

# LAB MANUAL

CS1421 Object Oriented Programming Lab

**Instructor**

Aman Farooq

Lab 2

Java Language Specification & Procedural Programming Constructs using Java

**Objective**

After completing this lab, the students should be able to

* Understand the difference between primitive data types and user-defined/reference type data types
* Understand the conditional statements
* Understand the usage of mathematic functions
* Understand the for, while and do while loops
* Learn how to take user’s input

**Relevant Lecture Material**

**Textbook:** Introduction to Java Programming, Daniel Liang

**Chapters:** 2, 3, 4, 5

**Types, Values and Variables**

The Java programming language is a statically typed language, which means that every variable and every expression has a type that is known at compile time.

The Java programming language is also a strongly typed language, because types limit the values that a variable can hold or that an expression can produce, limit the operations supported on those values, and determine the meaning of the operations. Strong static typing helps detect errors at compile time.

The types of the Java programming language are divided into two categories: primitive types and reference types. The primitive types are the Boolean type and the numeric types. The numeric types are the integral types byte, short, int, long, and char, and the floating-point types float and double. The reference types are class types, interface types, and array types.

**The Kinds of Types and Values**

There are two kinds of types in the Java programming language: primitive types and reference types. There are, correspondingly, two kinds of data values that can be stored in variables, passed as arguments, returned by methods, and operated on: primitive values and reference values.

