Day-7

Agenda

- OOPS
- Java Introduction
- Language Fundamentals
- Coding standards
- Introduction to Java API
- Array
- Strings

OOPS

- OOP stands for **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 System (OOPs)** is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism.

- It allows users to create objects they want and create methods to handle those objects.

- The basic concept of OOPs is to create objects, re-use them throughout the program, and manipulate these objects to get results.

- 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

- 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.

- OOPs Concepts:
  - Polymorphism
  - Inheritance
  - Encapsulation
  - Abstraction
  - Class
  - Object
  - Method
  - Message Passing

Java Introduction

- JAVA was developed by Sun Microsystems Inc in 1991, later acquired by Oracle Corporation.
- It was developed by James Gosling and Patrick Naughton. It is a simple programming language.
- Writing, compiling and debugging a program is easy in java. It helps to create modular programs and reusable code.
- The target of Java is to write a program once and then run this program on multiple operating systems.
- Java is defined by a specification and consists of a programming language, a compiler, core libraries and a runtime (Java virtual machine) The Java runtime allows software developers to write program code in other languages than the Java programming language which still runs on the Java virtual machine.
- The Java platform is usually associated with the Java virtual machine and the Java core libraries.

- The Java language was designed with the following properties:
  - Platform independent
  - Object-orientated programming language
  - Strongly-typed programming language
  - Interpreted and compiled language
  - Automatic memory management

Java terminology

- Java Virtual Machine (JVM)
- Bytecode
- Java Development Kit(JDK)
- Java Runtime Environment(JRE)

Java Virtual Machine (JVM)

- **Java Virtual Machine (JVM)** is a engine that provides runtime environment to drive the Java Code or applications.

- It converts Java bytecode into machines language. JVM is a part of Java Run Environment (JRE).

- The Java Virtual Machine is a program whose purpose is to execute other programs.

- The JVM has two primary functions: to allow Java programs to run on any device or operating system (known as the "Write once, run anywhere" principle), and to manage and optimize program memory.

Bytecode

- JDK compiles the java source code into bytecode so that it can be executed by JVM. The bytecode is saved in a .class file by compiler.
- Bytecode in Java is an intermediate machine-independent code. It is a set of instructions for Java Virtual Machine
- In general, bytecode is a code that lies between low-level and high-level language.
- The bytecode is not processed by the processor. It is processed by the Java Virtual Machine (JVM).
- The job of the JVM is to call all the required resources to compile the Java program and make the bytecode independent. It is the biggest reason why java is known as a platform-independent language.
- The intermediate code can run on any of the platforms such as Windows, macOS, and Linux.

Java Development Kit(JDK)

- While explaining JVM and bytecode, I have used the term JDK. Let’s discuss about it.
- As the name suggests this is complete java development kit that includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc.
- In order to create, compile and run Java program you would need JDK installed on your computer.

Java Runtime Environment(JRE)

- JRE is a part of JDK which means that JDK includes JRE. When you have JRE installed on your system, you can run a java program however you won’t be able to compile it.
- JRE includes JVM, browser plugins and applets support. When you only need to run a java program on your computer, you would only need JRE.

- **JDK (Java Development Kit)**
  - JDK contains everything that will be required to ***develop and run*** Java application.
- **JRE (Java Runtime Environment)**
  - JRE contains everything required to ***run*** Java application which has already been compiled. It doesn’t contain the code library required to develop Java application.
- **JVM (Java Virtual Machine)**
  - JVM is a virtual machine which works on top of your operating system to provide a recommended environment for your compiled Java code. JVM only works with bytecode. Hence you need to compile your Java application(.java) so that it can be converted to bytecode format (also known as the .class file).
  - Which then will be used by JVM to run an application. JVM only provide the environment needed to executed Java Bytecode.

Coding standards

- Naming Conventions
- Curly Braces
- Indentation
- White Space
- Comments

Introduction to Java API

- A package in Java is used to group related classes. Think of it as **a folder in a file directory**. A programmer can make use of various API tools to make its program easier and simpler.
- API is an abbreviation for Application Programming Interface which is a collection of communication protocols and subroutines used by various programs to communicate between them.
- A programmer can make use of various API tools to make its program easier and simpler. Also, an API facilitates the programmers with an efficient way to develop their software programs.
- The Java API added to the JDK explains the function of every element. In Java programming, several components are pre-created as well as widely used.

