Test 1

1) What is an Object ?

An object is an instance of a class that can be used to easily access instance members of a class.
Objects are important when multiple copies of a class are required for manipulation or any of the CRUD operations.
An object created is stored in the heap memory and is accessed using references.
Objects can be created in methods, constructors or blocks.

A general declaration of an object :

className referenceName = **new** className();

Eg : Game game = new Game();

2) What is a Class ?

A class is a blueprint which can be used to create objects.
It sets certain functionalities and properties of a particular class.
For eg, a class **Mobile** would have properties(variables) of **brand, modelNo, price,** etc. and functionalities such as **calling, watching, gaming,** etc.
All of these together create a class.
A class consists of states and behaviors.

States being the properties or variables, and
Behaviors being the functionalities or methods.

A general declaration of a class :

```
class MobilePhone //class ClassName
    {
        //instance variables
        int modelNo;
        String brand;

        //constructors(optional)
        void Mobile() {
                //initialization of variables
            }

        //methods or functions
        void display() {
                //print details
            }
```

```
        public static void main(String[] args) {
                //main method executable statements
            }
} //end of class
```

The class name is declared with the keyword ‘class’ followed by the class name in PascalCase with the first alphabet and suffix represented in uppercase letters.
A class created by a user can also be referred to as a user-defined primitive data type.

3) What are variables ? Why use them ?

Variables are containers used to store data. Data types are used to refer to the type of data being stored. A variable can store the type of value or data based on the data type used for its declaration.
For eg, a variable

int num=10; // the variable num can only store whole numbers that are integer values.

Variables are important to store multiple data.

There are 4 types of variables :

- Local Variables - these are variables that are only applicable to the method it is declared in and no other method can access these variables.
- Instance Variables - these are variables that are declared outside a method but inside the class and can be accessed by any of the methods declared in that particular class. These variables however cannot be accessed directly from the main method since the main method is a static method and therefore require object creation to access them from the main method.
- Static Variables - these are similar to instance variables. The difference here is that these variables can be accessed directly from the main method and do not require object creation for accessing these variables.
- Parametric Variables - these variables are the ones passed as parameters to a method. These are also considered as local variables that can only be accessed inside the method they are passed to. Eg : String[] args declared in the main method.

General declaration of a variable :

```
dataType variableName = data/value;
```

4) What are instance variables ?

Instance variables are those variables that are declared outside a method but inside the class and can be accessed by any of the methods declared in that particular class. These variables however cannot be accessed directly from the main method since the main method is a static method and therefore require object creation to access them from the main method.

Eg :

```
class Employee {
        int empID;
        String name;
        long phoneNum;
}
```

5) What is a static variable ?

A static variable is one declared with the keyword ‘static’. They are similar to instance variables in the sense that they are declared in a class outside a method. The difference here is that these variables can be accessed directly from the main method and do not require object creation for accessing these variables and they can be accessed using class name.

Eg:

```
class Employee {
        static int id;
        static String name;
}
```

6) What is the difference between class variables and instance variables ?

- A class variable or a static variable is declared with a static keyword and instance variables do not require the static keyword.
- A class variable can be accessed only from static methods while the instance variable can be accessed from any method in a class other than static methods.
- A class variable can only be accessed in another class with a ‘className.variable’ format however an instance variable can be accessed from another class using an object reference with a ‘objectReference.variable’ format.

7) How will we create an object ?

An object creation requires the class name, a reference variable and a **new** operator.

```
className objReferenceVariable = new className();
```

The above declaration creates a new object for the defined className and this object reference can now be used to access variables and methods from other classes, etc.

Eg :

```
class Student {
        int usn;
        String name;
        char gender;

        public static void main (String[] args) {
```

```
        Student stud = new Student();   //stud is the object used as a reference to class Student
        stud.name = “Diya”;
        stud.usn=1234;
        stud.gender=’F’;
        System.out.println(stud.name+” “ +stud.usn+” “+stud.gender);
    }
}
```

8) What is a method ?

A method is a procedure or a set of instructions to perform a particular task. It can be of different types :

- With return type, with arguments
- With return type, without arguments
- Without return type, with arguments
- Without return type, without arguments

General declaration of a method :

```
returnType methodName (parameter list with data types) {
    //executable statements
}
```

9) What is method overloading ?

Method overloading occurs when a class consists of methods with the same name but different signatures, i.e., with different arguments or different return types.
Method overloading is usually done with constructors using different numbers and types of parameters or arguments. This is also known as constructor overloading.

Eg:

```
void display() {
    System.out.println(“Display function 1”);
}

void display(int n) {
    System.out.println(“Display function “+n);
}

int display() {
    return 4;
}
```

10) Program for method overloading

Attached Calculator.java file along with output screenshot

11) Program to print month of the year

Attached MonthOfYear.java along with output screenshot

12) Program for factorial

Attached Factorial.java along with output screenshot.