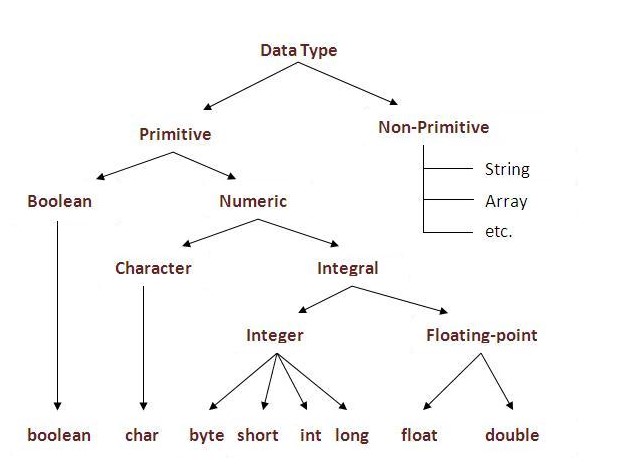
**Type:**

Primitive Type

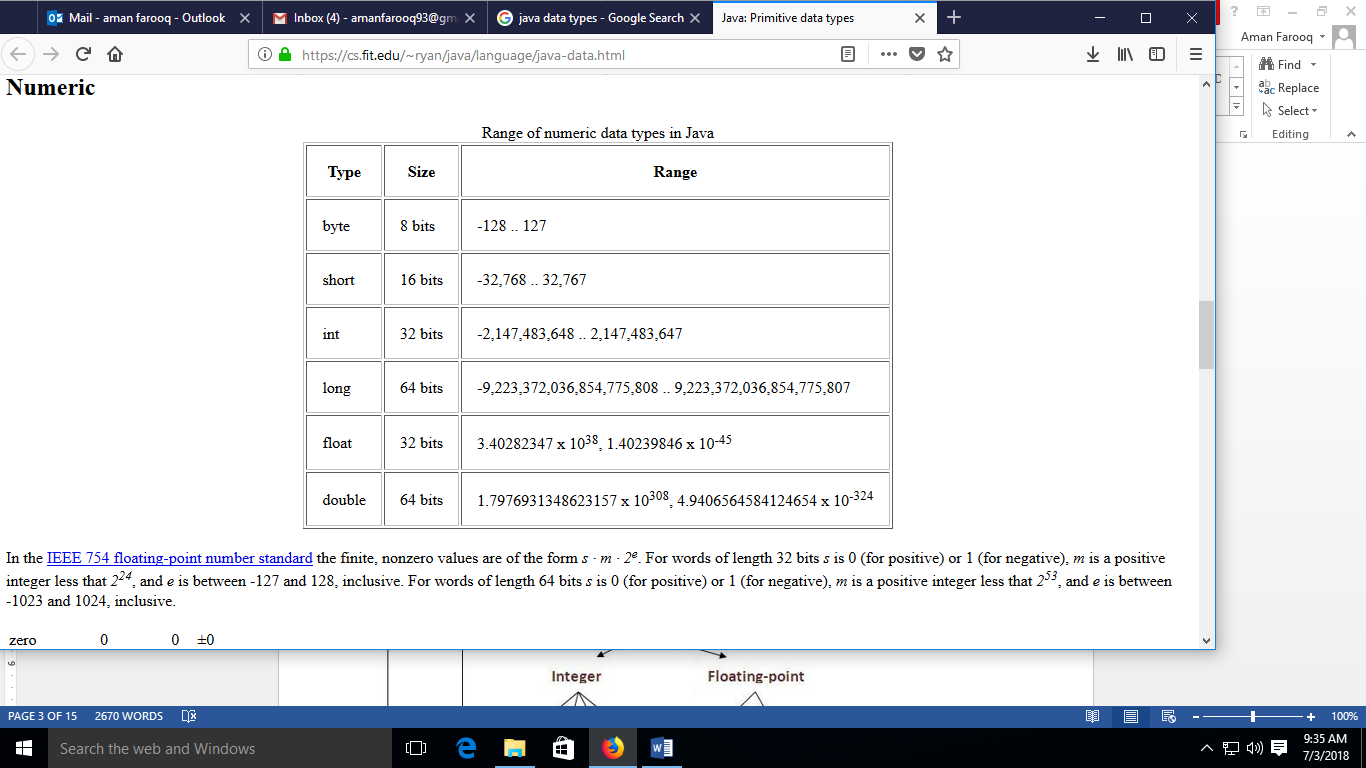
Reference Type

Primitive data types are the basic data types. These are predefined types of data, which are supported by Java. These data types are also called intrinsic or built-in types. The non-primitive data types in Java are objects and arrays. These non-primitive types are often called "reference types”.

Below are the primitive data types in Java.



|  |  |
| --- | --- |
| **Data Type** | **Size** |
| boolean | 1 bit (true or false, by default false) |
| char | 16 bits |



**Operators**

Operator in java is a symbol that is used to perform operations. For example: +, -, \*, / etc.

There are many types of operators in java which are given below:

## **Simple Assignment Operator**

= Simple assignment operator

## **Arithmetic Operators**

+ Additive operator (also used

for String concatenation)

- Subtraction operator

\* Multiplication operator

/ Division operator

% Remainder operator

## **Unary Operators**

+ Unary plus operator; indicates

positive value (numbers are

positive without this, however)

- Unary minus operator; negates

an expression

++ Increment operator; increments

a value by 1

-- Decrement operator; decrements

a value by 1

! Logical complement operator;

inverts the value of a boolean

## **Equality and Relational Operators**

== Equal to

!= Not equal to

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

## **Conditional** **Operators**

&& Conditional-AND

|| Conditional-OR

?: Ternary (shorthand for

if-then-else statement)

## **Type** **Comparison** **Operator**

instanceof Compares an object to

a specified type

instanceof is used to check if an object is an instance of a class, an instance of a subclass, or an instance of a class that implements a particular interface.

## **To get help: https://stackoverflow.com/questions/7313559/what-is-the-instanceof-operator-used-for-in-java**

## **Bitwise and Bit Shift Operators**

~ Unary bitwise complement : inverts a bit pattern; it can be applied to any of the integral types, making every "0" a "1" and every "1" a "0". For example, a byte contains 8 bits; applying this operator to a value whose bit pattern is "00000000" would change its pattern to "11111111".

<< Signed left shift

>> Signed right shift

>>> Unsigned right shift

& Bitwise AND

^ Bitwise exclusive OR

| Bitwise inclusive OR

**Control Statements**

**Java If-Else**

The Java if statement is used to test the condition. It checks boolean condition: true or false.There are various types of if statement in java.

* if statement
* if-else statement
* if-else-if
* nested if

**IF Statement**

 If statement is the simplest decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not. The Java if statement tests the condition. It executes the if block if condition is true.

**Syntax:**

**if**(condition)

{

//code to be executed

}

## **IF-else Statement**

The Java if-else statement also tests the condition. It executes the if block if condition is true otherwise else block is executed.

**Syntax:**

**if**(condition)

{

//code if condition is true

}

**else**

{

//code if condition is false

}

## **IF-else-if Statement**

The if-else-if statement executes one condition from multiple statements.

**Syntax:**

**if**(condition1)

{

//code to be executed if condition1 is true

}

**else** **if**(condition2)

{

//code to be executed if condition2 is true

}

**else** **if**(condition3)

{

//code to be executed if condition3 is true

}

...

**else**

{

//code to be executed if all the conditions are false

}

**Java Switch Statement**

The switch statement is a multi-way branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. Basically, the expression can be byte, short, char, and int primitive data types.

**Syntax:**

// switch statement

switch(expression)

{

// case statements

// values must be of same type of expression

case value1 :

// Statements

break; // break is optional

case value2 :

// Statements

break; // break is optional

// We can have any number of case statements

// below is default statement, used when none of the cases is true.

// No break is needed in the default case.

default :

// Statements

}

**Choosing Between If-Else and Switch-Case**

**Limitations of switch over if-else:**

* The variable expression are also not allowed in cases.  
  **case** i+2: is not allowed in switch, but is valid on if-else.
* You cannot use similar expressions for multiple cases. For instance, below **switch statement in c** is illegal:

**switch** (expression)

{

**case** 5: //...

**break**;

**case** 2+3: //...

**break**;

}

**Advantages of switch over if-else:**

* A switch statement works much faster than equivalent if-else.  
  *It is because compiler generates a jump table for a switch during compilation. Consequently, during execution, instead of checking which case is satisfied, it only decides which case has to be executed.*
* It is more readable and in compare to if-else statements.
* It is more manageable for having higher level of indentation than if. For instance, check below two source codes (solving same problem one using if where the other using switch) to check error messages.

**Where to use switch over if-else:**

* For menu-driven programs
* If there are large number of compares for a condition in your program, use switch over if-else ladder.
* For more complex comparisons.

