THEORY

1. Class: Class is a factory which creates Objects.
2. NEW keyword:

* it tells class to create an object, once the object is created new keyword will get the address and it will be stored in reference variable.
* It will automatically create a default constructor it is mandatory, when ever we write new keyword.

1. Garbage Collector: it helps us to manage the memory in an efficient way, it regularly keeps on removing unused objects so that overflow of memory is avoided.
2. Static Variable:

* Static variables are created outside method inside class, with a prefix of “static”.
* If the variable is static then it can be accessed anywhere in the program, it is like a global variable.
* It can be accessed with 3 types.
  + 1st method: with class name appended with variable name (A.age), (this is a best way to access the static variables).
  + 2nd method: directly by using variable name (there will be some flaw when the local variable has same name as that of the static variable).
  + 3rd method: by creating object and accessing with reference variable, it is bad practice but compiler is smart enough to convert that object to class common memory and access it.
* All these static variables belong or get stored in class common memory.
* Local variable name and static variable name can be same but if we access with directly variable name inside method where we create local variable with same name as local variable then local variable will be printed not static variable so we prefer always accessing using class name appended with variable name.
* We don’t have to initialize variables mandatorily, if not initialized it will get a default value based on data type of variable declared.

1. Non-static Variable:

* Non-Static variables are created outside method inside class, without a prefix of “static”, With mandatory Object creation.
* We have to create Object mandatory so that all non-static variables can only be accessed with object creation and reference variable.
* We don’t have to initialize variables mandatorily, if not initialized it will get a default value based on data type of variable declared.

1. Heap: In JAVA all the objects which are created are stored in Heap memory.
2. Stack: Stack will help us to maintain programs execution flow.
3. HEAP AND STACK CONCEPT: it helps us to maintain memory in efficient way by creating a program with better execution flow and deleting all unused obj with the concept of garbage collector.



1. LOCAL VARIABLE:

* Local Variables are created inside method and can only be used within created method, if they are used outside method, it will give error.
* It should be INITIALIZED mandatorily or else it will give error.
* Local variable can have the same name as that of the static variable.
* Accessing local variable is directly by using its variable name.
* Note: instead of writing “,” to read numbers in program like salary amount we can use “\_” so that it will not give an error.

1. DATA TYPES:

* 9 Types:
  + Byte-> 1 byte -> 0.
  + Short-> 2 byte -> 0.
  + Int-> 4 byte -> 0.
  + Long-> 8 byte ->0.
  + Float-> 4 byte ->0.0.
  + Double-> 8 byte -> 0.0.
  + String(class)-> N/A -> NULL.
  + Char-> N/A -> empty space.
  + Boolean -> N/A -> false.
* 1 Special Type:
  + Var: It is a not a data type because no data type can have a var name as that of the data type but in var we can pass variable name as var also so it’s a type rather than data-type.
  + Var var = 10; (accepted)
  + It was introduced in version 10 and above.
  + It cannot be passed as a method argument
  + It cannot be static or non-static variable it can only be local variable.

1. Reference Variable:

* DATA-TYPE of reference variable is Class name.
* Reference variable can store object address and null values.
* If reference variables are used in method, then it is called local reference variable.
* If reference variables are used in used outside method with static keyword, then it is called global reference variable.

1. Methods:

* A method is **a block of code which only runs when it is** called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.
* If we call a method with parameters same way, we have to create a method with same argument and data-type should also match
* Special case -> p v test (int… i) {sop(i[index]);}
* A method can have same name as that of the class name.\

1. Return-Keyword:

* It is used in void methods.
* We don’t have to write it mandatorily.
* It will pass control back to the method calling statement.
* We cannot write anything after return keyword it should be the last sentence in the method.
* It can be written inside the main method.
* It can be written in static methods.

1. Return-Value – keyword:

* If used in void method it will give an error.
* It is mandatory to write in non-void methods or it will give an error.
* It will pass control and value back to the method calling statement.

