**Homework 2**

**Programming Languages Principles and Implementation**

**Ivan Tang**

**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.

Polymorphism is the ability of an object to take on many forms. This is demonstrated by the fact that the two classes have the same methods, but have a different output. Class Circle is the parent, Class ColoredCircle is the child. ColoredCircle extends Circle, which means it inherits the method draw() and overrides it. This causes the method to return a different value than what was previously returned for the parent class.

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

We can’t write d=c due to the fact that ColoredCircle is a child of class Child. ColoredCircle is a Circle, but Circle is not a ColorCircle, in other words, the child can be like the parent, but the parent can’t be the child. Inerhitance only goes up and cannot go down into the class. The only assignment that would work is c=d; due to the fact ColoredCircle inherits the methods from its parent, Class Circle.

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?

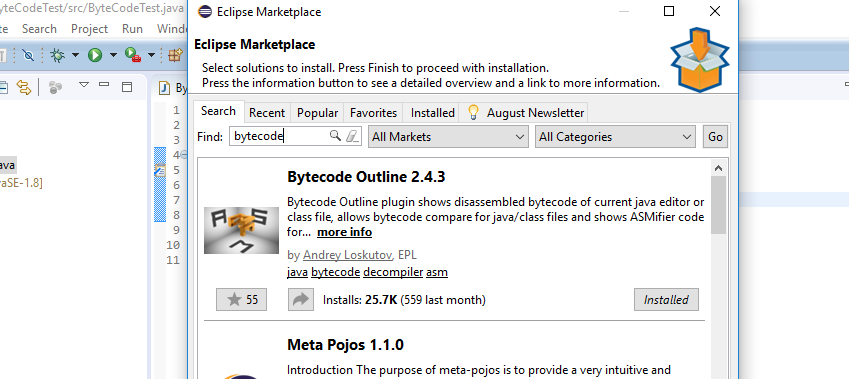
c = d;

c.draw();

This code can be executed, as explained in the previous question, ColoredCircle inherits everything from Circle. Inheritance only goes up and cannot go down into the class. Therefore, d can inherit c. The address for c will now point to a d object and that object has its own draw method which is part of the Circle class. The names are only the same because of method overriding, but since the variable is now pointing to a Circle object, then the method called would be coming from the circle object and not the ColoredCircle object.

**Question 2:**

Install the following Eclipse Bytecode Outline plugin from: <http://asm.objectweb.org/teclipse/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?

Version: Oxygen.3 Release (4.7.3)

1. What Java version are you using?

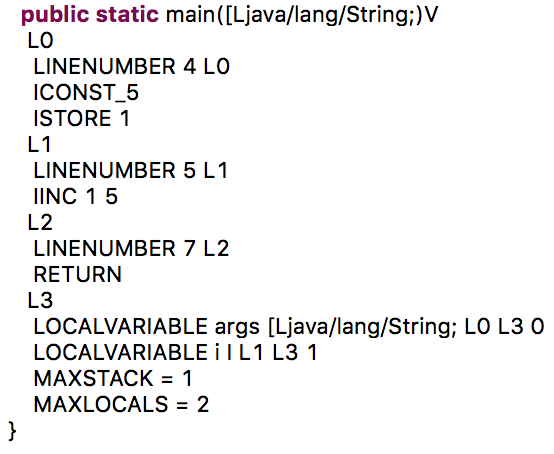
10.0.2 build 13 Mac

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.



The first block L0 reads linenumber 4 and then pushes an integer 5 onto the stack. Then the istore command pops the interger off of the stack and stores it into a local variable. The second block, L1 simply increments the value in the previous variable 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;