**Where to use if-else over switch:**  
In case of simple and few compares, if-else executes faster and easy write. Thus, as per program’s requirement, a programmer should decide himself where to use which one condition control.

**Loops**

**For loop**

Syntax:

for(initialization;condition;incr/decr)

{

//code to be executed

}

**Example**

class ForLoopExample{ public static void main(String[] args)

{

for(int i=1;i<=10;i++)

{

System.out.println(i);

}

}

}

**Another Example**

class ForLoopExampledecrement{ public static void main(String[] args)

{

for(int i=10;i>=1;i--)

{

System.out.println(i);

}

}

}

**While Loop:**

Syntax:

//initialization

while(condition)

{

//code to be executed

//increment/decrement

}

Example:

class WhileLoopExample

{

public static void main(String[] args)

{

int i=1;

*while*(i<=10)

{

System.out.println(i);

i++;

}

}

}

Note: The for(initialization; condition; increment/decrement)

{

// body of the loop

}

is equivalent to

initialization; while(condition)

{

// body of the loop increment/decrement;

}

### **Do-While loop**

**Syntax**

do {

//code to be executed

}

while(condition);

**Example**

public class DoWhileExample

{

public static void main(String[] args)

{

int i=1; do

{

System.out.println(i);

i++; }

while(i<=10);

}

}

We use do-while use when the requirement is to execute the program at least once, irrespective of the condition. So even if the condition is not met, the body of the loop will execute once.

**Which Loop to Use?**

The while loop and for loop are called pretest loops because the continuation condition is checked before the loop body is executed. The do-while loop is called a posttest loop because the condition is checked after the loop body is executed. The three forms of loop statements—while, do-while, and for—are expressively equivalent; that is, you can write a loop in any of these three forms. Use the loop statement that is most intuitive and comfortable for you.

In general, a for loop may be used if the number of repetitions is known in advance, as, for example, when you need to display a message a hundred times. A while loop may be used if the number of repetitions is not fixed, as in the case of reading the numbers until the input is 0. A do-while loop can be used to replace a while loop if the loop body has to be executed before the continuation condition is tested.

**Taking Input From the User**

**Using Buffered Reader Class**

This is the Java classical method to take input, Introduced in JDK1.0. This method is used by wrapping the System.in (standard input stream) in an InputStreamReader which is wrapped in a BufferedReader, we can read input from the user in the command line.

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Test

{

    public static void main(String[] args) throws IOException

    {

        //Enter data using BufferReader

        BufferedReader reader =

                   new BufferedReader(new InputStreamReader(System.in));

        // Reading data using readLine

        String name = reader.readLine();

        // Printing the read line

        System.out.println(name);

    }

}

Note: To read other types, we use functions like Integer.parseInt(), Double.parseDouble(). To read multiple values, we use split().

**Using Scanner Class**

This is probably the most preferred method to take input. The main purpose of the Scanner class is to parse primitive types and strings using regular expressions, however it is also can be used to read input from the user in the command line.

import java.util.Scanner;

class GetInputFromUser

{

    public static void main(String args[])

    {

        // Using Scanner for Getting Input from User

        Scanner in = new Scanner(System.in);

        String s = in.nextLine();

        System.out.println("You entered string "+s);

        int a = in.nextInt();

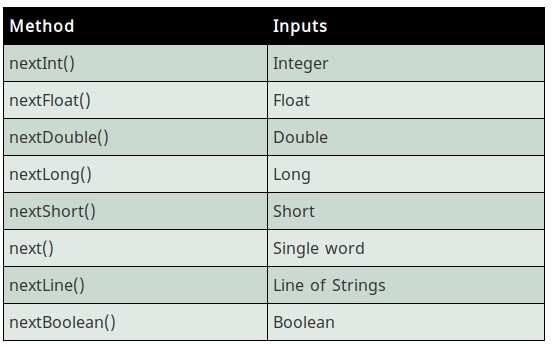
        System.out.println("You entered integer "+a);

        float b = in.nextFloat();

        System.out.println("You entered float "+b);

    }

}



**Java Math Class**

The Java programming language supports basic arithmetic with its arithmetic operators: +, -, \*, /, and %. The [Math](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html)class in the java.lang package provides methods and constants for doing more advanced mathematical computation.

The methods in the Math class are all static, so you call them directly from the class, like this:

Math.cos(angle);

**Note:** Using the [static import](https://docs.oracle.com/javase/tutorial/java/package/usepkgs.html#staticimport)language feature, you don't have to write Math in front of every math function:

import static java.lang.Math.\*;

This allows you to invoke the Math class methods by their simple names. For example:

cos(angle);