- Hence, the programmer can make use of a prewritten program with the Java API. After mentioning the obtainable API classes as well as packages, the programmer quickly creates the required program classes and packages to get executed.
- The Java API is a vital element of the JDK and identifies features of every element. Although Programming in Java, the component is already produced and done it. Through Java, the API coder can simply make use of the pre-written program.
- The programmers initially declare the classes and packages; then, this coder can simply use the application program of classes and packages to get executed.

- **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

- **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

- **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

- **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

BASIC JAVA SYNTAX

- A Java program is a collection of objects, and these objects communicate through method calls to each other to work together. Here is a brief discussion on the Classes and Objects, Method, Instance variables, syntax, and semantics of Java.

- About Java programs, it is very important to keep in mind the following points.
  - Case Sensitivity
  - Class Names
  - Method Names
  - Program File Name
  - public static void main(String args[])

Basic terminologies in Java

- **1. Object**: The object is an instance of a class, have Behavior and state.

  Example: A car is an object whose **states** are: brand, colour, number-plate.

  Behaviour**:** Running on the road.

- **2. Class**: The class is a blueprint(plan) of class objects and status.

  Example: Blueprint of the house is class.

- **3. Method**: The behaviour of an object is the method.

  Example: The fuel indicator indicates the amount of fuel left in the car.

- **4. Instance variables**: Every object has its own unique set of instance variables. The state of an object is generally created by the values that are assigned to these instance variables.

  Example: Steps to compile and run a java program in a console

```
public class MyFirstJavaProgram {

   /* This is my first java program.
    * This will print 'Hello World' as the output
    */

   public static void main(String []args) {
      System.out.println("Hello World"); // prints Hello World
   }
}
```

Java Comments

- Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

- Single-line comments start with two forward slashes (//).

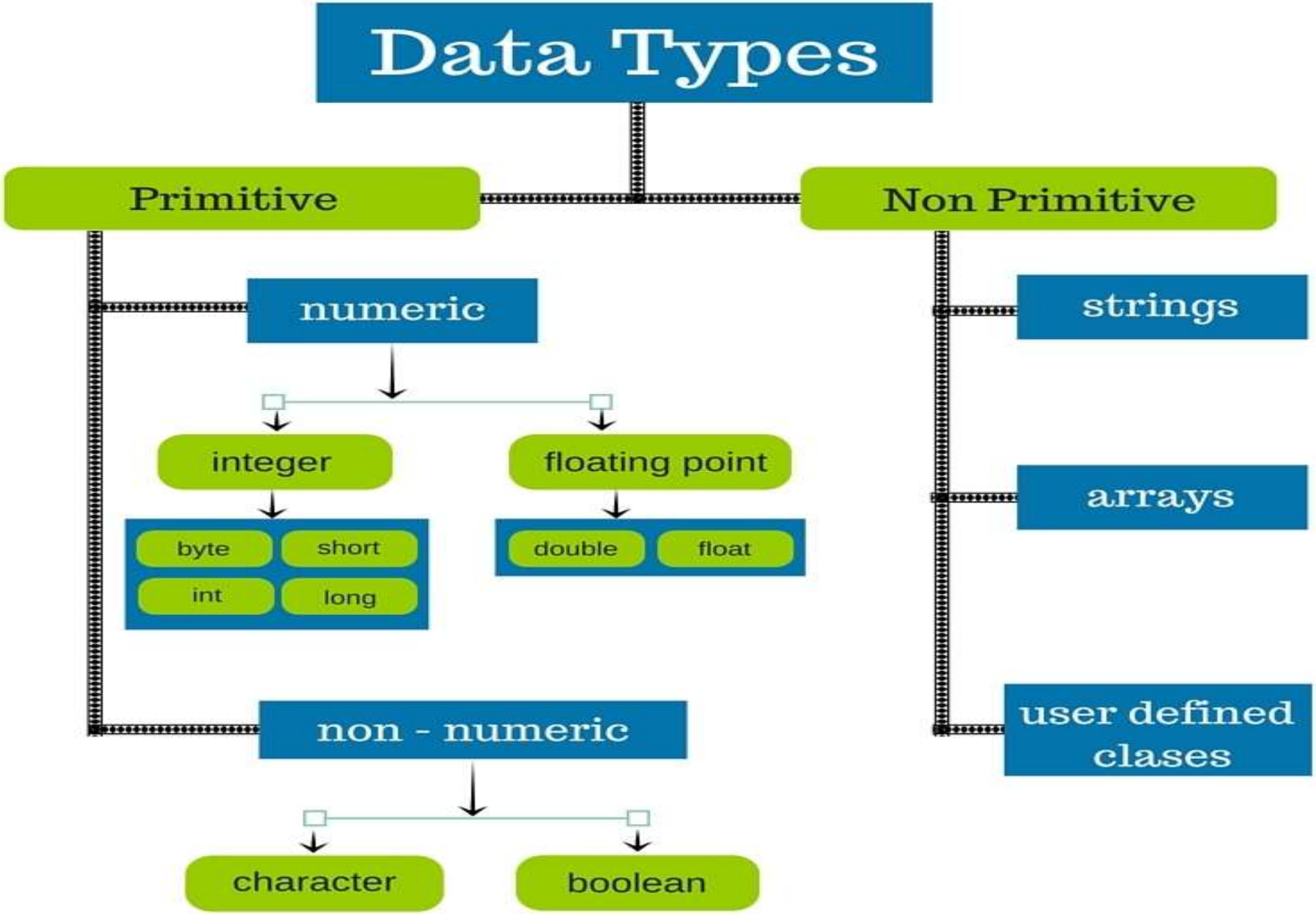
- Any text between // and the end of the line is ignored by Java (will not be executed).