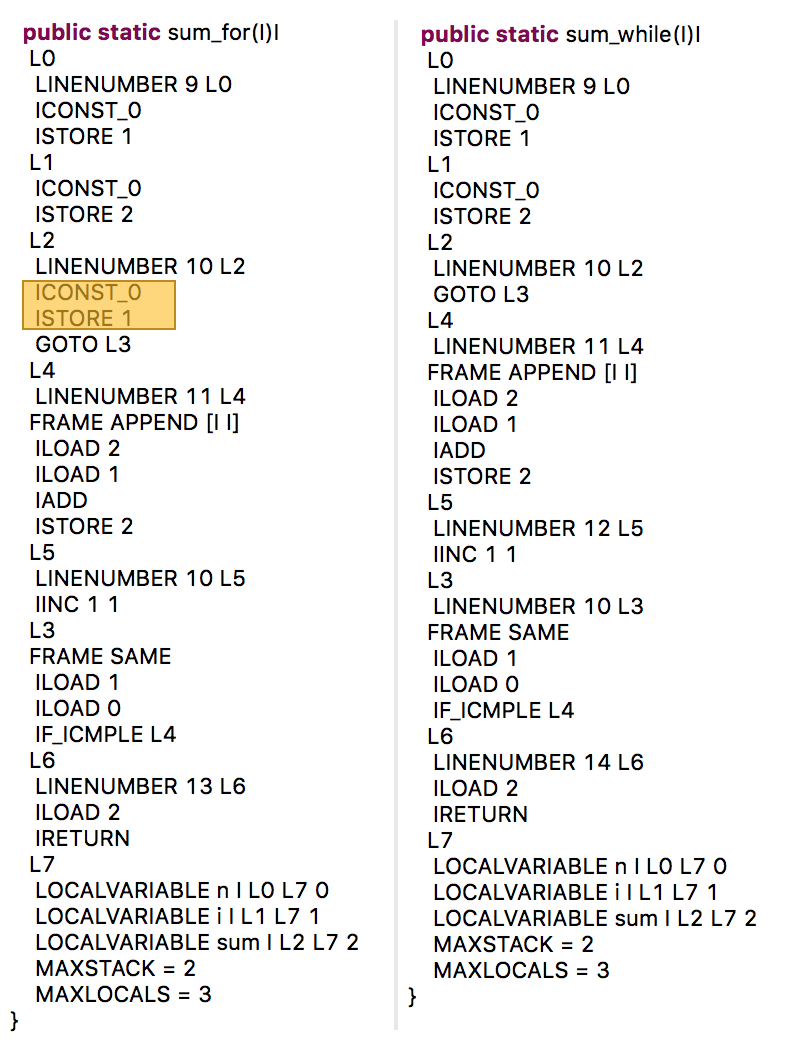
**while** (i <= n) {

sum += i;

i++;

}

**return** sum;

}

The two blocks of code are similar except for the fact that in L2, Iconst\_0 pushes zero onto the stack and then istore 1 pops it off the stack and stores it to variable i. This is double the work since the I was already assigned.

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

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

**int** result;

**if** (n==0 || n==1) {

**return** 1;

}

**else** {

result = n \* *fact*(n-1);

}**return** result;

}

The byte code that is generated from this is pretty straight-forward except for the recursive part. It begins with first retrieving the integer from the input parameter and then comparing that value with 0 by the command “IFEQ” and then if it is, the code moves onto L1, if not, then the code moves to L2 and throw an arithmeticexception. It uses INVOKESPECIAL and ATHROW in order to do this.

If the value is greater than zero then the code moves into L1, which is where the actual recursion happens. The base case if first tested and if it fails then a command called INVOKESTATIC is called with the appropriate parameters. The parameters are loaded and calculated in the 4 lines just above the invokestatic command. After the recursion, a multiply command is called before return is executed.

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?

Source : https://www.geeksforgeeks.org/tail-recursion/

class GFG { 1

// 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)

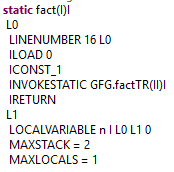
{

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

}

}

Factorial tail recursive function



As mentioned in the prior question, with recursive factorial, iconst\_1 loads a 1 onto the stack and the invokestatic invokes a static method and puts the result on the stack.

**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*.

Male(Charles)

Male(Harry)

Male(William)

Male(Edward VII)

Male(George V)

Male(George VI)

rule G(x,y):-P(x,z),P(z,y).