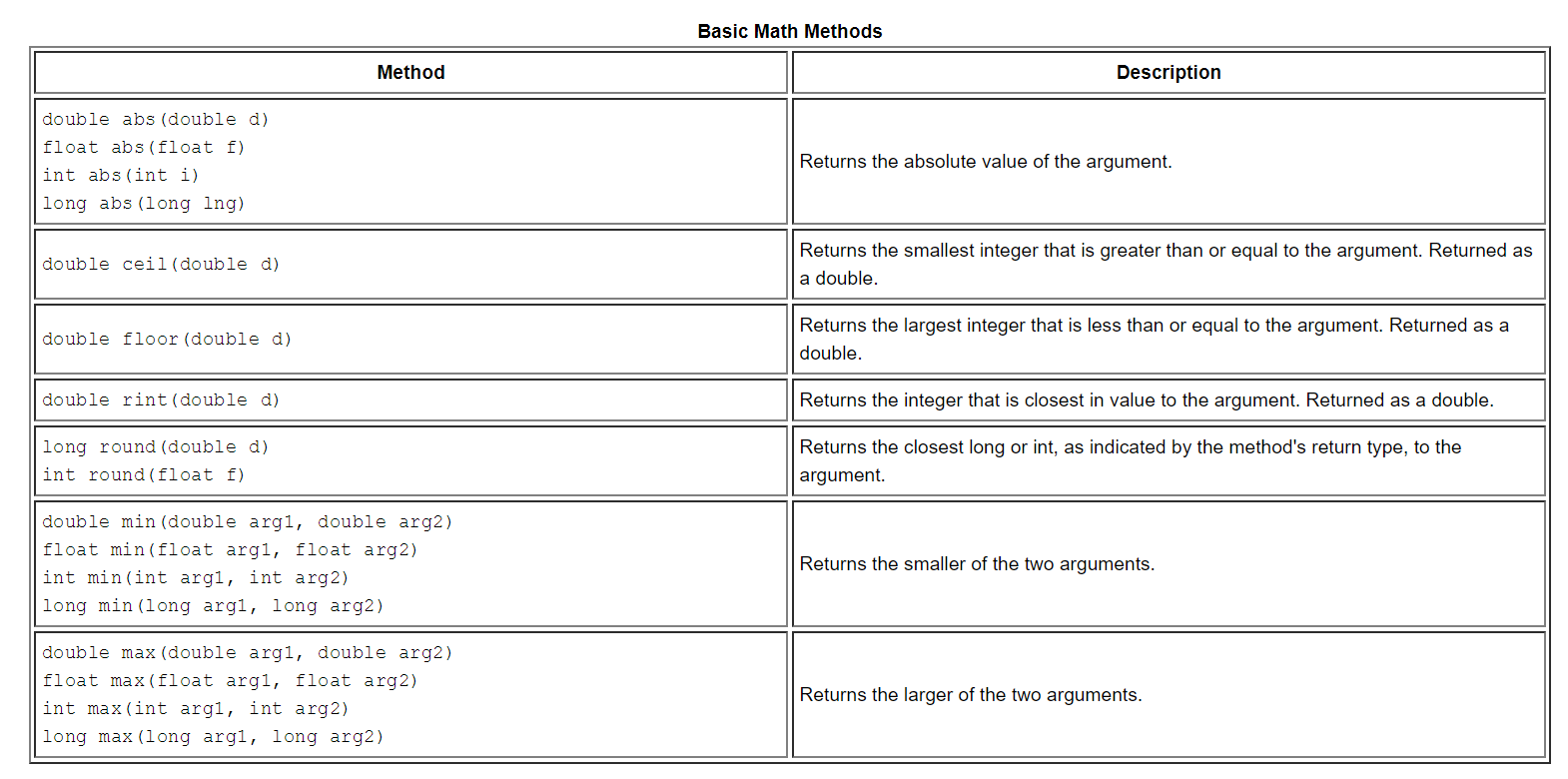
To generate a random number using Math class.

int rand = (int)(Math.random() \* range) + min

## **Constants and Basic Methods**

The Math class includes two constants:

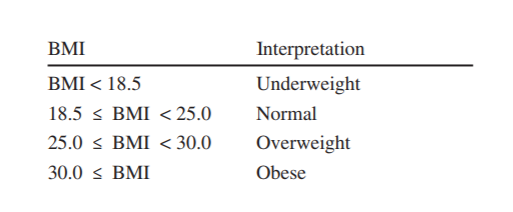
* Math.E, which is the base of natural logarithms, and
* Math.PI, which is the ratio of the circumference of a circle to its diameter.

The Math class also includes more than 40 static methods. The following table lists few of the basic methods.

