**Homework 2**

**Weili Zhong**

**Programming Languages Principles and Implementation**

**Instructions:**

* Due date: 10/8 (No late homework will be accepted. The solution of the homework will be posted on 10/9 after class. The midterm is shortly after.)
* 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 available on GitHub under the CS361 and Homework2 directories.

* 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:**

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.

Class Circle is the parent and ColoredCircle is the child. This is polymorphic because there are two draw methods that will return different outputs. ColoredCircle extends Circle which indicated that it inherits the method draw() then overrides it with its own draw method.

1. c is of type Circle and d is of type ColoredCircle. Can we write d = c;? Why?

No, because not all parent instances will have the same characteristics of the child. For instance C may be a circle but is no colored but D is a ColoredCircle which is colored therefor they are different.

1. 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?

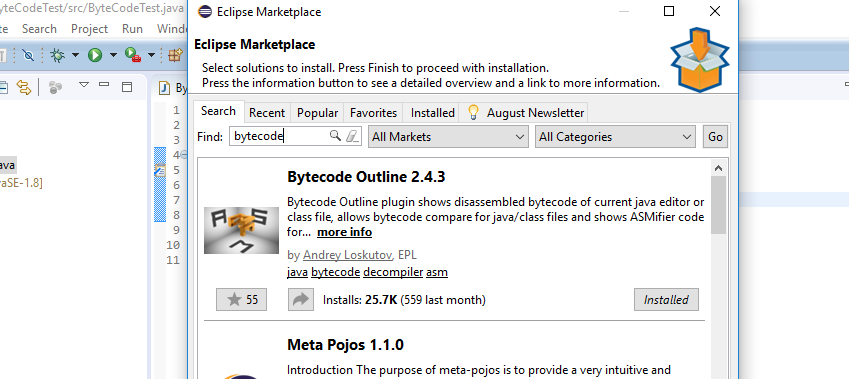
c = d;

c.draw();

Yes, because D is a child of C inherits all characteristics from C. This would mean that if a draw method is called, it would be the draw method in class Circle

**Question 2:**

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 Neon version of Eclipse and with Eclipse Marketplace Byte Outline 2.4.3 plugin and it works! ]*

1. What Eclipse version are you using?

Eclipse Oxygen

1. What Java version are you using?

Version 8 Update 171 (1.8.0\_171-b11)

1. What is the Bytecode generated by the following statements?

int i = 5;

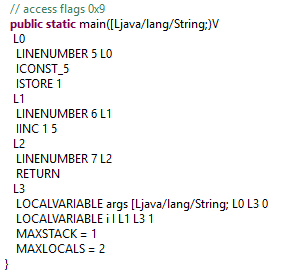
i = i+5;

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

**ICONST\_5** - The value 5 gets pushed into the stack

**ISTORE 1** - Value 5 gets popped off stack and stored

**IINC 1 5** - i get incremented by 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) {

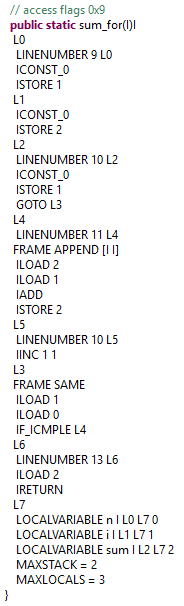
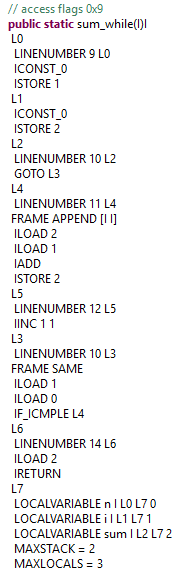
sum += i;

i++;

}

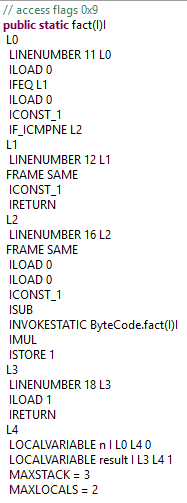
**return** sum;

}

In L2 in sum\_for 0 is pushed into the stack and later poped and stored in i in the for statement where i=0. On the otherhand in L2 for sum\_while there are no values being pushed or popped.

