Assignment 1

1. What is JDK? JRE? JVM?

JDK represents for Java Development Kit, it includes Java Runtime Environment, an interpreter and loader(java), a compiler(javac), an archiver(jar), a documentation generator(java doc) and other tools needed in java development.

JRE stands for Java Runtime Environment, it consists of Java Virtual Machine(JVM), core classes, and supporting files. JRE offers the minimum requirement to run a java application

JVM stands for Java Virtual Machine, and is responsible for executing java programs line by line. It consists of class loader, Runtime memory, Execution Engine. Class loader is for loading classes, Runtime memory offers memory space to store data and Execution Engine is for executing bytecode.

2. What is java compiler?

Java compiler is a compiler for Java program, the output from java compiler is mainly java class files containing java bytecode.

3. Why is java platform independent?

In java, programs are compiled into byte code and the byte code is platform independent, since the byte code is executed by JVM. JVM is platform dependent.

4. What is IDE? Why is it important for developers?

IDE stands for Integrated Development Environment, it offers editing source code, building executables, and debugging. Enables programmers to consolidate different aspects of writing a computer program. And it improves the efficiency of developing an application.

5. Is java case sensitive?

Yes, it is. Upper case of letter and lower case of letter matters in java

6. What do the following key words do?
   static, final, public, private, void, null, package, Class, new

Static keyword: means the particular field belongs to the class not an object, all the static fields are visible to object instances

Final keyword: when it is used for particular class, it means that class can't be inheritance. When it is used for particular variable, it means that the value of the variable can't be modified. When it is used for particular method, it means that the method can't be override.

Public keyword: it means that class, method or variable can be visible in global world.

Private keyword: it means that field only can be visible in the same class.

Void Keyword: it is one of the return types, it stands for nothing returned after calling that particular method.

Null keyword: it is a specific reference, when one reference of the instance points to null, it means that object hasn't been created on the heap.

Package keyword: it means the package that this file's class or classes belong to is specified with the package keyword

Class keyword: it is used to declare a new java class

New keyword: it is used to create an object instance

7. What is primitive type and reference type?

Java has total two types: primitive type and reference type

Primitive type contains: byte, short, int, long, float, double, boolean, char

Reference type: all the other types are reference type

8. Is parameter passed by value or reference?

In java, parameter is passed by value, for example when you call a method and pass an object to its parameter, if you change the value of the parameter, it makes no effect on the original object.

9. What is the output: System.out.println(1 > 0 : “A”:”B”);

Compile error

10. How to define constants in java?

For example, static final var = 1;

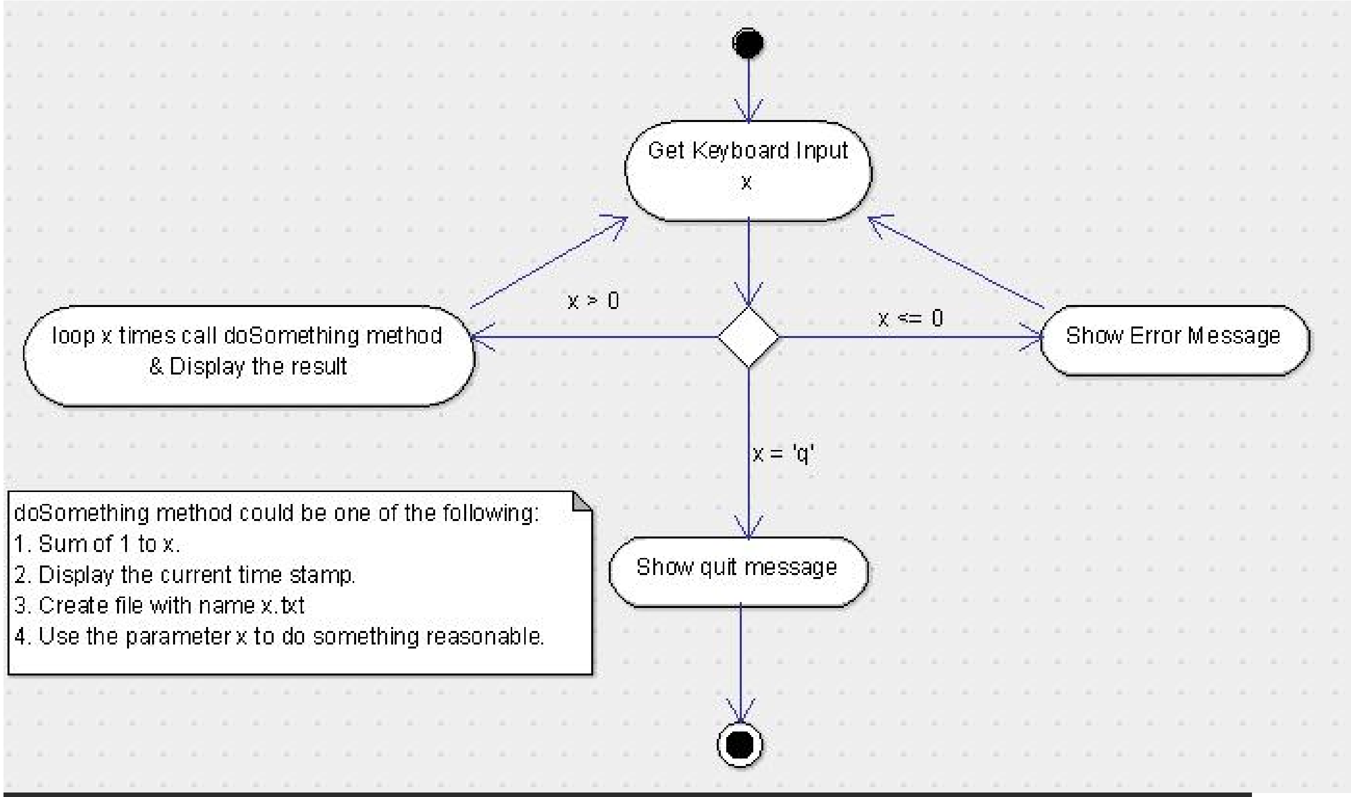
11. What is String? Is it primitive type?