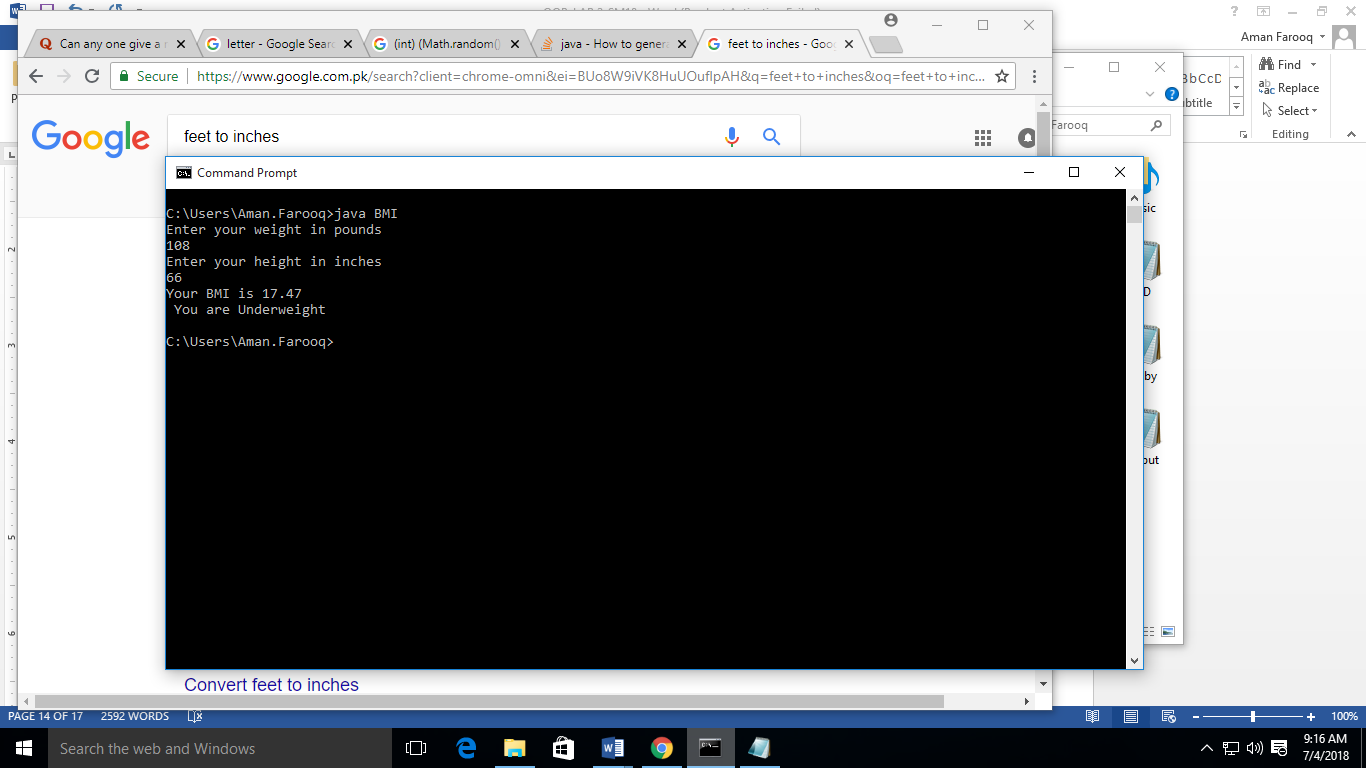
**Practice Tasks**

**Practice task 1**

Body Mass Index (BMI) is a measure of health based on height and weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. The interpretation of BMI for people 20 years or older is as follows.



Write a program that prompts the user to enter a weight in pounds and height in inches and displays the BMI.



**Practice task 2**

**Credit Limit Calculator**

A Java application is required to determine whether any of several department-store customers have exceeded the credit limit on a charge account. For each customer, the following facts are available:

a) account number

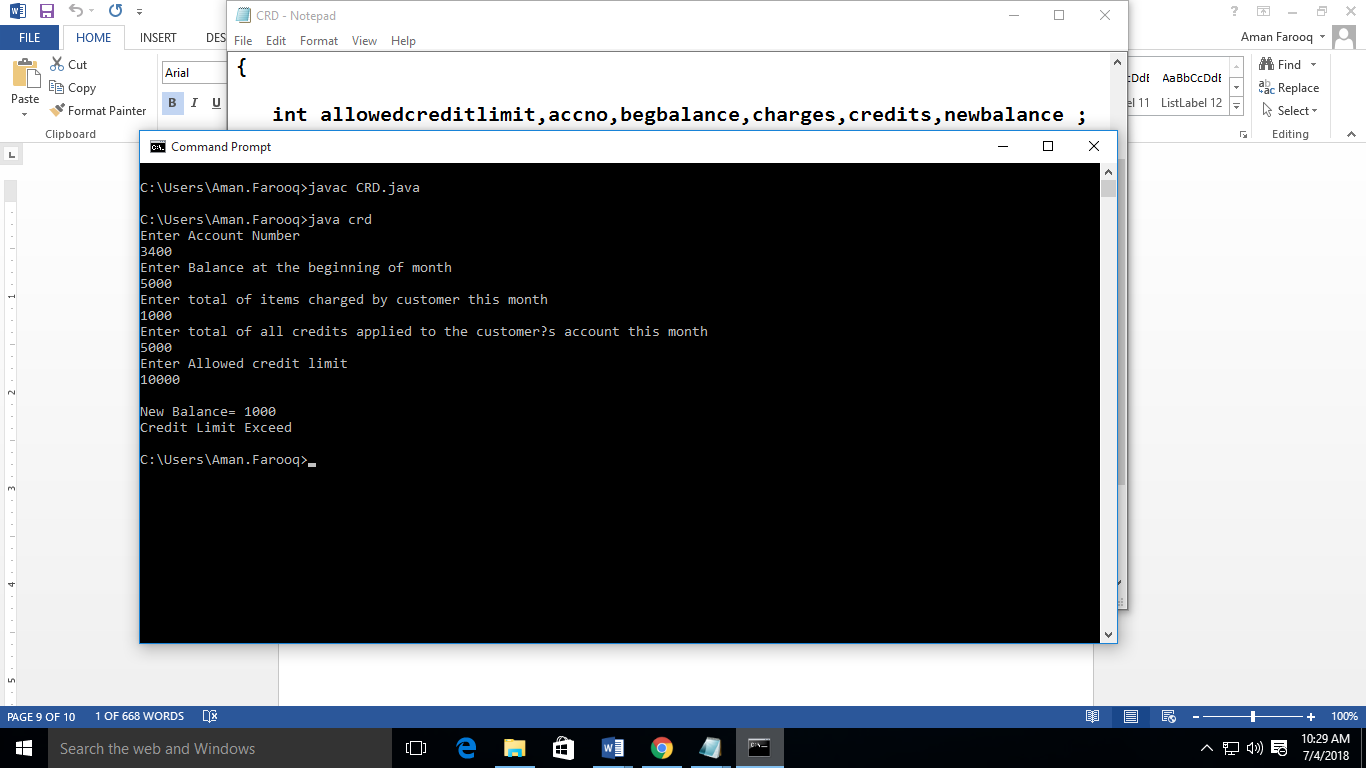
b) balance at the beginning of the month

