Class

Class is a representation of similar types of objects which have common properties. It is also an implementation of encapsulation. There are two main properties of a class, data members and member functions.

In Java, we have to write every statement within a class block. It is required.
Always keep the first letter of our class name as capital however it is optional but Java also follow this rule for the inbuilt class name. This rule increases the readability of the code.

We don't need to create the main function in every class. It’s not like we can't but it is idiocy because execution of a class and use of the class is different. Execution of class will contain main () method and used class don't require to have a main () method.

If we want to make any class of Java executable then we have to keep one function in a class which name will be main and it is required rule. In Java, no class can be executed without main function from JDK 1.7. But before 1.7 we could execute a class without main function and that was a bug.

The bytecode of every class is always stored in a separate .class file and the name of that **.class** file will be the same as your class name.

In Java, Every class is a child of Object class. The object is a topmost class in Java, it doesn't have any parent class.

#. We can use java access modifiers with Classes as well as Class variables and methods.

We are allowed to use only “**public**” or “**default**” access modifiers with java classes and interfaces. Nested interfaces and classes can have all access specifiers (private, protected, default, public).

1. If a class is “**public**” then we can access it from anywhere, i.e. from any other class located in any **other packages** etc.
2. We can have only **one** “public” class in a source file and file name should be same as the public class name.
3. If the class has “**default access**” then it can be accessed only from other classes in the same package.

If class is having default access then it will not be visible in other packages and hence **methods and variables** of the class will also be not visible.

#. A class which is not abstract is referred as Concrete class. It means the class does not have own abstract method.

If the portion of a byte code which is getting a memory at class loading time that can get the memory only ones in the life cycle of a class, then this concept is known as static

memory allocation.

The portion of a class which is getting a memory at runtime that can get the memory again and again in the life cycle of a class, this process is known as dynamic memory allocation.

A class that creates at run time called proxy class. It has no name. This process is also called making anonymous class. The memory which has been located at one time does not have any name, to reach the memory address we need a pointer and Java does not support pointer.

Data Members:

There are two types of data members in Java, Instance data members and Static data members.

Instance Data Member:

A variable which is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at run time when object (instance) is created. That is why, it is known as instance variable.

#. Instance data member is a part of object.

#. If the data members of a class are not static then by default they are instance data members.

#. Instance data member can only be used in any static function of the same class after creating the object inside that static function.

#. We should not initialize data member at class level. We must initialize them in constructor or create function for that.

Static Data Member: Static keyword in java is used a lot in java programming.

Java static keyword is used to create a Class level variable in java.

#. Static variables and methods are part of the class, not the instances of the class. All the Static things of a class always belongs to a class. They are treated as a properties of a class. All the Static data members are getting a memory at class loading time that means before execution of the main function.

#. All the Static data members of a class get the memory only ones in the life cycle of a class.

#. All the objects are sharing a same memory location for each static data member.

#. Java rule says, to use static data member in a same class use it direct don’t create object of a class. When we use static data member outside the class by creating class object then it also waste memory. So best is use the static data member directly “**ClassName.dataMember**”.

#. Static data members of a class can be used directly in any function of a class. But we can’t create static data member inside any method static or non-static;

#. Usually static variables are used with final keyword for common resources or constants that can be used by all the objects. If the static variable is not private, we can access it with ClassName.variableName

Member Function

Java programming language provides two kinds of methods, Instance methods and Class methods. Class methods also called static method. All functions, instance and static of a class are getting a memory at class loading time.

The difference between these two kinds of methods are that Instance methods require an instance before they can be invoked; Class Methods do not need instance to invoke.

Instance methods use dynamic binding also called late binding. Class methods use static also called early binding.

The JVM uses two different instructions to invoke these two types of methods. **InvokeVirtual** for Instance methods and **InvokeStatic** for class methods (Static Methods).

The Java Virtual Machine creates a new stack frame for Java methods it invokes.

The stack frame contains space from the methods local variables, its operand stack, and any other information required by a particular Virtual Machine requirement.

The size of the local variable and operand stack are calculated at the compiled time and placed into the class file, so the Virtual Machine knows just how much memory will be needed by the memory stack frame.

When it invokes a method it creates a stack frame of the proper size for that method.