1. Constructor:

* Constructor should have same name as that of class name, its mandatory.
* Every time we write a new keyword a default constructor is created and it is mandatory.
* As many objects we create as many times a constructor is called.
* If we put void in front of constructor then it is not a constructor it is a method, to access it we have to create obj reference variable and call it.
* If we create object with argument, it is mandatory to create a constructor with argument of same data type, it is mandatory if not created it will give an error.

1. JRE (java runtime environment):

* It consists of only java runtime environment.
* It is used by client/customers, where they only have to run the program.

1. JDK (java development kit):

* It consists of both java compile and runtime environment.
* It is used by developers, to compile and run the program or code.

1. CONCEPT OF PROGRAM RENDERING USING UNICODE VALUES:



* Unicode->binary->signal->output.

1. Packages:

* packages are nothing but folder, that is to store all data organized way.
* package names cannot be keywords or java or upper-case.
* we cannot write anything above package it should be the first word in program
* packages are folders created in java to store programs in organised manner.
* To access objects or anything from different package we have to import the package with class name (e.g., p1.A-> where p1 is package name and A is class name.)
* Multiple packages can be created like p1.p2.p3-> that is folder inside folder inside folder.
* When we have 2 packages, where one package has classes A, B, where as other package having class A, then resolve the naming conventions in creating objects in class B we write 2 objects having independently with 2nd method package import concept that is package name followed with class name (p1.A & p2.A) where ever there is A in the program in class B.

1. Constructor overloading:
   * here we create more than one constructor in same class.
   * create constructors with different number of arguments or different type of argument.
   * if those 2 different arguments belong to same data type like int and byte etc.
   * Note: in the below program, it will call int because all numbers by default is treated as int.
   * \*\*\* to pass byte value we write as new //Ex3((byte)10);
   * // \*\*\* to pass long value we write as new //Ex3(10L);
   * // in java if we write any number, it will always be treated as int, if we
   * // explicitly want to pass byte value then we do down casting.
   * Note: class name constructor name method name and variable name can be same as shown in the below example. Although it is not encouraged to write programs this way.
2. Oops concept- Inheritance:

* Inheritance is done for only non-static members.
* Here the class we inherit is called as parent class or super class, and where it gets inherited is called as child class.
* There is no multiple inheritance in class level in java language, i.e., to reduce complications of reading program.
* Extends: is a keyword which helps us to inherit
* Note: we cannot inheritance without importing package where other program is present in other class and other package.

1. Oops concept- Polymorphism:

* It can only be applied on methods and not on variables.
* here we develop a feature such that it can take more than one form.
* There are 2 topics in polymorphism i.e.,1) overloading 2) overriding.

1. Overloading:

* inheritance is mandatory to do this overloading.
* here we inherit a method from parent class and then we modify the logic of inherited method in child class by once again creating a method with a same signature in the child class
* in overloading methods won’t get copied but it gets replaced by the new method which we create in child class with same method name, as method is taking 2 forms, we tell here it is undergoing polymorphism.

IMPORTANT POINTS:

1. Note: instead of writing “,” to read numbers in program like salary amount we can use “\_” so that it will not give an error.
2. Note: we can use string with any data type it wont give error but any other if we use it will give error, (int String = 10; [accepted], int int =10; [error]).
3. Note: Special case -> p v test (int… i){sop(i[index]);}
4. Note: A method can have same name as that of the class name.
5. Note: A local variable can have same name as that of the static variable.
6. Note: we can’t write anything after return keyword, it should be the last argument inside the method.
7. Note: If we put void in front of constructor then it is not a constructor it is a method, to access it we have to create obj reference variable and call it.
8. Note: Every time we write a new keyword a default constructor is created and it is mandatory
9. Note: in the below program, it will call int because all numbers by default is treated as int.

* \*\*\* to pass byte value we write as new //Ex3((byte)10);
* // \*\*\* to pass long value we write as new //Ex3(10L);
* // in java if we write any number, it will always be treated as int, if we
* // explicitly want to pass byte value then we do down casting.

1. Note: class name constructor name method name and variable name can be same as shown in the below example. Although it is not encouraged to write programs this way.
2. Note: only one class should have public when they are created in same file, and which is public should be name of class.
3. Note: we cannot inheritance without importing package where other program is present in other class and other package.