c) total of all items charged by the customer this month

d) total of all credits applied to the customer’s account this month

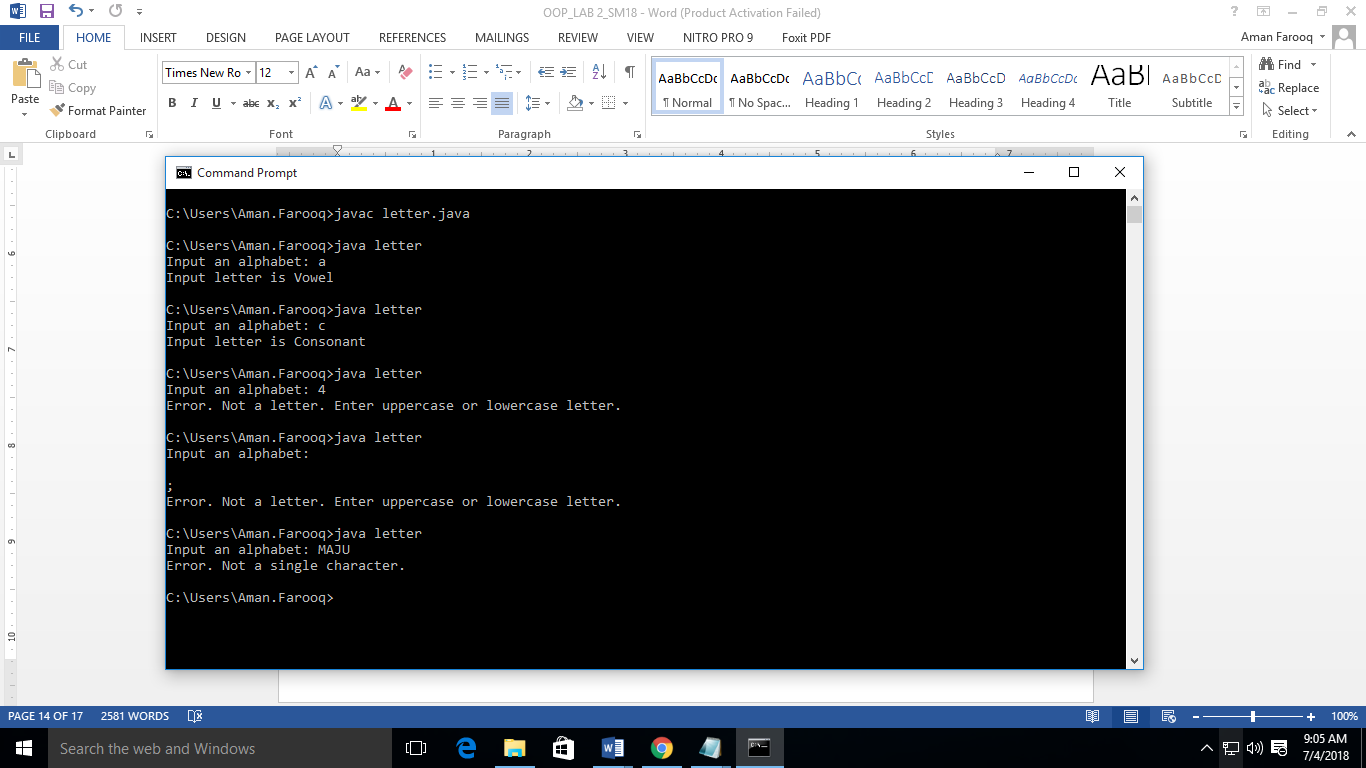
e) allowed credit limit.

The program should input all these facts as integers, calculate the new balance (= beginning balance+ charges – credits), display the new balance and determine whether the new balance exceeds the customer’s credit limit. For those customers whose credit limit is exceeded, the program should display the message "Credit limit exceeded".