Instance Member Function

#. Same as Instance variable, static method belong to object. All the instance things of a class always belongs to and object. They are treated as a properties of an object.

#. Instance member functions can only be used in any static function by the instance of the class inside that static function.

Static Member function:

Same as static variable, static method belong to class and not to class instances.

A static method can access only static variables of class and invoke only static methods of the class. To access instance things static methods needs reference id of a class.

The main () method that is the entry point of a java program itself is a static method.

Always keep those behavior of an object as a static function of a class which are not doing any task related to any object or in which we are not using instance Data member.

Static member functions of a class can be used directly in any function of a class. Never use Static things of a class via object.

Q. When to make Instance things?

Always keep those properties of an object as an instance data member of a class whose values are changing for each instance of an object.

Always keep those behavior of an object as an instance function of a class which are performing a task for an object only.

Q. When to make static thing?

Always keep those properties of an object as a Static data member of a class whose values are same for each object.

In Java we can call the function at class level to initialize data member of that class. If the data members are non-static then functions must be non-static. If the data members are static then function must be static. We don't need to create an object while initializing data member through method at class level.
In Java we can call the function at class level to initialize data member of that class. If the data members are non-static then functions must be non-static. If the data members are static then function must be static. We don't need to create an object while initializing data member through method at class level.

Variables

There are only two ways to access data from RAM through address and name.

Variables are just a name given to a memory location and that memory location is used to store data and fetch data.

By Lynda: There are two major types of Variables Primitives and Complex object.

Primitives:
Numeric (Integers and floating point decimals.
Single Character.
Boolean (true/false).

Referenced Variable Type:
Class
Interface
Enum
Array

Primitive variables are declared with 3 parts:
Data Type: Required.
Variable Name: Required.
Initial value: optional.

Complex variables are instance of a classes and it also declared in 3 parts. Data type, variable name, and then optionally the initialization. Initialization uses new keyword and class constructor.

Data Type tells 2 things about a variable.

1. How much memory the variable is going to take.

2. What kind of data is going to hold?

Other languages like C uses pointer but Java does not support pointers because pointers are complex and the developers wanted to make Java language simple.

Java has no signed, unsigned concept. In Java every data types are signed.

If we define a variable as a signed that means that very well can hold positive and negative both values.

If we define any variable unsigned that means that variable can hold in only positive values.

1. byte: 8 bit (1 byte) -128 ,127

2. short: 16 bit (2 byte)
-32,768 to 32,767

3. int 32 bit (4 byte)
-2 power 31 to 231 power 31-1

4. long: 64 bit (8 byte)
-2 power 63 to 2 power 63-1

5. float: 32 bit IEEF754

6. double: double precision 64 bit IEEE754

Every data type of class level variable has default value but not for local variable. Local variable must be initialized before the first use. JVM don't set default value for local variables.

#. In Java all the integer constant/literals are treated as a int.
show (10);

When we use literal long then we have to put capital "**L**" after the value. We avoid small "**l**" because it looks like 1 also.

#. Calculation result does not belongs to any data type. It is literal value and it will convert according to the data-type variable that is going to hold it.

#. In Java all the floating literals are treated as double.

Class level and Local level Variables:

Variables declared outside the function belong to the whole class and it called class variable or field. It can be referenced in any function within the class.

Variable declare inside a function are local to the function called local variable. And when the function is finished executing the variable goes away, it’s no longer available.

Whenever class level variable and local level variable as same name then this concept is called **data shadowing**.

Every data type of class level variable has default value but not for local variables. JVM don't set default value for local variables. Local variable must be initialized before the first use. “**Before first use means**” if local non-initialized variable is declared and not used yet then there will be no error. Compiler throws error when local variable is going to use.

New Operator

The "**new**" operator is used to allocate new memory dynamically. It only assign memory to instance class members. It stores inside heap area. When we calculate and store memory size it is called buffer.

In Java "**new**" operator is only used to allocating a memory for an **object** only.

Constructor

Constructor is the special member function of a class which are used to initialize an object.

#. The process, putting the values into the data members of an object while creating it, is known as object initialization.

#. If we want to perform any task only once in the life cycle of an object then we can put the code of that task inside the constructorfunction block.

#. Every constructor and non-static function has default implicitly "**this**" feature.

Rules:
R1: Constructor function name and class name function should be same.

