**JAVA NOTES**

Java was conceived by james goslin, Patrick naughton, chris warth,ed frank and mike Sheridan at sun microsystems,inc.

In 1991. it took 18 months to develop the first working version.

Java was initially named as Oak.

Java is a hybrid language mean (both interpreted and compiled language) the nature of its hybrid is that’s why because the application which is created by java is platform independent.

Because the java execution is based on two parts.

First the code written in java converted into bytecode through the java compiler by using command (javac filename.java).

This bytecode made the java platform independent.

This bytecode is not in executable form because it contains the instruction which is execute by using java interpreter using the command (java filename.class).

Java is fully object oriented programming language based on classes and revolving around the object.

**JAVA VIRTUAL MACHINE**

Java virtual machine is a process virtual machine which execute the bytecode. Every java source file is translated into bytecode. Which is after execute by JVM. As we know that java is designed to create the application which run on a different platform. So a java virtual machine makes this possible.

**Difference between JDK and JRE**

The JRE stand for java runtime environment is basically a JVM where our code is going to be executed. And The JDK stand for java development kit which is fully featured software kit which include the JRE, compiler (java Doc, java debugger) which is used to developed the software’s.

**Object oriented programming**

Object oriented programming is an approach which is based on real world which uses the concept of classes and objects in programming.

it provides the code reusability.

it also helps to hiding data and provide us the security.

**class:**

Class is the template for an object which contains the member (instance variables and instance methods) of the class.

We can declare the class using “class” keyword.

Class does not acquire any space in memory.

Class is user define data type which is used to create the objects.

Objects:

Object is also called the instance of a class. which is use to access the member of class.

An object is a real world entity which have its functionality and property.

Object also consume the memory in heap. because object is always created at run time.

**Declaring object:**

When you create a class you will creating a new data type.

Through this we will creating the object of that type obtaining object of a class is a two type process.

First you will declare the variable type class name which does not define any object and simply refer to the object.

In second step you must acquire the physical copy of that variable and assign it into that variable we can do this by using the new keyword (the new keyword allocate the memory at the run time ) and then new return the reference (memory address of that object) and assign it into that variable.

**Class object**

Class is the collection of object object is the instance of class

Class don’t acquire any space object acquire space in memory

It doesn’t exist in physically it exists in physically.

Class was created just one’s object was created when

required

**Instance variables and instance methods:**

The variables and method declared with in a class is called instance variable and methods because each instance of the class (object of the class) has its own copy. If we change the variable value of any particular object. then it will not affect the value of instance variable.

***Pillars of object oriented programming***

There are four main pillars of OOP’s

1. **Encapsulation**: Encapsulation mean hiding important data and show unnecessary data.

**Def:**

Encapsulation is a mechanism to binding data and methods together in a single unit.

**or**

also called encapsulation provide us an object with the ability to hide their characteristics and behaviors

the main advantage of encapsulation is that it protects the random access of class data from outside.

**Note:**

In encapsulation we use Access modifiers to set the access right (where the data of class will be available to access).

There are four access modifier in OOP’s.

1. Private:

The member declared in class with private access modifier so it will not be access from outside of that class.

We use “private” keyword to declare the member as a private.

2.public:

The member declared with public keyword so it can be access from anywhere (outside the class, outside the package, subclasses).

3.protected:

The member declared with protected keyword so it can be access from subclass if it is in same package or different package and (can’t be access by class which is not in same package).

**Inheritance**

Inheritance is the way through which we can inherit the one class to another class.

Or

Inheritance provides an object with the ability to access the member of the another class (base class).

We can inherit one class to another class by using the “**extends**” keyword.

* The main purpose of inheritance is to increase the reusability of code.

There are three type of inheritance supported in java.

* Single level inheritance

In which we have only one base class and their only one derived class.

* Multiple inheritance

In which derived class inherit the property of more than one base class

* Hierarchal inheritance

In which we have more than one derived class inherit property of single base class.

* Multi-level inheritance:

In which derived class is created from another derived class.

* Hybrid inheritance:

Hybrid inheritance is the combination of more than one type of inheritance.

**Polymorphism**

Polymorphism mean more than one form or also called more than one behavior.

Polymorphism is the ability of an object to take many forms.

**Or**

Polymorphism refers to the ability of a class to provide the different implementations of a method depending on the type of object that is passed to the method.

**Or**

Polymorphism refers to the ability of message to be displayed in more than one form

It makes the code flexible and dynamic.

**There are two type of polymorphism.**

**Compile time polymorphism:**

Compile time polymorphism mean we have defined the behavior at the compile time.

**Ex:**

Suppose we overloaded the two add method in which one take three integer parameter

**add (int, int, int)** and second take two integer parameter **add (int, int).** when we compile it we know that which method will run.

**Method overloading:**

The method overloading is the phenomenon in which we have more than one method with the same name but having different in type of parametersor no of parameters.

**Constructor overloading:**

In which we have more than one constructor

in the class so which having different with each other in type of parameters, no of parameters, change in arrangement of parameters

**Run time polymorphism:**

Run time polymorphism means define the behavior at the run time.

