**:::Bits & Bytes of JAVA:::**

**DAY 0(SEP 14): Induction Program**

**DAY 1(SEP 23): JAVA Fundamentals**

Java: High level, Object oriented programming language, Platform independent, Strongly typed.

Java is more secure

Everything in java are objects.

Used to create huge enterprise applications.

Post compilation everything is converted into bytecode and can run in any platform.

Objects can have states and actions.

Can be used for developing standalone apps, wed applications, android apps and enterprise applications

Class can hold variables and functions.

Field – variable - attribute

Action – function - method

Byte code can run on any operating system post compilation.

Object: Instance of a class or blueprint of class

In order to use any action or field in a file we need to access through object.

Class: It is a blueprint that object needs to follow or It is a combination of methods and attributes.

Or collection of objects

Example;

Class employee

{

ID,first name, last name, Phone(These are the attributes of a class)

{

And a class can also contain methods

Data types: Int, char, float, double, string, short, long – Primitive datatypes

User defined datatypes are non primitive.

How can a object be created?

It is created using a constructor(Constructors creates instance of a class that is object)

It is for creating and initializing the object.

\*\*\*In Java, always class name and file name should be same.\*\*\*

By default constructor takes zero argument as parameter in not explicitly defined.

Function/method: A Set of code to do a particular task.

All our applications need to secure. So we use access modifiers.

To access variables or fields and functions or methods we use objects.

Objects are created using constructor.

Constructor: It is used to create an instance of a class.

Ex: new caluculate();

A zero argument constructor is supplied by default.

Private, public can be defined then default is not supplied.

Access modifiers: Visibility of a class. Till where this code is available.

Private: can used with variables. It is only available within the class.

**Return type:** Type of value that method will return.

Typecasting:

Operators: Arithmetic, logical, bitwise, Relational.

**Conditioning statements:**

If

If-else

Switch

**Loops:**

For

While

Do-while

**OOPS:**  
To make data more secure we will make the fields private, another thing we can use our method inside the class to add more security.

**Data encapsulation:** Wrapping up of data or hiding from this world

**Inheritance:** Using extends keyword, a child can access all parent class properties.

Reduce redundant code, Reusability is there.

**Polymorphism:** Exisistance of same thing in multiple formats.

Method overloading

Method Overriding

IDE: Integrated development environment (Editor, Compiler, Output console)

**Data abstraction:**

Signature of my method, if U are extending me so define me.

Abstract methods: These are methods which are declared but not defined.

End method with **;**

Declaring is only writing signature.

Method(){

} - Defining

For super class the definition is not required so we made it abstract. So, it is child classes responsibility to define the above declared method.

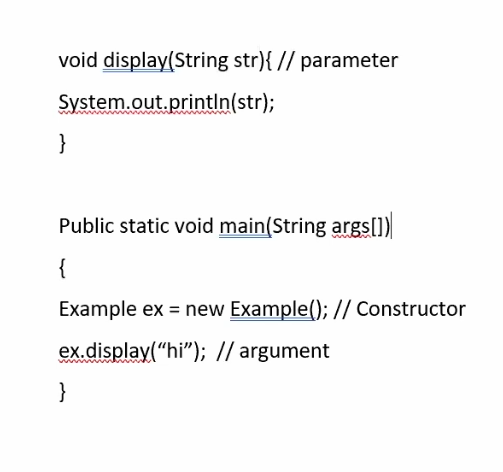
Whoever extends me they have to define me.

If you are not writing any logic then u extend me, make yourself abstract and send me to next below level. So, they will define me atlast.

Generic type – Animal and type of food

Relevent – Dog or cat and food type is important

**Parameters and arguments**



The values which we pass to function are called arguments.

The values which we can see at the receiver end like Ex: Int x, int y.

One of the classes in a file must be public.

A file can have multiple classes.

Non-primitive datatypes:

These are which user defines like

Collections: List, set, map

**There are 3 types of variables:**

**Variable types:**

1. Local variables
2. Instance variables
3. Static variables

Local: Defined inside method

Instance: Defined in a class

Static: It is global variable, not related to object

I or count: These are interactive variable. Used to count objects

Methods will be having () , followed by { } to define its functionality.

Ex:

Method(){-------------}

Developers don’t use main much, they use more frameworks instead of main.

