Assignment 1

1. What is JDK? JRE? JVM?
    - JDK is a software development environment used for making applets and Java applications.
    - JRE is Java Runtime Environment. if you want to run Java program you need JRE. If you are not a programmer, you don't need to install JDK
    - JVM is Java Virtual Machine. JVM is an engine that provides a runtime environment to drive the Java Code or applications. It converts Java bytecode into machine language. JVM is a part of Java Run Environment (JRE).
      In many other programming languages, the compiler produces machine code for a specific system. However, Java compiler produces code for a virtual machine which is called as JVM.
2. What is java compiler?
   The most common form of output from a Java compiler is Java class files containing platform-neutral Java bytecode.
   The Java virtual machine (JVM) loads the class files and either interprets the bytecode or just-in-time compiles it to machine code and then possibly optimizes it using dynamic compilation.
3. Why is java platform independent?
    - The java compiled code(byte code) can run on all operating systems.
    - Whenever, a program is written in JAVA, the javac compiles it.
    - The result of the JAVA compiler is the .class file or the bytecode and not the machine native code (unlike C compiler).
    - The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.
    - And finally program runs to give the desired output.
4. What is IDE? Why is it important for developers?
   Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program. IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging.
5. Is java case sensitive?
   Java is case-sensitive because it uses a C-style syntax. Case sensitivity is useful because it lets you infer what a name means based on it's case. For example, the Java standard for class names is uppercasing the first letter of each word
6. What do the following key words do?
   static, final, public, private, void, null, package, Class, new
    - Static
        - Static variable:
          it is a not good choice to go for instance variable.

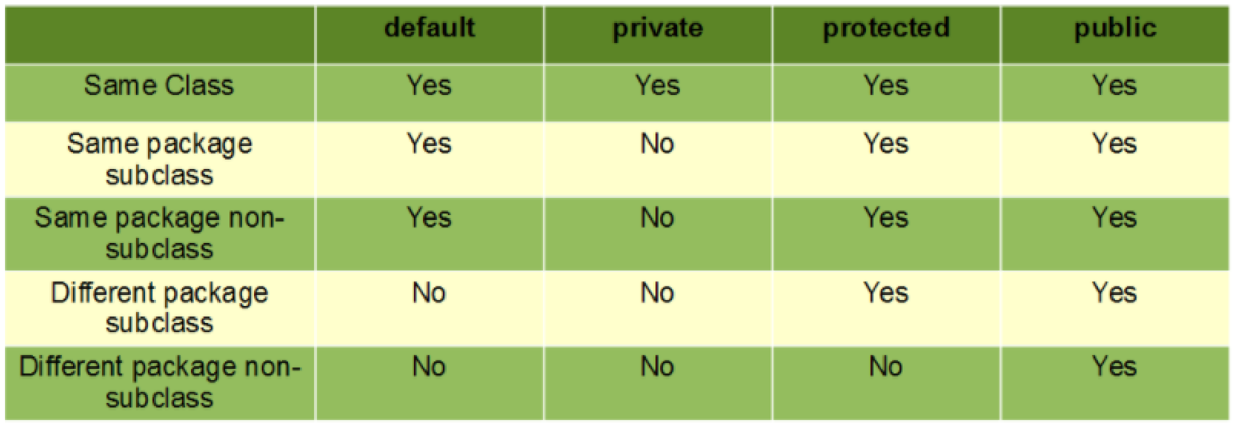
    Whenever we declare variable as static, then at the class level a single variable is created which is shared with the objects.

    - Static blocks:
      If you need to do computation in order to initialize your static variables, you can declare a static block that gets executed exactly once, when the class is first loaded.

    - static method:
      They can only directly call other static methods.
      They can only directly access static data.
      They cannot refer to this or super in any way.
      (non-static method can access static variable)

    - Static nested(inner) classes:
      A static nested class may be instantiated without instantiating its outer class.
      Inner classes can access both static and non-static members of the outer class. A static class can access only the static members of the outer class.
      https://www.geeksforgeeks.org/static-class-in-java/

- Final
    - Final variable: constant variables
    - Final methods: prevent overriding
    - Final classes: prevent inheritance
- Public , private (Access Modifiers)



	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

- Void: a method should not have a return value

- null
  null is the default value of any reference type variable which is not initialized at the time of declaration. The object reference to nowhere. null can only be assigned to reference types.