**Ex:**

suppose we have two classes A and B and class B extending class A and overriding their method **add (int a, int b)** but we don’t know at compile time which method will run because overriding is associated with objects and as we know that objects were created at run time and then we decided which method is running.

**Method overriding:**

In method overriding we rewrite the method of base class into derived class.

**Abstraction**

In OOPs abstraction allow us to hide unnecessary data from the user and show only necessary data.

**Abstract class:**

Abstract class is a class which is used as a blue print for another class.

An abstract class contains the abstract methods as well as non-abstract methods.

An abstract class is declared by using “abstract” keyword.

**Note:**

An abstract method is the type of method which have declaration in abstract class and implementation in derived class.

**Contructure:**

Constructor is a special type of function whose name were same as class name. constructor has no any return type.

The main purpose of constructor is to initialize the object or also called the initialize the object in valid state.

Constructor is invoking (called) when an object is created.

Every class has a constructor.

**Note:**

In the case if programmer doesn’t have any constructor then java compiler will create a default constructor.

**Types of constructor**

There are three type of constructor.

**Default constructor:**

Default constructor is a type of constructor which take no any parameter. if we want to set the initial value in instance variables then we must set the value Inside the that class.

If we don’t create any constructor, then java compiler will also create the constructor which is known as default constructor.

**Parametrize constructor:**

Parametrize constructor is a type of constructor which take parameter to set the initial value of instance variables.

If we want to use the parametrize constructor, then we explicitly declare and implement it.

**Copy constructor:**

Copy constructor is a constructor which take the object in parameters. And it is use to set the value of one object to another object.

**Private constructor:**

it is a type of constructor which is declared as a private keyword. If we declared it then we can’t create the object of that class.

**Static constructor:**

A constructor which is declared with static keyword. It is called when the first instance of class is created or any static member is referenced.

**This keyword**

The **This** keyword is used to refer to the current object in a method or a constructor.

or also called refer to the object on which it is invoking.

It is also used to differentiate between the instance variable and parameters (if both parameter and instance variable has the same name).

**Note:**

**This keyword** alwaysstore the address of memory where the current object is stored)

**Garbage collection**

Garbage collection is the process in java which occur automatically and it is used to reclaimed the unused memory by destroy the unused object.

There are three state in which we called the object are unused.

**Non reference object:**

the object that don’t contain any object (contains the “null”).

**Anonymous object:**

Those object which is created without using its name.

**Same referenced objects:**

Those object which contain the same reference.

Protected Finalize method ():

It is a method which is declared in object class that is base class of all java classes.

It is used to release the resources (such as file handling) of unused object.

Before the garbage collector was going to destroy unused object.

Interface

Interface is defining as the template for a class or also called creating the design for another class.

An interface in the java is used to describe the behavior that classes must implemented.

The methods inside the interface are by default “public” and “abstract”.

Which contain the abstract method as well as default method.

Interface can be used by the class by using the “**implements**” keyword.

Through interface we achieve the multiple inheritance in java.

We can’t make the object of interface but we can make the reference of interface.

**Note:**

The variable declared in interface are by default “final” and “static” so that’s why we can’t only declare the variable we must be initialize it.

**Nested classes**

The class with in the class is called the nested classes.

The scope of nested class is bounded by the scope of its enclosing class (means that nested class has no any scope outside the outer class).

Suppose we have the class A and in which we define the class B in it. So class B can access all the member including private member of class A. but in another way the class A can’t access the member of class B.

**Ex:**

Class A {

int age=10;

public static void main (String args []) {

B obj=new B ();

Obj.show\_name\_from\_inner ();

}

Class B {

Void show\_name\_from\_inner () {

Sout (“age is ” + age);}

}

}

**Heap and Stack**

Heap is called the free pool of space which has not fixed size.

Heap is the part of memory which we use when we don’t know that how much memory our program will take so that’s why we use the Heap. In heap we store the data at the run time.

In heap we manually allocate the memory in heap using **“new”** keyword and deallocate the memory using garbage collector using command (“**System.gc()**”).

**Stack:**

It is part of memory which contain the fixed size.

In our program each function has its own stack in memory which is allocated at run time contains the local variable of that function.

In stack the memory allocation and deallocation is done automatically.

**Anonymous**

Anonymous means the thing which have no any name.

**Anonymous object:**

Anonymous objects are those object which has no any reference variable or also called no any name.

**Ex: new age ();** => creating object of class age which have no any reference.

**Anonymous classes:**

Anonymous classes are those classes which have no any name, which is instantiated and declared at a same time.

They are like a local class.

We can use them if we need to use local class only once.

**Syntax:**

Person p1=new Person ()

{

Public void show () {

Sout (“hello world”)

}

}

**Explanation:**

If we want to show the multiple behavior of one method then we often use the method overriding technique by creating a new class but we can also do it by creating anonymous class. Which help us to concise the code (without creating another class and extends it).

**Jagged\_Array**

It is the type of an array which contain the property to give the size of 2D array element separately.

