-loop, which is the only place in the byte code where there is a difference. Other than this small difference, the bytecode is identical.

1. Write the factorial function (with the profile: public static fact(int n)) and describe the bytecode generated by this function.



* + The function starts with loading variable 0 and checking if it is equal to 0. If true, it uses the ‘goto’-function to go to L1. Here it will subtract 1 from the variable 0, and invoke the static method recursively and return a value which it will multiply with variable 0 and return. L3 states that the variable n is variable 0, and the maximum for the memory stack is 3, where 1 local variable is stored.

1. Choose a tail recursive function and describe the bytecode generated by this function. Compare with the code generated for a recursive function obtained in c).



* + The difference between the recursive function in c) starts in L2 where the tailrecursive function returns a variable stored at 1, where the recursive function in c) returns the value 1.   
    Both functions will check if n is equal to 0, and if not the functions goes to L1. Here there is a slight difference; the tailrecursive function will subtract 1 from the variable stored at 0, and will then multiply the variable at 0 with the variable at 1. These variables will be fed into the static method tailRec again and return the value. L3 states that the variable n is variable 0, and sum is stored in variable 1, and the maximum for the memory stack is 3, where 2 local variable is stored.

**References**

* The Java Virtual Machine Specification <https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf> (Java 8 SE)
* 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)