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



In L0 constants 1 and 0 are loaded in and compared to variable n. If they n is equal to either 1 or 0, 1 will be popped into the return value. If else n will be loaded in L2 in which the factorial method is called recursively.

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).

What tail recursive function did you choose?

Tail recursive for calculating factorials

**class** GFG {

// A tail recursive function

// to calculate factorial

**static** **int** factTR(**int** n, **int** a)

{

**if** (n == 0)

**return** a;

**return** *factTR*(n - 1, n \* a);

}

// A wrapper over factTR

**static** **int** fact(**int** n)

{

**return** *factTR*(n, 1);

}

// Driver code

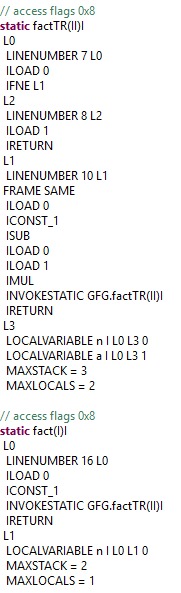
**static** **public** **void** main (String[] args)

{

System.***out***.println(*fact*(5));

}

}



In factTR method L0 starts with loading in 0 into the stack and comparing it to 0. If it’s not 0, L1 is executed. If it is it executed L2 in which 1 loaded in and is returned. Then it loads in n and 1 into the stack and subtracts 1 from n and loads n again and a to be multiplied. factTR is then called and returned.

In the fact() method n and 1 is loaded in and used to call the method factTR.

**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)

**Question 3:**

1. Write a PROLOG program that describes the British family until nowadays. Kate, William and their children should be cited in the facts. Your program will start with the facts available in the slides (slide 31) and ends with Kate, William and their children. Draw a genealogy tree first.

P(Edward VII, George V)

P(Victoria, Edward VII)

P(Alexandra, George V)

P(George VI, Elizabeth II)

P(George V, George VI)

P(Elizabeth II, Charles)

P(Charles, Harry)

P(Charles, William)

P(Diana, Harry)

P(Diana, William)

P(William, George)

P(William, Charlotte)

P(Kate, George)

P(Kate, Charlotte)

1. Write a **rule** that describes the father predicate. *Father(X,Y)* means that *X* is the father of *Y*.

Father(X,Y) := Parent(X, Y), Male(X)

**Question 4:**

Write a **recursive** function *recPow* that computes 2n for n >= 0 in Java. The function will have the following profile:

public static int recPow(int n)

The function must consider all cases and be tested exhaustively. Show your testing!

**public** **class** Power {

**public** **static** **void** main(String[] args) **throws** Exception {

System.***out***.println(*recPow*(1));

System.***out***.println(*recPow*(5));

System.***out***.println(*recPow*(10));

System.***out***.println(*recPow*(15));

System.***out***.println(*recPow*(20));

System.***out***.println(*recPow*(30));

System.***out***.println(*recPow*(100));

}

**public** **static** **int** recPow (**int** n) {

**if** (n == 0) {

**return** 1;

}

**if** (n<0) {

**throw** **new** IllegalArgumentException("No negative numbers!");

}

**if** (n>30) {

**throw** **new** IllegalArgumentException("Number too large!");

}

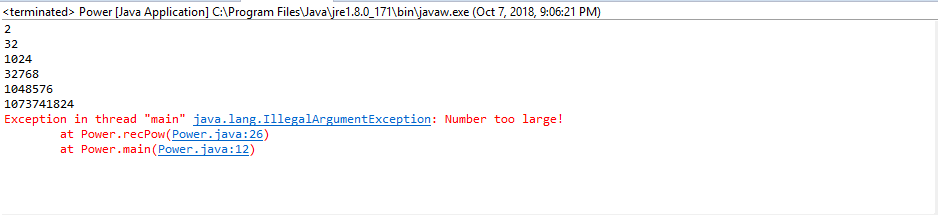
**else** {

**return** 2 \* *recPow*(n-1);

}

}

}



**Question 5:**

Write a **recursive** function merge that merges 2 arrays in Java. The function will have the following profile:

public static void mergeSort(int[] a)

The function must be tested exhaustively. Show your testing!