**Ex:**

Int [][] num=new int[3][];

num[0]=new int[2];

num[0]=new int[2];

num[0]=new int[2];

**Final keyword:**

**Notes:** We can use final keyword with variables, methods, and class.

* **With variables:**

the variable declared with “final”keyword we can’t change it value.

* **With methods:**

the method declared final keyword we can’t overload it or override it.

* **With class:**

when we declared the class with final keyword so we bound it to that no everyone class can inherit it.

**Object Class**

The object class is the parent class of every java class which contain come built methods.

**Ex:** toString(), equals(), Hashcode(), getClass(),.

**Note:**

where we want to print the object then obj is by default calling the toString method which will return the information of object in hexadecimal value of their Hashcode.

Whenever we compare the two objects by using equals() method it will compare the Hashcode of these objects as well as their value of instance variables with each other.

**Wrapper class:**

A wrapper class in Java is a class whose object contains the primitive data type. Because when we create the object of that class it contains the data of primitive type.

Or also called the class which wraps the primitive data into an object.

**Note:**

Each primitive data type we have corresponding class.

Ex:

* int => Integer
* float => Float
* double =>Double
* short => Short
* boolean => Boolean

etc.

**Explanation:**

The java supports 7 to 8 primitive data type which makes the java 99% OOP language but because of 1% java is not fully OOP language. In the result java have some frame work which only use the objects. So by solving these situation javas have the wrapper class which convert (wrap) the primitive data to objects.

**Boxing:**

When we wrap the primitive data type into object type explicitly is called Boxing.

**AutoBoxing:**

When the primitive data value is wrap or convert into object type automatically is called Boxing.

**UnBoxing:**

When we getting the object type value into primitive type explicitly is called Boxing.

**AutounBoxing:**

When we get the object type value into primitive type automatically is called Boxing.

**INNER CLASS**

The class with in a class is called inner class.

Which is used to create the helper class.

We can’t create the object of inner class directly in the outside of outer class.

We can create it by use the help of outerclass.

**Note:**

We can only use the static keyword with that class which is inner.

**EX:**

**class A {**

**Class B {**

**}**

**}**

**Class C {**

**main {**

**for creating object of class B firstly we create the object of class A**

**A obj2=new A ();**

**A.B obj=new A.B();**

**}**

**}**

**Enums:**

Enums in Java is a reserved keyword that is used to create the new datatype. The emus contain the constant which is by default “public”, “static”, and “final” and the constants are separated by commas.

The constant of the enum is called the object of an enum.

The enum can’t be declared inside the function

But we can declare enum in class, like inner class.

**Ex:**

enum Month {

January,

February,

March,

}

We can create the constructor of Enum but it should be private because we can’t create the object of the class from outside the enum and we can also create the method and instance variable in enum.

**Annotation:**

The Annotation in Java is a tag representation of metadata of our class, function, and variables.

The annotation is used to give more information to compile or JVM.

The annotation is always start with “@” sign.

**Ex: @override, @Depreciated etc.**

**Exceptions:**

An exception is unexpected, unwanted, abnormal event which occurs at the run time that causes the termination of execution.

Exceptions are the run time errors which stop the normal flow of execution so that’s why we have to handle it by using exception handling.

**Types of Exception:**

There are two type of Exception.

Checked Exception:

Those Exception which compiler notify or bound you that this block of code could generate the error so that’s why you must handle it.

**Ex:** InFile Handling.

Unchecked Exception:

Those Exception which is occurred through the wrong logic which compiler don’t notify.

**Exception handling:**

Exception handling is the process in which we handle the exception.

There are two ways to handle the exception.

1. Try catch block
2. Throws exception keyword.

**Try and catch Block:**

In try block we try to execute the code, if any error will occur then it executes the catch block which take the argument “exception type” here are some of the “exception-type” which is occurred at some time.

* Exception: which catch-all type of error
* AirthmeticException: which catch only arithmetic exception.
* ArrayoutofBoundIndexException: which catch the index error of array.

etc.

**Finally Block:**

In finally block we write the that which we want to be execute at any situation i: e (successful execution, after handling the exception).

**Throw keyword:**

Throw keyword is used to throw the Exception object with some message to the catch block.

**Throws Keyword:**

Throws keyword is used to bound you that whenever the specific method => (in which we have the possibility of exception) is called then at that time you will be handled it.

**Functional Interface**

Functional interface is that special type of interface which have only one abstract method.

We distinct the functional interface by using the annotation “@FunctionalInterface”.

***Lambda Expression***

Lambda expression is the methodology to implement the abstract method but lambda expression only implements the method of functional interface.

Lambda expression is used to summarize the code or reduce the code of anonymous inner class.

**Ex:**

@FunctionalInterface

Interface A {

Void show();

}

Main() {

By using anonymous inner-class

A obj=new A () {

public void show () { sout (“hello”) };

};

By using lambda expression

A obj= () -> Sout (“hello”);

The both lambda and anonymous method are equal.

}

Primitive type and their range.

Byte contain the number from (-128,127).