- Package
  Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

  - Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
  - Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
  - Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
  - Packages can be considered as data encapsulation (or data-hiding).
- Class
  A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

  - Modifiers : A class can be public or has default access (Refer this for details).
  - class keyword: class keyword is used to create a class.
  - Class name: The name should begin with a initial letter (capitalized by convention).
  - Superclass(if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
  - Interfaces(if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
  - Body: The class body surrounded by braces, { }.

  An object consists of :

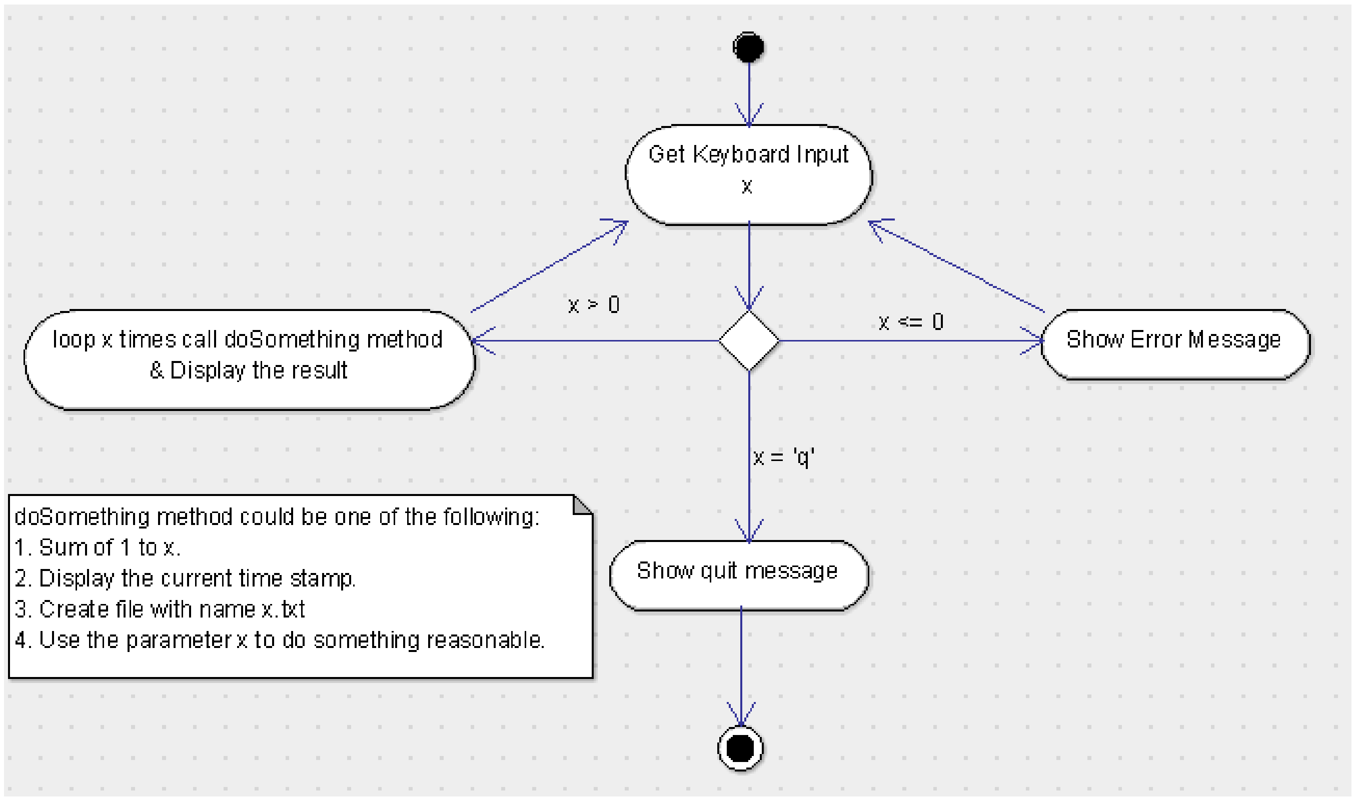
  - State/attributes : It is represented by attributes of an object. It also reflects the properties of an object.
  - Behavior/method : It is represented by methods of an object. It also reflects the response of an object with other objects.
  - Identity(name) : It gives a unique name to an object and enables one object to interact with other objects.
- New
  Create an object
  Can be used for Declaration and Instantiation