String is a reference type, and it is used to define String objects.

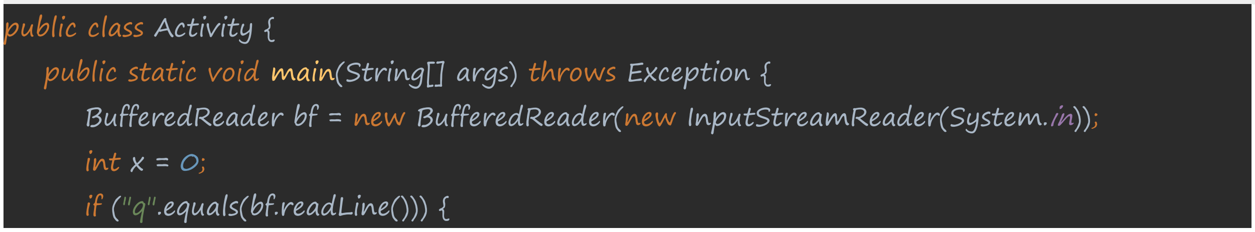
12. How to check if a String is representing a number?

Using regular expression or linear scan the string to check if each character is a number

13. Write a program to implement the following activity diagram:



doSomething method could be one of the following:
1. Sum of 1 to x.
2. Display the current time stamp.
3. Create file with name x.txt
4. Use the parameter x to do something reasonable.



```
public class Activity {
    public static void main(String[] args) throws Exception {
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        int x = 0;
        if ("q".equals(bf.readLine())) {
            System.out.println("quit");
            return;
        }
        try {
            x = Integer.valueOf(bf.readLine());
        } catch (IOException e) {
            e.printStackTrace();
        }
        while (x > 0) {
            System.out.println(doSomething(x));
            x--;
        }
        if (x <= 0) {
            throw new Exception("Error Message");
        }
    }
    private static int doSomething(int x) {
        int res = 0;
        for (int i = 1; i <= x; i++) {
            res += i;
        }
        return res;
    }
}
```

14. Write a program to merge two array of int.

```
public class Merge {
    public static void main(String[] args) {
        Merge m = new Merge();
        System.out.println(Arrays.toString(m.merge(new int[]{1,2}, new int[]{4})));
    }
    public int[] merge(int[] arr1, int[] arr2) {
        if (arr1 == null) {
            return arr2;
        } else if (arr2 == null) {
            return arr1;
        }
        int one = arr1.length;
        int two = arr2.length;
        int[] res = new int[one + two];
        int i = 0;
        int j = 0;
        int k = 0;
        while (i < one + two) {
            if (j < one) {
                res[i] = arr1[j];
                i++;
                j++;
            }
            if (k < two) {
                res[i] = arr2[k];
                i++;
                k++;
            }
        }
        return res;
    }
}
```

15. Write a program to find the second largest number inside an array of int.

```
public class Second {
    public static void main(String[] args) {
        Second s = new Second();
        int r = s.second(new int[]{1, 7, 4, 5});
        System.out.println(r);
    }
    private int second(int[] arr) {
        if (arr == null || arr.length < 2) {
            return Integer.MIN_VALUE;
        }
        int fir = arr[0];
        int sec = arr[0];
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] > fir) {
                fir = arr[i];
            } else if (arr[i] > sec) {
                sec = arr[i];
            }
        }
        return sec;
    }
}
```