CS 361

HOMEWORK #2

By Jason Antigua

Cooperated with: Niaz Ahmed

**Question 1: History of programming languages**

Put the following programming languages on a chronological timeline. The year must be provided. **In addition,** indicate the name of the designer of the programming language, where it was created (company, national lab, higher education institution etc.), and the country.

* Fortran – 1957
* Cobol – 1959
* SML – 1984
* Prolog – 1972
* EIFFEL – 1986
* C – 1972
* PASCAL – 1970
* C++ – 1983
* Java – 1995
* Ruby – 1995
* Python – 1991
* ADA – 1980
* ISETL – 1986
* Lisp – 1958
* Perl – 1987
* Kotlin – 2011

**Timeline of Programming Languages:**

**1950s:**

* **Fortran (1957) –** Designed by John Backus and created at IBM in the U.S.
* **Lisp (1958) –** Designed by John McCarthy and created at the University of Massachusetts Institute of Technology, which is in the U.S.
* **Cobol (1959) –** Designed by CODASYL otherwise known as the Committee on Data Systems and Languages. The committee was a joint effort of industry, major universities, and the Unites States Government.

**1970s:**

* **PASCAL (1970) –** Designed by Niklaus Wirth at
* **C (1972) –** Designed by Dennis Ritchie at Nokia Bell Labs, which is a scientific research and development company in the U.S.
* **Prolog (1972) –** Designed by Alain Colmerauer at the University of Aix-Marseille, France.

**1980s:**

* **Ada (1980) –** Designed by Jean Ichbiah at CII Honeywell Bull which is a computer company located in France.
* **C++ (1983) –** Designed by Bjarne Stroustrup at Nokia Bell Labs, U.S.
* **SML (1984) –** no specific designer can be found but SML of New Jersey was made in Nokia Bell Labs, U.S.
* **EFFEIL (1986) –** Designed by Bertrand Meyer at Eiffel Software. Which resides in Goleta, California, U.S.
* **ISETL (1986) –** This programming language is derived from SETL which was originally developed in New York University. The designer for SETL was Jacob Schwartz.
* **Perl (1987) –** Designed by Larry Wall while he was working for System Development Corporation in Santa Monic, California.

**1990s:**

* **Python (1991) –** Designed by Guido van Rossum at Centrum Wiskunde & Informatica, which is a research institute for mathematics, and computer science is residing in the Netherlands.
* **Java (1995) –** Designed by James Gosling at Sun Microsystems, California, U.S.
* **Ruby (1995) –** Designed by Yukihiro Matsumoto in Japan.

**2000s:**

* **Kotlin (2011) –** Designed by a team of people who work for the company JetBrains, but developed the language in Saint Petersburg, Russia.

**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.

Since ColoredCircle extends Circle, it’s utilizing Circle’s draw method. This method is going to always change in ColoredCircle, and because of that the output of the draw method is also going to change depending on how it is invoked.

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

We can’t write that because the subclass may be any instance of the superclass, but all of the instances of the superclass isn’t always going to be in the subclass. An example would be a superclass named Pasta and the subclass being named Spaghetti. Not every form of pasta is going to be spaghetti, so we can never say an object type of the subclass can be assigned to an object class of the superclass.

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 would work because like mentioned before all subclasses are instances of their parent’s class. Like my previous example, all spaghetti is a pasta, so you make the parent class assigned to the subclass. When the code gets executed the object is type Circle, so it will execute the first draw method.

**Question 3:**

1. What Eclipse version are you using?

I am using Oxygen Release (4.7.0).

1. What Java version are you using?

I am using Java version 8 (build 1.8)

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

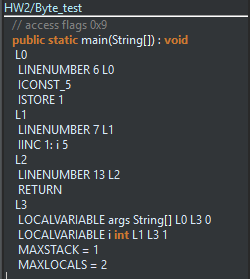
int i = 5;

i = i+5;

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

The syntax the of the bytecode starts at line 6 for the method. It does and Iconst\_5, which means it pushes the integer value 5 into the stack. Then Istore 1 is when the stack pops the value 5 and stores into a local variable. When you get to line 7 it says Iinc 1: i 5, that means increment the two parameters and put them into a local variable. Finally line 13 calls the return method which pops

the value stored inside the local variable and returns the value stored inside.



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) {

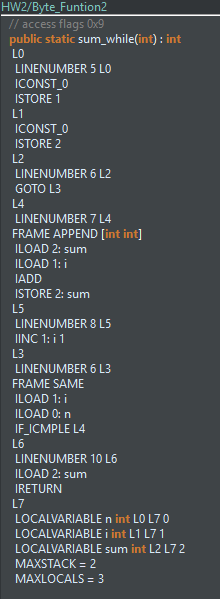
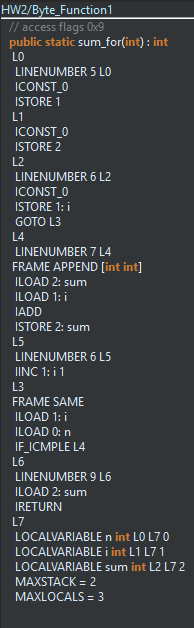
sum += i;

i++;