**public** **class** MergeSort {

**static** **int**[] *a*= {932,2319,312,1231141,123,0,-123,2138,3,1,432,14,5};

**public** **static** **void** main(String[] args) {

*mergeSort*(*a*);

**for** (**int** i = 0; i < *a*.length; i++) {

System.***out***.print(*a*[i] + " ");

}

}

**public** **static** **void** mergeSort(**int**[] a) {

// Sort the contents of array a in ascending numerical order

**if**(a.length>1)

{

**int** i,mid = a.length/2;

**int**[] half1 = **new** **int**[mid];

**int**[] half2 = **new** **int**[a.length-mid];

**for**(i=0; i<mid; i++)

half1[i]=a[i];

**for**(; i<a.length; i++)

half2[i-mid]=a[i];

*mergeSort*(half1);

*mergeSort*(half2);

**int** j=0, k=0;

**for**(i=0; j<half1.length&&k<half2.length; i++)

**if**(half1[j]<half2[k])

{

a[i]=half1[j];

j++;

}

**else**

{

a[i]=half2[k];

k++;

}

**for**(; j<half1.length; i++, j++)

a[i]=half1[j];

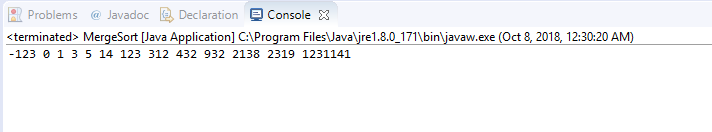
**for**(; k<half2.length; i++, k++)

a[i]=half2[k];

}

}

}



If you use code online, you will need to cite your sources.

Source: http://www.eecs.qmul.ac.uk/~mmh/ItA/algorithms/resources/mergesort.html

**Question 6:**

Dijkstra's algorithm for gcd is the following:

gcd(m,n) = m if m = n

gcd(m-n, n) if m > n

gcd(m, n-m) if m < n

1. Is this definition well-formed? Explain.
2. Is this definition well-defined? Explain.

Yes, because if m and n are integers gcd() is called which will involve substitution and and simplification until the function terminates in the end.

1. Is this definition tail recursive? Explain.

Yes, this definition is tail recursive because the last task to be executed in this algorithm is a recursive call.

1. Evaluate gcd(20,30) and show EACH step.

gcd(20,30)= gcd(20,30-20)

gcd(20,10)= gcd(20-10,10)

gcd(10,10)= **10**

1. Implement gcd in Java with the following profile:

public static int gcd(int n,int m)

**public** **class** GCD {

**public** **static** **void** main(String[] args) {

**int** result;

result = *gcd*(20,30);

System.***out***.println(result);

}

**public** **static** **int** gcd(**int** n, **int** m) {

**int** result = 0;

**if**(m == n) {

**return** m;

}

**if**(m > n) {

result = *gcd*(m-n, n);

}

**if**(m < n) {

result = *gcd*(m, n-m);

}

**return** result;

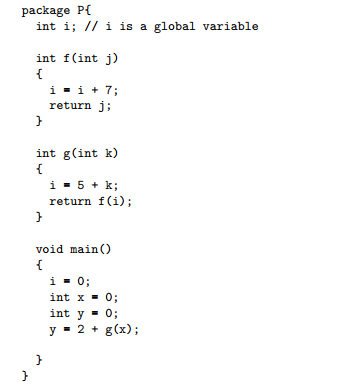
}

}

https://i.gyazo.com/e2da124fbc49a175fd362be9af888bc5.png

**Question 7:**

We consider the code below:

**

Draw the state of the memory during the execution of the code above using the following specifications:

* k is passed by value in g.
* j is passed by reference in f.
* You will use the drawing conventions seen in class.

Each step must be visible on the drawing.

What is the value of y at the end of the execution of the code? **12**