7. What is primitive type and reference type?
   Types in Java are divided into two categories—primitive types and reference types. The primitive types are boolean , byte , char , short , int , long , float and double . All other types are reference types, so classes, which specify the types of objects, are reference types.
8. Is parameter passed by value or reference?
   - Java always passes parameter variables by value.(primitive type)
   - Object references are passed by value(reference type)

   For immutable classes(Integer, Double, Float, Long, Boolean, BigDecimal, and of course the very well known String class)
   Even you assign new value to immutable classes. The address is new. So you can't change the variable by method.
   https://www.infoworld.com/article/3512039/does-java-pass-by-reference-or-pass-by-value.html
9. What is the output: System.out.println(1 > 0 : “A”:”B”);
   “A”
10. How to define constants in java?
    Public final static
11. What is String? Is it primitive type?
    String is an immutable object; it isn't a primitive type.
12. How to check if a String is representing a number?

```
private boolean IsInt_ByException(String str)
 {
    try
    {
       Integer.parseInt(str);
       return true;
    }
    catch(NumberFormatException nfe)
    {
       return false;
    }
 }
```

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.

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

13.
```
package com.antra;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle errors


public class Sample {
	static void doSomething(int num) {
		int s = 0;
		Timestamp timestamp = new Timestamp(System.currentTimeMillis());
		for (int i = 1; i <= num; i++)
			s+=i;

		System.out.println(s);
		System.out.println(timestamp);
		try {
	      File myObj = new File(String.valueOf(num) +".txt");
	      if (myObj.createNewFile()) {
	        System.out.println("File created: " + myObj.getName());
	      } else {
	        System.out.println("File already exists.");
```

```
            }
        } catch (IOException e) {
          System.out.println("An error occurred.");
          e.printStackTrace();
        }

    }
    public static void main(String[] args) {
           System.out.print("Enter a number or q to quit:");
      while (true) {
           Scanner scanner = new Scanner(System. in);
          String inputString = scanner.nextLine();
           int num = 0;
           try{
              num = Integer.parseInt(inputString);
              if(num<=0) {
                  System.out.println("wrong input");
                  continue;
              }
          }
          catch(NumberFormatException nfe){
           if (inputString.equals("q")) {
                  System.out.println("quit");
                          break;
                  }else {
                          System.out.println("wrong input");
                  }
          }

           doSomething(num);

           }
     }
}
```

14. (merge sort to merge)

```
package com.antra;

public class Merge {
    static int[] merge(int[] a1,int[] a2) {
            int []ans = new int[a1.length+a2.length];
            int j=0,k=0;
            for (int i = 0; i < ans.length; i++) {
                    if(k>=a2.length || j<a1.length && a1[j]<a2[k]) {
                            ans[i]=a1[j];
                            j++;
                    }else {
                            ans[i]=a2[k];
                            k++;
                    }
            }
            return ans;
    }
    static int[] sort(int[] arr) {
```

```
		int size1 = arr.length/2;
		int size2 = arr.length-size1;
		if(arr.length>1) {
			int arr1[] = new int[size1];
			int arr2[] = new int[size2];
			System.arraycopy(arr,0,arr1,0,size1);
			System.arraycopy(arr,size1,arr2,0,size2);
			arr1 = sort(arr1);
			arr2 = sort(arr2);
			int[] ans = merge(arr1,arr2);
			return ans;
		}
		return arr;
	}
	public static void main(String[] args) {
		int arr1[] = {9,11,7,6,3,20};
		int arr2[] = {30,1,5,3,10,6,16,15};
		int ans[] = new int[arr1.length+arr2.length];
		System.arraycopy(arr1, 0, ans, 0, arr1.length);
		System.arraycopy(arr2, 0, ans, arr1.length, arr2.length);
		ans = sort(ans);
		for (int i : ans) {
			System.out.println(i);
		}
	}
}
```
15.
```
package com.antra;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class SecondLargest {
	public static void main(String[] args) {
		Integer array[] = {30,1,5,3,10,6,16,15};
		List<Integer> list = Arrays.asList(array);
		Integer maxVal = Collections.max(list);
		Integer maxIdx = list.indexOf(maxVal);
		list.set(maxIdx,Integer.MIN_VALUE );
		System.out.println(Collections.max(list));
	}
}
```