R2: Constructor does not have permission to return anything because by default it returns reference ID of the current the object. We can return "blank return" (return) in Java constructor.

Advantages of constructor:

A1. The constructor function helps the new operator to create an object.

A2. If you want to provide certain resources like (database connection, GUI, etc.) then put the resources inside the constructor function.

#. There are two types of constructor in Java Parameterized and non- Parameterized constructor (Default Constructor).

#.If there is no constructor in a class then one non-parameterized (default constructor) is inserted into the class by the compiler.

#. We can create more than one constructor function in a class by applying method overloading.

#. If you want to initialize the data members of each object with a same values, then

always use default constructor and this concept is known as static initialization of non-static data member.

#. If we want to initialize the data member of each object with a different-different values then always use parametrized Constructor and this concept is known as dynamic initialization of non-static data members.

#. When we initialize variable at class level then compiler creates default constructor and put that initialization code inside default constructor. We can see this if we convert the byte code into Java file via de-compiler.

#. In Java we can also make a function with a same name as our class name but it is not recommended.

If "**new**" operator is behind the function name then it will find for **constructor** function. If the (**.**) is behind the function then it will look for member function.

#. Copy constructor creates the duplicate copy of some existent object. It is like Photostat machine.

#. We can also make constructor as private but we won't be able use it outside the class.

#. We can't make constructor as final.

When you set a method as final it means: "You don't want any class override it." But the constructor (according to the Java Language Specification) can't be overridden, so it is clean.

#. When you set a method as abstract it means: "The method doesn't have a body and it should be implemented in a child class." But the constructor is called implicitly when the new keyword is used so it can't lack a body.

#. When you set a method as static it means: "The method belongs to the class, not a particular object." But the constructor is implicitly called to initialize an object, so there is no purpose in having a static constructor.

Constructor Chaining

The process of calling a one constructor from another constructor with respect to a current object is known as Constructor Chaining.

Through this way, we can make single object and call multiple constructors.

Rules:

R1: Whenever we are achieving a constructor chaining using "this()" then it must be the

first line in any constructor.

R2: Whenever we are doing constructor chaining using "this()" then there must be at least one constructor which is not having a "this()" as a first line. Otherwise it will go on infinite function calling.

R3: Constructor chaining can be achieved in any order.

Advantage: Whenever we want to provide a multiple resources to an object while creating it then instead of putting the each resources in a single constructor we can make a separate constructor for each resources and then create their chain.

This:

#. "**this**" is the reference variable that holds the reference Id of current object.

#. "**this**" keyword cannot be used in any in static function because it does not create any object instance.

#. If you are passing "this" as an argument in any non-static function then it must be the first argument.

#. "**this**" only works with JDK8 and newer versions.

Static Block

Static Block is used to initialize the static data members dynamically.

Static Block is always executes at class loading time that means before the execution of Main function. Mostly it's used to create static resources when class is loaded.

If we want to perform any task only once in the life cycle of a class then we put the code of that task in a Static Block.

Whenever we initialize static data members at class level then JVM Implicitly put that codes inside static block.

Why we need static block when we have constructor and init Block?

Because Constructor depends on object and Init block depends on constructor. So to execute codes at class loading time we use static block.

We can't access non-static variables in static block. We can have multiple static blocks in a class, although it doesn't make much sense. Static block code is executed only once when class is loaded into memory.

Init Block

Init Block is always executed before any constructor. We can define any number of init block in a single class and they will execute in order we create.

If we have not done constructor chaining then the complete code of Init block is inserted as a first line in every constructor of a class.

If we have done constructor chaining then the complete code of a Init block is inserted as a first line in that constructor in which we have not used "this()" as a first line.

If we want to perform a common task in each constructor of our class then rather than putting the code of that task in each constructor, we can put that code of that task in Init Block.

“Object” Class:

In Java every class is child of Object class. Object is a top most class in Java, it doesn't have any parent class.

Total 11 methods are implicitly comes in an object class and they are very-very important methods.

The Object class, in the java.lang package, sits at the top of the class hierarchy tree. Every class is a descendant, direct or indirect, of the Object class. Every class you use or write inherits the instance methods of Object. You need not use any of these methods, but, if you choose to do so, you may need to override them with code that is specific to your class. The methods inherited from Object that are discussed in this section are:

1. clone():

```
protected Object clone() throws CloneNotSupportedException
```

Creates and returns a copy of this object.