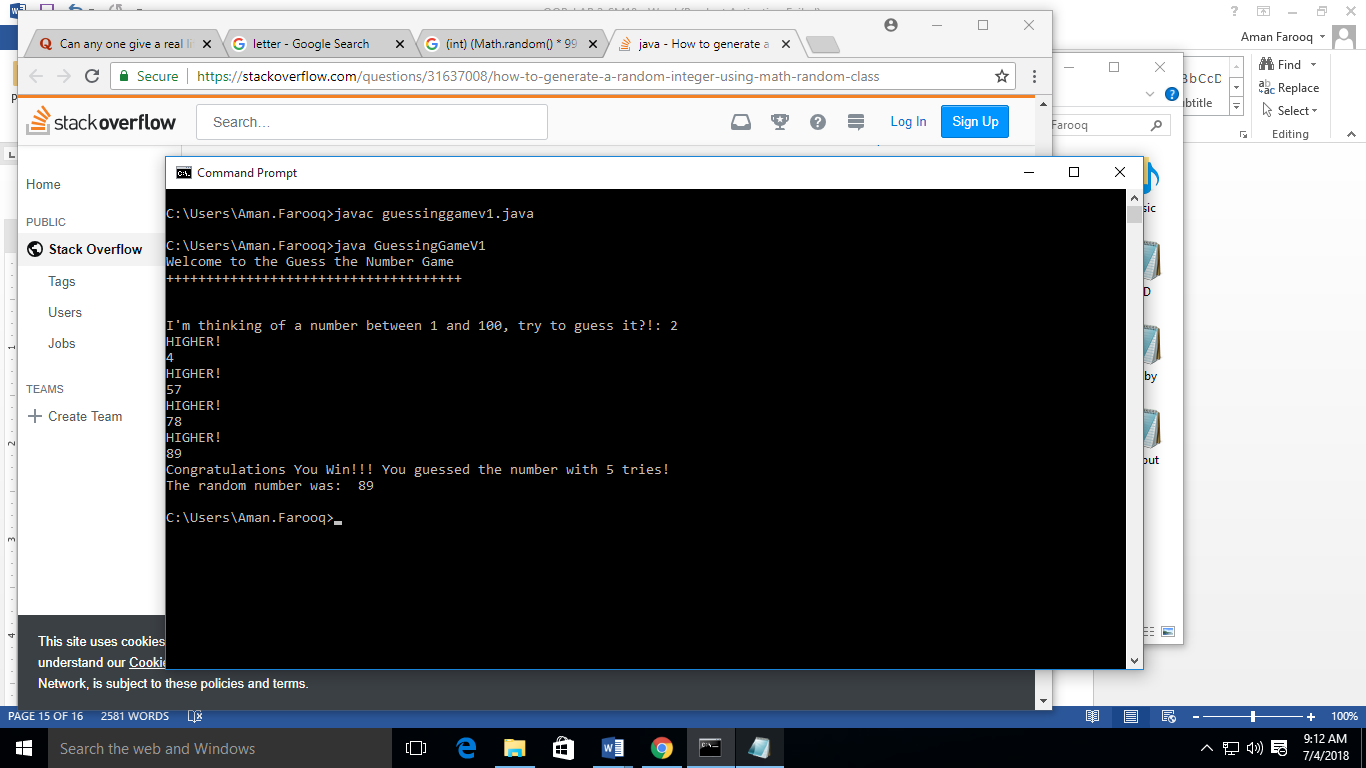
**Practice task 3**

Write a program that takes input from the user and tells if the input is vowel or consonant. If the input is invalid, it shows an error. (Hint: use ASCII codes, .charAt(0))



**Practice task 4**

*(Game: Guess the Number?)* Write a program that prompts user to guess the number that the system is thinking between 1 and 100. The program should then tell the user whether the guesses number was correct or not and also the number of attempts that the user made. ***(Hint: Use math functions)***



**Practice task 5**

Suppose that a person take loan of 200000 from bank and is bound to pay 23% every year. After 3 years how much loan will be paid?

**Practice task 6**

**Wind-chill Temperature**

How cold is it outside? The temperature alone is not enough to provide the answer. Other factors including wind speed, relative humidity, and sunshine play important roles in determining coldness outside.

In 2001, the National Weather Service (NWS) implemented the new wind-chill temperature to measure the coldness using temperature and wind speed. The formula is

**twc = 35.74 + 0.6215ta - 35.75v0.16 + 0.4275tav0.16**

where ta is the outside temperature measured in degrees Fahrenheit and v is the speed measured in miles per hour. twc is the wind-chill temperature. The formula cannot be used for wind speeds below 2 mph or temperatures below -58 oF or above 41oF.