For using constructor the syntax is

Ex:

Example(int x, int y){

}

Here we define constructors with class name

JVM: Java virtual machine takes zero argument by default and provides it to file.

JVC: Java compiler

Java.exe

**Data types:**

**Char – 2 bytes**

**Byte – 1 byte**

**Short – 2 bytes**

**Int – 4 bytes**

**Float – 4 bytes**

**Long – 8 bytes**

**Double – 8 bytes**

**Boolean – 1 bit**

**What is an abstract method?**

It is a method that is only declared but not defined. It is the responsibility of child/sub class to define it. Or child/sub shall pass this method to next sub levels where it needs to defined by those methods

Abstract is a non-access modifier.

Parameter & Argument:

Parameters are at the receiver end while inputting.

Arguments are the values which are passed to the functions/methods.

Operators:

Arithmetic, logical, Increment/decrement, Assignment, Comparision, bitwise

OOPS:

Data abstraction

Data encapsulation (Security / Data hiding)

Polymorphism(Method overloading)

Inheritance(Re-usability)(Method overriding)

**Conditionals:**

If

If-else

Switch

We use switch when we have multiple if-else conditions to execute.

Switch(Condition){

Case 1:

Break;

Case 2:

Break;

Case 3:

Break;

Default:

}

Default statement is not mandatory in switch condition.

Default statement can be situated anywhere in the switch. It is allowed.

The value of a case should be constant. Only primitive types we can use switch like usual data types, enums and strings.

A switch can take an expression

Cases will take int value, characters and string type as well

Cases should be constant.

String is a class type in java

Non-primitive types are user defined data types

Loopings: for, while, do-while

For(), while() and do while():

Iterators

Example:

1)

For(i=0;i<5;i++){

Sop(a[i]);

}

2)

I=0

While(i<5){

I++;

}

3)

Do{

I++;

}while(i<5);

//Do-while will execute one more time than while

For(int=0;i<5;i++){

Sop(i)

}

Int i=0;

While(i<5){

Sop(i)

I++;

}

Int i=0;

Do{

Sop(i);

I++;

Sop(i)

}while(i<5);

In a java program objects communicate by invoking methods.

Each object has a state/variable/attribute/field and behaviour/method/function/action.

Example:

Here the object is dog,

State: name, breed, size, weight

Behaviour: bark, jump, run, sit

**DAY 2(SEP 24): JAVA Fundamentals**

Type Casting: Conversion of one datatype into another.

Comments:  
// ---------------------------------------// -> Single line comments

/\*---------------------------------------\*/ -> Multiline comments

While using a class first letter of each word should be capital. Ex: MyClassNum

While using a method first letter should be smallcase and from second word first letter should be capital. Ex: collegeHolidayStudents()

Method always has parenthesis

In java both class names and file names are case sensitive.

Public static void main(String[] args)

Public: Available both inside and outside package

Static: Method that can run without creating an object or which can be globally used

Void: return nothing

Main: Place where programs start execution

Parameters are declared inside parenthesis

Do not use ; after classes and methods

We use comments to increase the code readability and to avoid confusion while solving complex programs.

All the names that we use in java are known as identifiers.

Char – ‘a’ -> It is defied in single quotes

String – “str” -> Is is defined in double quotes

Scanner object is used for inputting data from user in java.

In a file one class must be public.

Constructors: By default zero argument constructor will be passed by jvm if not defined explicitly.

Casting:

Array: 1-D

Break and continue:

Break comes out of the loop

Continue skips that particular iteration

Blocks

{

}

Pre – Post increment operators

Class path setting

Env – path

Command promt – java –version

New for loop

For(int i:numbers){

Syso

}

Scanner takes input form the user

Package: Java.util.scanner – Import for the taking input from user

System.in is a built in class

Nextint() – To take input for a number.

X=x+1

X+=1

y-=2

y=y-2

Array: Can hold elements of same datatype

Give size while declaring. They are not dynamic.

Array.length – Gives the count of elements present

We call super class constructor using super method

Instance variables by default takes 0.

But local variables needs to defined if not it errors out.