2. equals(Object obj):

```
public boolean equals(Object obj)
```

Indicates whether some other object is "equal to" this one.

3. finalize():

```
protected void finalize() throws Throwable
```

Called by the garbage collector on an object when garbage collection determines that there are no more references to the object

4. getClass():

```
public final Class getClass()
```

Returns the Run-time class of an object.

5. hashCode():

```
public int hashCode()
```

Returns a hash code value for the object.

6. toString():

```
public String toString()
```

Returns a string representation of the object.

The notify, notifyAll, and wait methods of Object all play a part in synchronizing the activities of independently running threads in a program, which is discussed in a later lesson and won't be covered here. There are five of these methods:

```
7. public final native void notify()
8. public final native void notifyAll()
9. public final native void wait()
10. public final native void wait(long timeout)
11. public final native void wait(long timeout, int nanos)
static {}
```

Java Anthem Object:

Write Once Run Anywhere

In Java, everything is represented in the form of object except, primitive data type

The object is nothing but just memory area or a buffer in a heap area in which all the instance data member taking memory.

In Java, Objects are run time entities they always get the memory at runtime even in Java only objects are getting a memory at run-time.

Creating an object of a class is the same thing creating an instance of a class.

The reference ID of one object can be point into the any number of reference variable.

Whenever we try to print reference ID of any object then JVM prints a string rather than reference ID which is a combination of three things class name, @ symbol and hexadecimal representation of a # code of object.

In Java, there are 6 way to create Object. Java?

1. By new keyword
2. By newInstance() method
3. By clone() method
4. By deserialization
5. By factory method etc.

Basic Info and Tips

Java has inherited a complete syntex from the C language.

In Java, we have to write every statement within a class block. it is required.

Always keep the first letter of your class name as capital however it is optional but Java also follow this rule for the inbuilt class name. This rule increases the readability of the code.

We don't need to create the main function in every class. it's not like we can't but it is idiocy because execution of a class and use an of the class is different.

If you want to make any class of Java executable then you have to keep one function in a class which name will be main and it is required rule.

Source code of a Java Program must be store having an extension .java and it is required rule.

Empty .java file name is valid source name. We can compile by command " javac .java" and run it by " java ClassName".

Always keep the name of .java file name same as your class name however it is not required but it improves the file managing.

If I enter filename like this "filename.java" while saving then I don't need to select "all types".

To compile all .java file of a folder at a time, we can give command "javac *.java"

In JDK 6 premain() is came that can calculate the size of the object.

In Java, no class can be executed without main function from JDK 1.7. but before 1.7 we could execute a class without main function and that was a bug.

In Java, Every class is a child of Object class. The object is a topmost class in Java, it doesn't have any parent class.

Total 11 methods implicitly come in a class and they are very important methods.

One instance of Java is responsible to run one class at a time which is having a main() method.

Execution of class and use of the class is different. Execution of class will contain main() method and used class don't require to have a main() method.

Instance: A runtime Instance of the Java Virtual Machine (JVM) has a clear mission in his life, to run one Java application. When a Java application starts, a runtime instance is born. When the application completes the Instance dies.

If we start three Java application at the same time on the same computer then we will get three instances. Each Java application runs inside its own JVM (Java Virtual Machine).

Some paragraph missing. Recover it.

The bytecode of every class is always stored in a separate .class file and the name of that .class file will be the same as your class name.

Java does not support pointers. Pointers are complex and they wanted to make language simple.

#. Decompiler Command.

Convert .class file into java code again. But from JDK 9, many things are hidden. JDK 7 gives little more information in comparison to JDK 8 and 9.

Predefined keyword (Reserved Keyword) are also called identifier.

All wrapper classes are immutable that means those classes are defined as final. Like String System Integer Float Double etc.

#. Know Class Information.

" javap java.util.Arrays>abc.txt "
"abc.txt"

It will open text file that will have Arrays class information.

#. When we run the program using java command, it loads the class into JVM and looks for main function in the class and runs it. The main function syntax should be same as specified in the program, else it won’t run and throw exception as Exception in thread "main" java.lang.NoSuchMethodError: main.