- Multi-line comments start with /* and ends with */.

- Any text between /* and */ will be ignored by Java.

Java Variables

- Variables are containers for storing data values.
- type variable = value;
- In Java, there are different types of variables, for example:
  - String - stores text, such as "Hello". String values are surrounded by double quotes
  - int - stores integers (whole numbers), without decimals, such as 123 or -123
  - float - stores floating point numbers, with decimals, such as 19.99 or -19.99
  - char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
  - Boolean - stores values with two states: true or false


Data Types
Primitive
Non Primitive
numeric
integer
floating point
byte
short
int
long
double
float
non - numeric
character
boolean
strings
arrays
user defined clases

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Lets Practice……

Java Type Casting

- Widening Casting (automatically) - converting a smaller type to a larger type size

  byte -> short -> char -> int -> long -> float -> double

  double myDouble = myInt; // Automatic casting: int to double

- Narrowing Casting (manually) - converting a larger type to a smaller size type

  double -> float -> long -> int -> char -> short -> byte

  int myInt = (int) myDouble; // Manual casting: double to int

Lets Practice……

Object and Classes

Class

- **Class** are a blueprint or a set of instructions to build a specific type of object.
- It is a basic concept of Object-Oriented Programming which revolve around the real-life entities.
- Class in Java determines how an object will behave and what the object will contain.
- It represents the set of properties or methods that are common to all objects of one type.
- Syntax:-

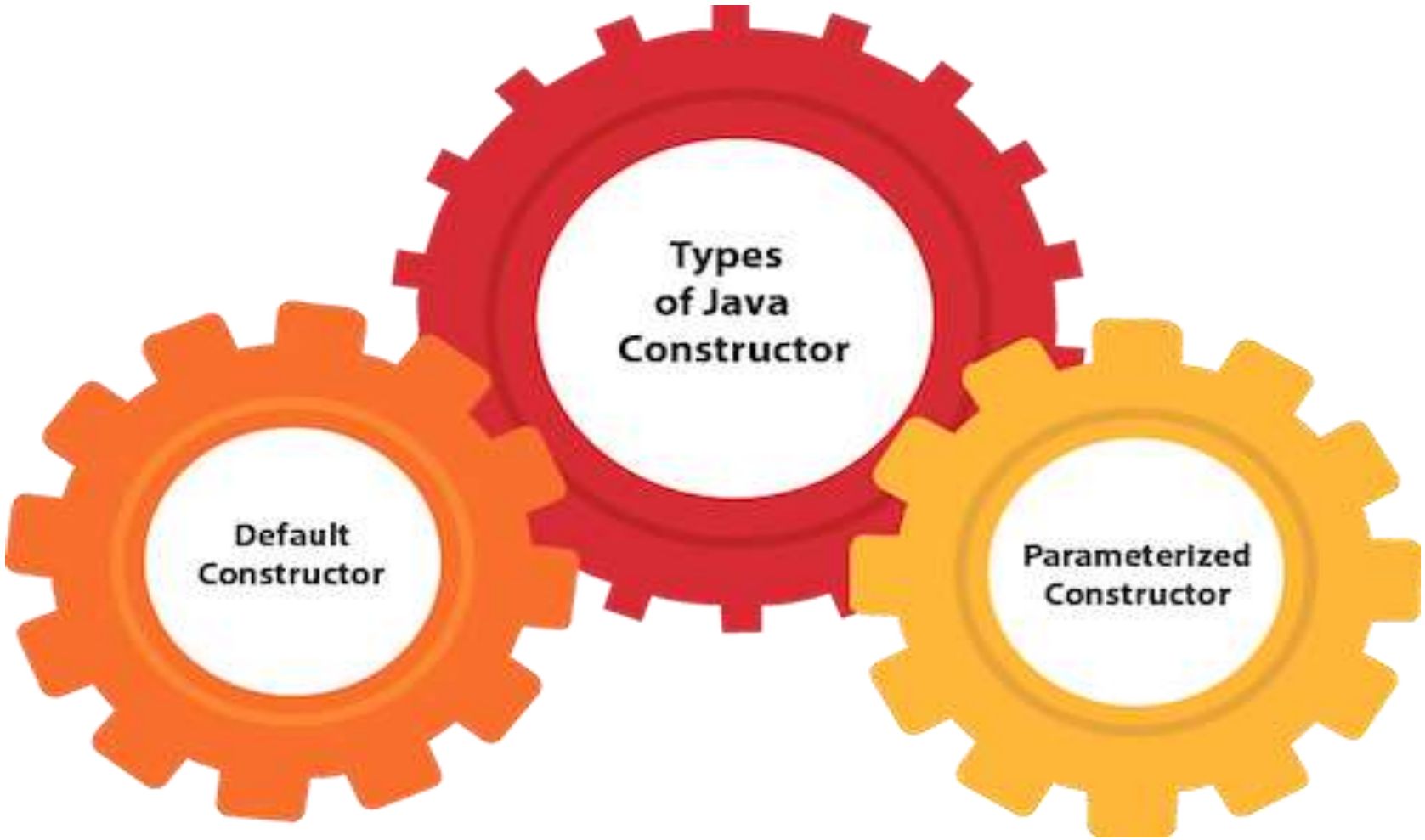
```
class <class_name>{
    field;
    method;
  }
```