This adds uniqueness

This:::

Zero arg constructor ::: Is not provided when one explicitly defines another constrcutor

**DAY 3(SEP 27):**

\*this keyword

\*super keyword

\*static keyword

\*method overloading & method overriding

static variable can be accessed globally(Globally). It is a class variable. It is not an instance variable means its not a part of an object. It can used across on objects.

Any static method is not referenced using a obj

To call a non static method we need to use an object to call

Variables, methods, blocks, classes can be static

Static variable used one memory

This: It is reference variable which points to current objects.

We can use this on a instance variables, can be passed as an argument, can be used with constructor also

There is only one memory allocation done to the static variable, only one reference for the static variable

Defined constructor has no return type.

Method overloading: They should differ in number of parameters or type of parameters

This will call same class constructor

Super is for inheritance, for the parent class

By default always prefer the childs because it has all latest things and child will be having access to all parents things as well.

The parent class has no visibility for childs methods

Inheritance is defined using extends keyword.

Method overloading(Return type is not bothered, what is required the number of parameters and type of parameters. Method overloading and method overriding are polymorphism.

Your method overriding needs inheritance concept.

Super class object doesn’t fit into subclass object.

Super: It is a reference variable which points to super or parent class

Can used with methods, constructors and methods.

It is not a instance variable. Instance means not a part of an object. It is a class variable which can be accessed globally.

We can have variables, classes, blocks{….},methods(Main() method is static) 🡪Can be static

To count number of objects we can use static to save memory.

this operator: It is a reference variable which points to the current object.

It can used on instance variables, passed as an argument, with constructor also

Super: It is a reference variable which can point the super class/parent class objects.

It can be used with methods, constructors and variables.

Package is a place where all our programs are available. We can import, 2 packages can have same name

Count hold common value across the objects

we can use a static variable in a non-static method.

Any static method is not referenced used an object

To call a non static method we have call it through an object. If everything is static in a program then no issues.

Methods are defined using parenthesis

If your method is of int type then you can have integer value or else the compiler complains

There is only one memory allocation done for the static variable, there is only one reference possible for the static variable

Method overloading bothers on number of parameters and type of parameters.

Method overriding needs inheritance

Access modifiers:

1)Public

2)Private

3)Protected

4)Default

**DAY 4(SEP 28):**

Abstraction: Hiding the implementation details from the outside world. We can make our abstract using abstract keyword.

Abstarct class: atleast one method should be abstract.

Is it possible to create an obj using abstract class?  
You can create it but u cant instantiate it, we need to make it work through a subclass only.

Abstract is not applicable for the variable. It is only applicable for class and methods

Abstracts are only declared. Once after extending it is the job of child class to define it.

Interfaces: These are 100% abstract, when you working with interfaces it deals with implements. When you are working inheritance it deals with extends

Class A extends B,C – Not possible

ClassA implements B,C – Possible

Interface Policy

{

}

In interface all the methods are not defined.

Only declaration can happen.

Interface can hold a variable which is public static final int a=1;

Implicitly abstract is applied for methods and variables even we don’t provide

What is it meant overriding cannot make access modifier restrict your right, so if I have a project that if I have a public I cannot make in a subclass protected I showed you yesterday, she likes me do you understand, we did that in your system yesterday.

So public was they're protected it it gave problem right protected to public is that variable you cannot make that small even more restrictive when it comes to method over lighting right.

And, for example, the object as a man like superclass object is being assigned to a subclass object like that or not, so you have to write all that Okay, you have to make a note of all.That.

So when you when you cannot judge OK, now the superclass object can sit in his class variable instance now, it gives the.User not so, these are the Java things right, you should know and the tone, I want you to reference document clear.Subclass cant be more restrictive(public to protected), but protected to public is possible. Superclass cant sit in subclass object.

It is possible to have non-abstract methods along with abstract methods in a class.

Abstraction: I don’t implement but I will declare and keep it is the responsibility of others who extends me, should define me

Abstraction

Access modifiers: To define the visibility of code

Abstraction concept:

Abstract class VS Interface

Extends VS implements

There are 4 types of access modifiers

1)public

It can accessed all over the package and outside the package as well

No big restrictions

2)Private: Only defined to use in a particular class.

This is not available for the other class in the same file

Heavily restricted

3)Protected

This is available for sub classes but not outside file.

This little restrictive

4)Default

This is available all over the current package

Non restricting

Interfaces: There are 100% abstract class. It will have public static final and followed by variables, All the methods are abstract, only declarations made but not definations.

