Assignment 2

Principles of Programming Languages

By

Akshay Mehta

U01452014

Professor C scharff.

**Homework 2**

**Programming Languages Principles and Implementation**

**Instructions:**

* Due date: 10/16 (No late homework will be accepted.)
* 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).
* Bring a hard copy of your homework to class.
* Grade: 100 points

**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
* Cobol
* SML
* Prolog
* EIFFEL
* C
* PASCAL
* C++
* Java
* Ruby
* Python
* ADA
* ISETL
* Lisp
* Perl
* Kotlin

**Answer:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LANGUAGE** | **YEAR** | **NAME OF DESIGNER** | **PLACE WHERE IT WAS CREATED** | **COUNTRY** |
| FORTRAN | 1957 | JOHN W BACKUS | IBM | USA |
| LISP | 1958 | JOHN McCARTHY | MIT | USA |
| COBOL | 1959 | CODASYL(Conference/Committee on Data Systems Languages) |  | USA |
| ISETL | 1969 | JACOB T. SCHWARTZ | NEW YORK UNIVERSITY | USA |
| PASCAL | 1970 | NIKLAUS WIRTH |  |  |
| PROLOG | 1972 | ALAIN COLMERAURER WITH PHILIPPE ROUSSEL |  | FRANCE |
| C | 1972 | DENNIS RITCHIE | BELL LABS | USA |
| ADA | 1977 | MIL-STD-1815/Ada 83:JEAN ICHBIAH  Ada 95: TUCKER TAFT  Ada 2005: TUCKER TAFT  Ada 2012: TUCKER TAFT |  | USA |
| C++ | 1983 | BJARNE STROUSTRUP | BELL LABS | USA |
| EIFFEL | 1986 | BERTRAND MEYER |  |  |
| PERL | 1987 | LARRY WALL |  | USA |
| PYTHON | 1991 | GUIDO VAN ROSSUM, CWI(STICHTING MATHEMATISCH CENTRUM) |  | NETHARLANDS |
| JAVA | 1995 | JAMES GOSLING | SUN MICROSYSTEMS | USA |
| RUBY | 1998 | YUKIHIRO “MATZ” MATSUMOTO |  | JAPAN |
| KOTLIN | 2016 | JETBRAINS |  | RUSSIA |
| SML | 1973 | ROBIN MILNER | UNIVERSITY OF EDINBURGH | LOTHIAN |

**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.

**Answer**:

Polymorphism is the ability of an object to take on many forms.

Any Java object that can pass more than one IS-A relation is considered to be polymorphic.

The draw methods in the above two classes states that ColoredCircle IS-A form of Circle class and hence the object of ColoredCircle class can be referenced to Circle object. This is called Polymorphism.

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

**Answer**:

No. Because the compiler will throw Type mismatch error. The base class object reference cannot be assigned to its child class object.

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();

**Answer**:

Yes. We can write c = d. Because C is an object of class Circle and d is an object of class ColoredCircle which is extending class Circle. Hence this statement is valid.

c = d;

c.draw();

**Execution** :

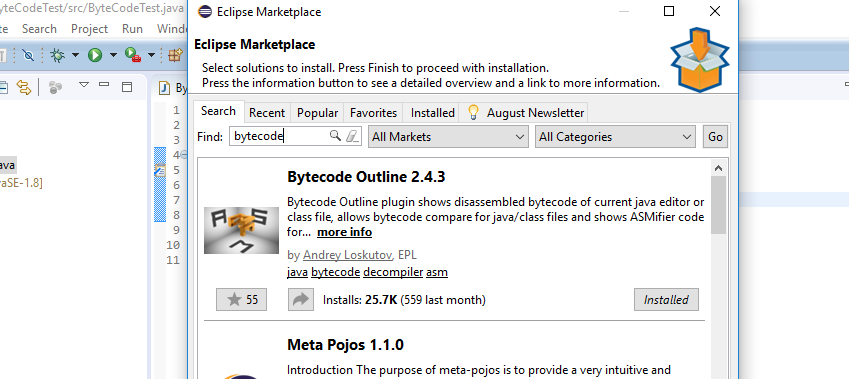
Draw method called in ColoredCircle.

Because C holds the reference of ColoredCircle object.

**Code Attachment:**

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

1. What Eclipse version are you using?

**Answer**:

Oxygen Release (4.7.0)

1. What Java version are you using?

**Answer:**  
java version "1.8.0\_111"

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.