}

**return** sum;

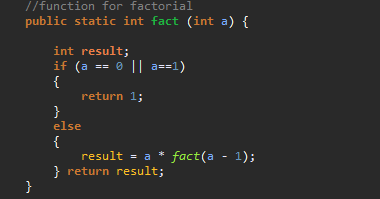
}

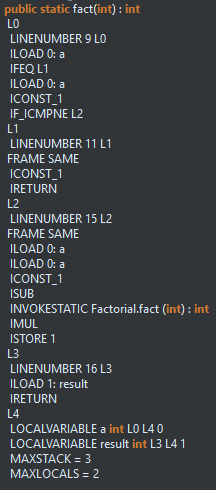


Above are pictures of bytecodes for function 1 and function 2 respectively. As you can see they both start off the same way. They each store and integer value of 0 twice, into the stack to then be popped so they can both be stored in a local variable, which can be seen in line 5. After that they both diverge into different bytecode because of their functions. Function 1 stores another integer of 0 into a local variable representing i. For function 2 it’s checking to see if the while loop is true before running. Afterwards they both update their variable named sum assigning it to the variable name i, and then it does the add function to both variables and stores it into the local variable named sum inside the stack. Function 2 it goes through the same byte process as you can see and then right after it rechecks its while condition to see if it should add another value into the stack. Once they’re both done they both have a return which pops the numbers from their stacks and returns the values.

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

The pictures below are the code I wrote for the function factorial and its bytecode. As you can see starting at line 9 the function loads two constants 1 & 0 into the stack since it’s being compared to see if variable a equals to either of them. If one them does the stack will pop 1 to return the value. If not ‘a’ gets loaded back into the method because factorial gets invoked in the stack. Afterwards there is a local variable in the stack with the value of the method that got invoked. Later on return gets called and pops the value of variable result and the stack is empty again.





The above pictures are the code I wrote for the function factorial and its bytecode. As you can see starting at line 9 the function loads two constants 1 & 0 into the stack since it’s being compared to see if variable a equals to either of them. If one them does the stack will then pop 1 to return the value. If not ‘a’ gets loaded back into the method because factorial gets invoked in the stack. Afterwards there is a local variable in the stack with the value of the method that got invoked. Later on return gets called and pops the value of variable result and the stack is empty again.

**Question 4:**

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.

P(Victoria, Edward VII).

P(Edward VII, George V).

P(Alexandra, George V).

P(George V, George VII).

P(George VI, Elizabeth II).

P(Elizabeth II, Charles).

P(Charles, William).

P(William, George).

P(William, Charlotte).

P(Charles, Harry).

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

Rule: Father(y):-P(x,y).

**Question 5:**

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

**if** (n == 0)

{

**return** 1;

}

**if** (n == 1)

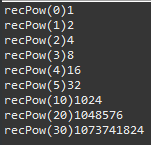
{

**return** 2;

}

**else**{

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



**Question 6:**

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

public static int[] mergeSort(int[] a, int[] b)

You will use the split function of slide 18 (odd and even positions).

**import** java.util.Arrays;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** **mergeArray** {

**static** **int**[] *answer*;

**static** **int** *spotNew*;

**static** **int** *spotOld*;

// WRITE A RECURSIVE FUNCTION THAT MERGES TWO ARRAYS

**public** **static** **int**[] **mergeSort**( **int**[] x, **int**[] y)

{

// X IS EMPTY

**if** ( x.length == 0 )

{**return** y;}

// Y IS EMPTY

**if** ( y.length == 0 )

{**return** *answer*;}

**else**

{

// ADD VALUE OF Y TO THE ANSWER ARRAY

*spotOld* = *spotNew*;

*answer*[*spotNew*] = y[*spotNew*-*spotOld*];

// MAKE A NEW Y ARRAY

**int**[] **new\_y** = **new** **int**[y.length - 1];

// POPULATE THE NEW Y ARRAY

**int** **old\_y\_counter** = 1;

**for**(**int** **i** = 0; i < new\_y.length; i++)

{

new\_y[i] = y[old\_y\_counter];

old\_y\_counter += 1;

}

// INCREASE SPOT COUNTER

*spotNew*+=1;

// RECURSE

**return** *mergeSort*(x,new\_y);

}

}

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

{

**Scanner** **scan** = **new** Scanner(**System**.***in***);

**Scanner** **scan\_2** = **new** Scanner(**System**.***in***);

**System**.***out***.println("Enter Length of A: ");

**int** **length\_a** = scan.nextInt();

**System**.***out***.println("Enter Length of B: ");

**int** **length\_b** = scan.nextInt();

// -------------------------------------------------------- INITIALIZE ALL ARRAYS

**int**[] **a** = **new** **int**[length\_a];

**int**[] **b** = **new** **int**[length\_b];

*answer* = **new** **int**[length\_a + length\_b];

*spotNew* = length\_a;

*spotOld* = *spotNew*;

// -------------------------------------------------------- POPULATE A ALL ARRAYS

**for** ( **int** **i** = 0; i < length\_a; i ++)

{

**Random** **generator** = **new** Random();

**int** **value** = generator.nextInt(50);

a[i] = value;

}

**for** ( **int** **i** = 0; i < length\_b; i ++)

{

**Random** **generator** = **new** Random();

**int** **value** = generator.nextInt(50);

b[i] = value;

}

**for** ( **int** **i** = 0; i < length\_a; i++ )

{*answer*[i] = a[i];}

// --------------------------------------------------------

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

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

// PRINT ARRAY A

**System**.***out***.println("Array A: ");

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

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

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

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

// PRINT ARRAY B

**System**.***out***.println("Array B: ");

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

{**System**.***out***.print(b[i] + " ");}

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

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

// TEST MERGE FUNCTION

**System**.***out***.println("Result: " + **Arrays**.*toString*(*mergeSort*(a,b)));

}

}