We use implements keyword.

Here the variable types is public static final

**DAY 5(SEP 19):**

Exception handling and File handling

Exception: Something going not as per plan(Run time errors)

1. Null pointer exception is an example, like this we have many exceptions.

Exception is a class in java.

Throwable interface

Error class(machine level problems)

Exceptions are handled by programmers

**Exceptions types:**

Arrayindexoutofbound

Filenotfound

Arthimeticexception

Numberformatexception

classcastexception

Nullpointerexceptions

IOexception

We can have **finally** after try block directly without catch block. Compulsory things we need put in finally block.

There are different type exceptions in java:

Output of a program is error or exception:

We can handle exceptions but we can’t handle errors

Exceptions are meant to be handled.

So, we use try and catch block to handle exceptions.

To handle exceptions java has try, catch, finally, throws, throw

Nested try - possible

Multiple catch – possible

Catch inside a catch

**Syntax:**

Public static void main(str args[])

Try{

}catch(){

}

**Throws** means this method is tend to throw an exception, You better catch it.(Throws are used in the function definition – Signature of the method)

**Throw** is used inside try-catch block. It is to purposefully create an exception when required.

We can extend and customise our own exception class

Byte is a type of data.

**File handling:**

File is a class in java.

TO read and write data into file.

Create, delete, update, restrict access to file, Copy of file all using java

Stream is flow of data. Here we have byte and character stream.

We have reader and writer classes.

In File handling we can perform reading and writing data in multiple ways:

Syntax:

1)

To try catch inside a catch

2)

One try multiple catch block

3)

Differences between the several file classes like

Buffer is a temporary strorage.

Files can be stored in various locations:

BufferedReader

BufferedWriter

InputStreamReader

OutputStreamWriter

Filewriter

Filereader

FileOutputStream

FileInputStream

**DAY 6(SEP 19):**

Collections: To overcome the disadvantage of array we use collections. These are dynamic in nature

Collections is a class in java.

**Types of collections:**

**List : arraylist, linkedlist**

**Map : Hashmap**

**Set: Treeset, Hashset**

**Queue : PriorityQueue**

**These classes will implement the interface**

All the above are interfaces

Collections is a class but collection is an interface.

Interface are 100% abstract

List will contain ordered elements. Can have duplicate elements

Map is a key value pair

Set(Unique elements, No duplicated allowed)

We are sure what type of data we will get as output. If we dint use proper data type to receive that data. Then exceptions may be occur

So to avoid this, we use Generics

These Generics came into picture.

Generic : In common

Syntax: <T> , <E>

ArrayList<String>strLst = new ArrayList<String>();

This statement will allow only string type.

ArrayList<Employee>strLst = new ArrayList<Employee>();

It takes datatype and object types.

**Wrapper class:**

To use data types in Generics we use wrapper classes.

Collections are meant to work with objects.

How to make our arraylist hold integer values???

Using wrapper classes

We have classes like Integer, Float, Double, Character, Boolean

Conversions of objects in datatypes and vide versa

**Autoboxing**

**Integer I = new Integer(5);**

An is int type(obj) is being boxed integer class type.

**Unboxing**

**Int x=i.value();**

Class we want convert a class type into data type we do this

Iterator built in class for printing the values of any collection type

For(int i:numbers){

}

**Day 7(Sept 20)**

**Generics, Threads and Inner classes.**

**Thread: It is a class which allows to perform multitasking.**

Multitasking:

Sequence of execution of threads is not defined. But we can set priority

Printer follows a queue system.

All requests are lined up. This leads to deadlock.

But it is possible in threads

How to create thread?

1)Using the thread class(Extending thread class)

2)Runnable interface(Implementing runnable interface)

Write logic in run() method

Class example extends thread

Or

Class example implements runnable

{

Example e = new Example();

e.strat();

}

Public void run(){

}

When start method is called then run method is executed.

Generics are brought for type safety.

**INNER CLASSES:**

Class inside a class is knows as inner class

Reduces code size and increases readability.

**Types of inner classes:**

We have static inner class

Inner class created within a method

Anonymous inner class

Regular inner class

**How to access inner class?**

Outer.inner in = new Outer.inner();