Topics to cover:

1. Object-oriented programming
2. Classes and objects
3. Constructors

1. Object-oriented programming

1.1 What is object-oriented programming

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Note: The "Don't Repeat Yourself" (DRY) principle is about reducing the repetition of code. You should extract out the codes that are common for the application, and place them at a single place and reuse them instead of repeating it.

2. Classes and objects

2.1 Classes:

A class in Java is a set of objects which shares common characteristics/ behavior and common properties/ attributes.

or,

It is a user-defined blueprint or prototype from which objects are created.

For example, Student is a class while a particular student named Ravi is an object.

Syntax for a class in Java lools like follows:

Syntax:

```
access_modifier class <class_name>
{
   data member;
   method;
}
```

Example:

```
public class Employee
{
   int age;  //data member

   void show(){
        System.out.println("The age is: "+age)      //method
   }
}
```

2.2 Objects:

An object in Java is a basic unit of Object-Oriented Programming and represents real-life entities.

Objects are the instances of a class that are created to use the attributes and methods of a class.

A typical Java program creates many objects, which as you know, interact by invoking methods.

Example:

An object for the Employee class from the previous example can be created as follows:

```
Employee e = new Employee();
```

3. Constructors

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling the constructor, memory for the object is allocated in the memory.

In other words, a constructor is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such cases, Java compiler provides a default constructor by default.

Example 1:

```
//Java program to overload constructors
class Student{
```

```
    int id;
    String name;
    int age;
    //creating two arg constructor
    Student(int i,String n){
    id = i;
    name = n;
    }

    //creating three arg constructor
    Student(int i,String n,int a){
    id = i;
    name = n;
    age=a;
    }
    void display(){System.out.println(id+" "+name+" "+age);
    }
}

class HelloWorld{
    public static void main(String args[]){
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student(222,"Aryan",25);
    s1.display();
    s2.display();
   }
}
```

Output:

```
111 Karan 0
222 Aryan 25
```

Example 2**:**

Write a program to form the following pattern using a square number >=4. Use two classes: (i) one with with the variables and display method that prints the desired pattern, and (ii) the other with the main method creating and using the object

```
class formPyramid{
   formPyramid(int n){
      int sqrt=(int)Math.sqrt(n);
      int num=1;
      for(int i=1;i<=sqrt;i++){
         for(int j=1;j<=i;j++){
            System.out.print((num++)+"\t");
         }
         System.out.println();
      }
      for(int i=1;i<=sqrt-1;i++){
         for(int j=1;j<=sqrt-i;j++){
            System.out.print((num++)+"\t");
         }
         System.out.println();
      }
   }
}

public class Main
{
        public static void main(String[] args) {
                formPyramid fp = new formPyramid(25);
        }
}
```

Output:

```
1
2       3
4       5       6
7       8       9       10
11      12      13      14      15
16      17      18      19
20      21      22
```

23 24
25