Write a program that prompts the user to enter a temperature between -58 oF and 41oF and a wind speed greater than or equal to 2 and displays the wind-chill temperature. ***(Hint:Use Math.pow(a, b) to compute v0.16.)***

**Further Practice(Home Task)**

* 1. What does this code print?

class Pattern

{

public static void main(String[] args)

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5 - i; j++)

{

System.out.print(" ");

}

for (int k = 0; k <= i; k++)

{

System.out.print(k + " ");

}

System.out.println();

}

}

}

* 1. Find the errors in the following program to produce desired output.

class StarPyramid

public static void main(String args)

{

for (int i = 0; i < 5; i++)

for ( j = 0; j < 5 - i; j++)

{

System.out.print(" ");

}

for (int k = 0; k <= i; k++)

{

System.out.print("\* ");

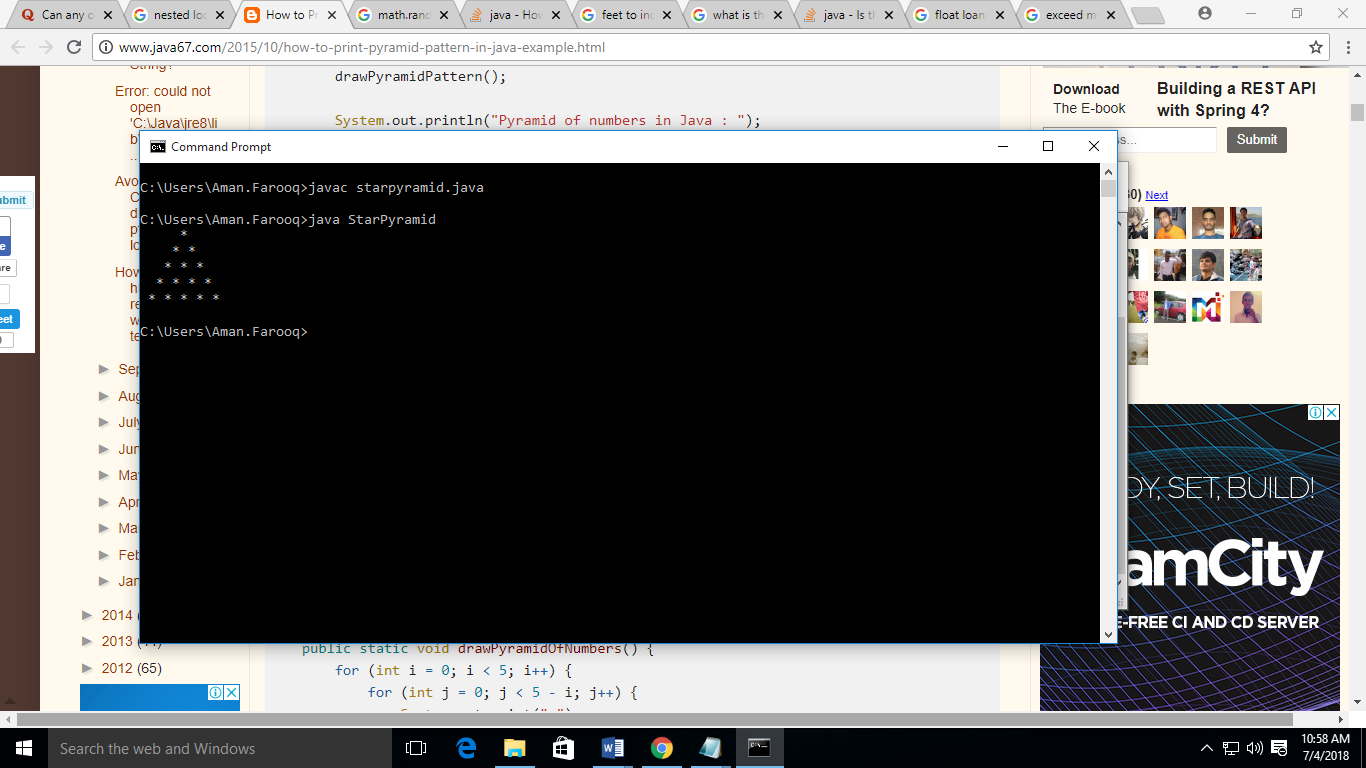
}

System.out.println();

}

}

}



2. **Game: heads or tails**

Write a program that lets the user guess whether the flip of a coin results in heads or tails. The program randomly generates an integer 0 or 1, which represents head or tail. The program prompts the user to enter a guess and reports whether the guess is correct or incorrect.

3. **Tax Calculation**

Write a program to calculate income tax by using the following rules.

**Inputs** to be taken from user (Income, age)

* If age is less than 60
* If income is less than or equals to 250000 tax percentage is 0.
* If income is greater than or equals to 250001 and less than or equals to 500000 tax percentage is 10.
* If income is greater than or equals to 500001 and less than or equals to 1000000 tax percentage is 20.
* If income is greater than or equals to 1000001 tax percentage is 30.
* If age is greater than or equals to 60 and less than 80
* If income is less than or equals to 300000 tax percentage is 0.
* If income is greater than or equals to 300001 and less than or equals to 500000 tax percentage is 10.
* If income is greater than or equals to 500001 and less than or equals to 1000000 tax percentage is 20.
* If income is greater than or equals to 1000001 tax percentage is 30.
* If age is equals to or greater than 80
* If income is less than or equals to 500000 tax percentage is 0.
* If income is greater than or equals to 500001 and less than or equals to 1000000 tax percentage is 20.
* If income is greater than or equals to 1000001 tax percentage is 30.

4. Suppose the tuition fee of a university is Rs.122600 this year and the fee increases 6.5% every year. In how many years will the fee be doubled?

5. Write a Java program in which user input month number and year to find the number of days in a month.