**Answer** :

**Byte code** **of** :

**int i = 5;**

LINENUMBER 5 L0

ICONST\_5 push the integer constant 0, 1, 2, 3, 4 or 5

ISTORE 1 push the integer constant 0, 1, 2, 3, 4 or 5

L1

**i = i+5;**

LINENUMBER 6 L1

IINC 1 5 increment integer in local variable

L2

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;

}

**Answer:**

**Byte Code :**

// access flags 0x9

public static sum\_for(I)I

L0

LINENUMBER 11 L0

ICONST\_0

ISTORE 1

L1

ICONST\_0

ISTORE 2

L2

LINENUMBER 12 L2

ICONST\_0

ISTORE 1

GOTO L3

L4

LINENUMBER 13 L4

FRAME APPEND [I I]

ILOAD 2

ILOAD 1

IADD

ISTORE 2

L5

LINENUMBER 12 L5

IINC 1 1

L3

FRAME SAME

ILOAD 1

ILOAD 0

IF\_ICMPLE L4

L6

LINENUMBER 15 L6

ILOAD 2

IRETURN

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

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

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

sum += i;

i++;

}

**return** sum;

}

**Answer:**

**Byte Code :**

public static sum\_while(I)I

L0

LINENUMBER 19 L0

ICONST\_0

ISTORE 1

L1

ICONST\_0

ISTORE 2

L2

LINENUMBER 20 L2

GOTO L3

L4

LINENUMBER 21 L4

FRAME APPEND [I I]

ILOAD 2

ILOAD 1

IADD

ISTORE 2

L5

LINENUMBER 22 L5

IINC 1 1

L3

LINENUMBER 20 L3

FRAME SAME

ILOAD 1

ILOAD 0

IF\_ICMPLE L4

L6

LINENUMBER 24 L6

ILOAD 2

IRETURN

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

**Answer :**

**Function :**

/\*\*

\* Factorial function to calculate factorial of n using recursion

\* **@param** n

\* **@return** BigInteger is used to handle outOfRange Values

\*/

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

**try** {

**if** (n > 1) {

**return** BigInteger.*valueOf*(n).multiply((*fact*(n-1)));

} **else** {

**return** BigInteger.*valueOf*(1);

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

**return** BigInteger.*valueOf*(1);

}

}

**Byte Code:**

// access flags 0x9

public static fact(I)Ljava/math/BigInteger;

TRYCATCHBLOCK L0 L1 L2 java/lang/Exception

TRYCATCHBLOCK L3 L4 L2 java/lang/Exception

L0

LINENUMBER 16 L0

ILOAD 0

ICONST\_1

IF\_ICMPLE L3

L5

LINENUMBER 17 L5

ILOAD 0

I2L

INVOKESTATIC java/math/BigInteger.valueOf (J)Ljava/math/BigInteger;

ILOAD 0

ICONST\_1

ISUB

INVOKESTATIC Factorial.fact (I)Ljava/math/BigInteger;

INVOKEVIRTUAL java/math/BigInteger.multiply (Ljava/math/BigInteger;)Ljava/math/BigInteger;

L1

ARETURN

L3

LINENUMBER 19 L3

FRAME SAME

LCONST\_1

INVOKESTATIC java/math/BigInteger.valueOf (J)Ljava/math/BigInteger;

L4

ARETURN

L2

LINENUMBER 21 L2

FRAME SAME1 java/lang/Exception

ASTORE 1

L6

LINENUMBER 23 L6

ALOAD 1

INVOKEVIRTUAL java/lang/Exception.printStackTrace ()V

L7

LINENUMBER 24 L7

LCONST\_1

INVOKESTATIC java/math/BigInteger.valueOf (J)Ljava/math/BigInteger;

ARETURN

1. Choose a tail recursive function and describe the bytecode generated by this function. Compare the code that is generated with the code of a recursive function. What do you observe?