- A class can contain any of the following variable types.
  - **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
  - **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
  - **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

- A class can have any number of methods to access the value of various kinds of methods.

➢ Constructors:-

- Constructors are used to initialize the object's state. Like methods, a constructor also contains collection of statements(i.e. instructions) that are executed at time of Object creation.
- A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.
- Note that the constructor name must match the class name, and it cannot have a return type (like void).
- Also note that the constructor is called when the object is created.
- All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you. However, then you are not able to set initial values for object attributes.


Types
of Java
Constructor
Default
Constructor
Parameterized
Constructor

Default constructor

```
class Bike1{
 Bike1(){System.out.println("Bike is created");}
public static void main(String args[]){
Bike1 b=new Bike1();
}
}
```

Parameterized Constructor

```
class Student4{
    int id;
    String name;
        Student4(int i,String n){
                id = i;
                 name = n;
    }
    }
```

Lets Practice……

Objects

- **Object** is an instance of a class. An object in OOPS is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful.
- For example colour name, table, bag, barking. When you send a message to an object, you are asking the object to invoke or execute one of its methods as defined in the class.
- From a programming point of view, an object in OOPS can include a data structure, a variable, or a function.
- It has a memory location allocated. Java Objects are designed as class hierarchies.
- An entity that has state and behavior is known as an object

- An object has three characteristics:
  - **State:** represents the data (value) of an object.
  - **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
  - **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

- There are three steps when creating an object from a class −
  - **Declaration** − A variable declaration with a variable name with an object type.
  - **Instantiation** − The 'new' keyword is used to create the object.
  - **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

- Java provides five ways to create an object.
  - Using new Keyword
  - Using clone() method
  - Using newInstance() method of the Class class
  - Using newInstance() method of the Constructor class
  - Using Deserialization

Lets Practice……

Array

- An array refers to a data structure that contains homogeneous elements. This means that all the elements in the array are of the same data type.
- **Java array** is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location.
- It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.
- Array can contain primitives (int, char, etc.) as well as object (or non-primitive) references of a class depending on the definition of the array. In case of primitive data types, the actual values are stored in contiguous memory locations.

- To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference. Here is the syntax for declaring an array variable −

Syntax:

```
dataType[] arrayRefVar;   // preferred way.
dataType arrayRefVar[];  // works but not preferred way.
```

- Creating Arrays:

```
arrayRefVar = new dataType[arraySize];
```

Lets Practice……

Strings

- **Strings** in Java are Objects that are backed internally by a char array. Since arrays are immutable(cannot grow), Strings are immutable as well. Whenever a change to a String is made, an entirely new String is created.

  Syntax:

  <String_Type> <string_variable> = "<sequence_of_string>";

  char[] ch={'j','a','v','a','t','p','o','i','n','t'};

  String s=new String(ch);

- **Java String** class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

Lets Practice……