**Father Rule**

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

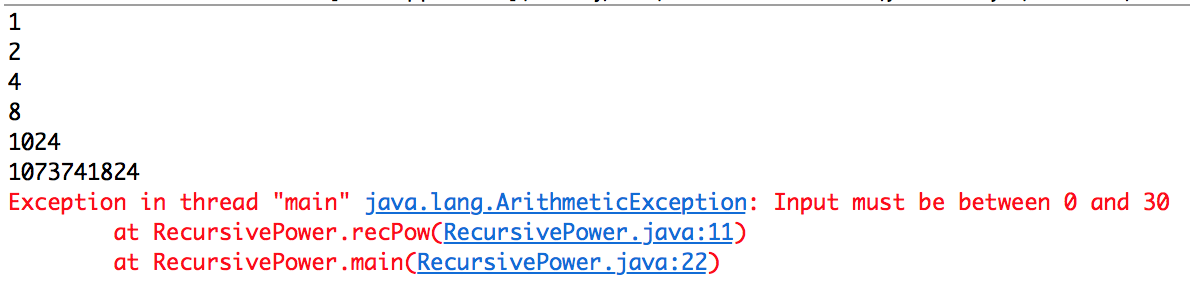
**Question 4:**

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 RecursivePower { |
|  |  |
|  | public static int recPow(int n) { |
|  | if(n >= 0 && n < 31){ |
|  | if(n == 0) |
|  | return 1; |
|  | else{ |
|  | return (2 \* recPow(n-1)); |
|  | } |
|  | }else{ |
|  | throw new ArithmeticException("Input must be between 0 and 30"); |
|  | } |
|  | } |
|  |  |
|  | public static void main(String [] args){ |
|  | System.out.println(recPow(0)); |
|  | System.out.println(recPow(1)); |
|  | System.out.println(recPow(2)); |
|  | System.out.println(recPow(3)); |
|  | System.out.println(recPow(10)); |
|  | System.out.println(recPow(30)); |
|  | System.out.println(recPow(31)); |
|  | } |
|  | } |



**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!

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

Source: http://web.cs.iastate.edu/~smkautz/cs227s18/examples/week11/MergeSortAlt2.java

public class MergeSort {

public static void mergeSort(int[] a){

if (a.length > 1) {

int i, mid = a.length / 2; //split array in 2 halves

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); //recursive call for first half of array to keep splitting

mergeSort(half2); //recursive call for second half of array to keep splitting

//sort and merge

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];

}

}

public static void main(String[] args) {

int i;

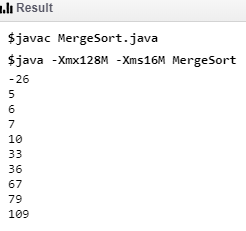
int[] A = { 5, 6, 7, 109, -26, 67, 79, 33, 10, 36 };

mergeSort(A);

for (i = 0; i < A.length; i++) {

System.out.println(A[i]);

} // System.out.println(A);}}



**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.

A well-formed recursive definition of a function f consists of two parts:

– the basis case defines the function f for the “smallest” arguments in terms of previously defined functions (including constants), (no f).

– the general case defines values f(x) in terms of previously defined functions and values f(y) for “smaller” arguments y

Yes. There are both a base case and a general case.

1. Is this definition well-defined? Explain.

A recursive function is said well-defined, if it is possible to compute f(n) for all n for which the function is defined. Otherwise it is said partially defined.

This definition is well-defined. This function can handle all n.

1. Is this definition tail recursive? Explain.

This is not a tail recursive function because the function does not end with a recursive statement. In tail recursion, you perform your calculations first, and then you execute the recursive call, passing the results of your current step to the next recursive step.

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)

Gcd(10 , 10) = 10

1. Implement gcd in Java with the following profile:

public static int gcd (int n, int m)

{

if (m == n)

return m;

else if (m > n)

return GCD(m-n, n);

else

return GCD(m, n-m);

}

public static void main (String args[])

{

System.out.println(gcd(30, 20));

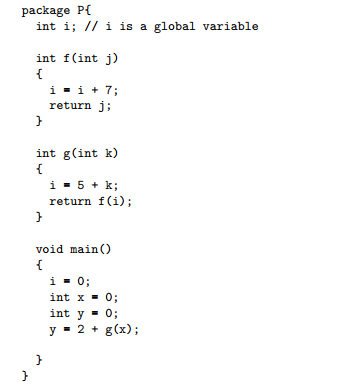
}

}

Output: 10

**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?