**Answer:**

**public** **static** **long** factTail(**long** n, **long** val) {

**if** (n == 1)

**return** val;

**return** *factTail*(n - 1, n \* val);

}

**Byte Code :**

// access flags 0x9

public static factTail(JJ)J

L0

LINENUMBER 29 L0

LLOAD 0

LCONST\_1

LCMP

IFNE L1

LLOAD 2

LRETURN

L1

LINENUMBER 30 L1

FRAME SAME

LLOAD 0

LCONST\_1

LSUB

LLOAD 0

LLOAD 2

LMUL

INVOKESTATIC Factorial.factTail (JJ)J

LRETURN

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

**Answers:**

parent('Edward VII','George V').

parent('Victoria','Edward VII').

parent('Alexandra','George V').

parent('George VI','Elizabeth II').

parent('George V','George VI').

parent('William','George').

parent('William','Charlotte').

parent('Kate','George').

parent('Kate','Charlotte').

parent('Charles','William').

parent('Diana','William').

parent('Charles','Kate').

parent('Diana','Kate').

father('Edward VII','George V').

father('George VI','Elizabeth II').

father('George V','George VI').

father('William','George').

father('William','Charlotte').

grandparent('x','y'):-parent('x','z'),parent('z','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!

**Answer :**

/\*\*

\* Function to calculate power with base n

\* **@param** base

\* **@param** n

\* **@return**

\*/

**public** **static** **float** recPow (**int** base, **int** n) {

**if**(n > 0) { // when power is positive{

**return** (base\**recPow*(base,n -1));

}

**else** **if** (n<0) { // when power is negative

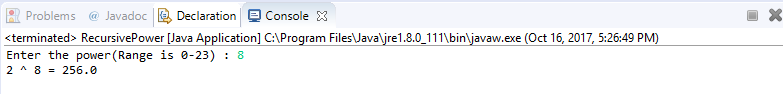
**return** 1/ (base \**recPow*(base,-n-1));

} **else** // when power is zero

**return** 1;

}

**Output :**



**Question 6:**

Write a **recursive** function that implements merge sort in Java. . The function will have the following profile:

public static int[] mergeSort(int[])

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

The function must be tested exhaustively. Show your testing!

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

**Answer :**

// Merges two subarrays of arr[].

// First subarray is arr[l..m]

// Second subarray is arr[m+1..r]

**int**[] merge(**int** arr[], **int** left, **int** middle, **int** right)

{

// Find sizes of two subarrays to be merged

**int** n1 = middle - left + 1;

**int** n2 = right - middle;

/\* Create temp arrays \*/

**int** L[] = **new** **int** [n1];

**int** R[] = **new** **int** [n2];

/\*Copy data to temp arrays\*/

**for** (**int** i=0; i<n1; ++i)

L[i] = arr[left + i];

**for** (**int** j=0; j<n2; ++j)

R[j] = arr[middle + 1+ j];

/\* Merge the temp arrays \*/

// Initial indexes of first and second subarrays

**int** i = 0, j = 0;

// Initial index of merged subarry array

**int** k = left;

**while** (i < n1 && j < n2)

{

**if** (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

**else**

{

arr[k] = R[j];

j++;

}

k++;

}

/\* Copy remaining elements of L[] if any \*/

**while** (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

/\* Copy remaining elements of R[] if any \*/

**while** (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

**return** arr;

}

// Main function that sorts arr[l..r] using

// merge()

**int**[] split(**int** arr[], **int** l, **int** r)

{

**if** (l < r)

{

// Find the middle point

**int** m = (l+r)/2;

// Sort first and second halves

split(arr, l, m);

split(arr , m+1, r);

// Merge the sorted halves

**return** merge(arr, l, m, r);

}

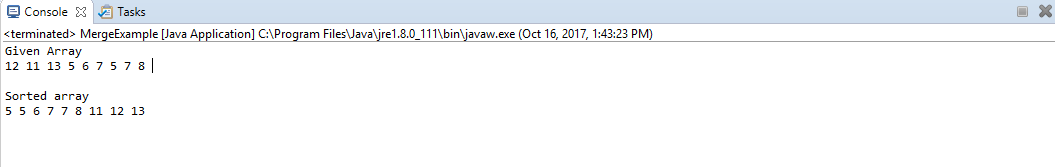
**return** **null**;

}

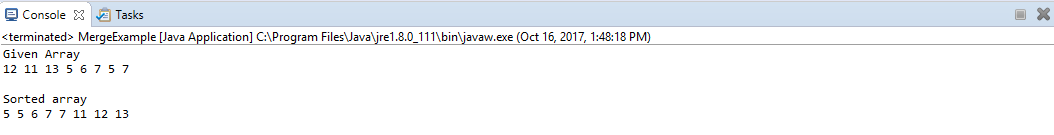
Source: http://www.geeksforgeeks.org/merge-sort/

**Output:**

**When array length is Odd**

****

**When array length is Even**

****