Mid-1

1. **Classes :-** It is a template used to create object and to define object data types and methods.

Syntax :-

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

object :- They are real time entity. There are created by using class. by new keyword

syntax :-

```
classname <objectnames> = new classname();
```

Constructor :- It is a block of codes similar to the method. It is called instance of the class is created.

Syntax :-

```
<classname>() {}  // default constructor
<classnames (arguments) {}  // parameterized constructor
```

Method :-

A method is a block of code (or) collection of statements (or) set of code used to perform a certain task (or) operation. It is used to achive reusability of code.

syntax :-

```
return type methodname()
{
  // method body
}
```

```
// program to see class, object, constructor and method

impost. java. io.* ;
class A {          // class
  int num;
  String name;
  A()   // constructor
  {
system.out.println("constructor is default");
  }
}
class B {
  public static void main( String [] args) //main method
  {
    A  AA = new A() ;  // object
    system. out. println( AA. name) ;
    system. out. println (AA. num) ;
  }
}
```

O/p:— constructor is default—

NULL

0

5) String :— string is a sequence of characters. But in java, string is an object that represents a sequence of characters

literal method :— String <object> = "string";

syntax :—

here string is quoted in double quotes

new method :—

syntax :— String <object> = new String("string");

Eg :— class StringEx()

```
{
public static void main(String[] args)
{
String s = "saudhani shaik"  //String declaration
system.out.println(s.toUppercase())
system.out.println(s)
}
}
```

String Handling Functions :— string Handling Functions are predefined. It is immutable object

Some Handling functions are

1. Concat()
2. toupperCase()
3. to lowerCase()
4. charAt()
5. equals()
6. length() etc.,

Ex:- class StringEx ()

```
{
 public static void main ( String [] args)
 {
   String s = "shaik sandhani" ;
   print (s) ;
   print ( s.touppercase) ;
   print ( s.tolowercase) ;
   print ( s.length() ) ;
   print ( s.index (h)) ;
 }
```

o/p :- SHAIK SANDHANI
shaik sandhani
14
1

2 a) Method overloading :- In a class containing same method name but parameters are different

Syntax :- class <classname>

```
{
  int method-name (arg) ;
  int method-name (arg1, arg2) ; // method name same argument different
}
```

Constructr overloading :-

using more than one constructor in a Instance of class.

Syntax :- class classname {

```
classname()      // classname same
{
...
}
classname(arg)
{
...
}
}
```

Ex :-

```
class MOCO()
{
~~public static void~~
int a;
int add(int a);
int add(int a, float b);
MOCO()
{
print("default constructor")
}
MOCO(int a)
{
print("paramcterized Constructor");
}
}
class MainMOCO()
{
pushic stabic void main(String[] args)
{
MOCO A = new MOCO();
x= M A.add(10)
y= A.add(20, 20.5)
System out.printfu(x);
System.out.printfu(y);
}
```

b) import java.util.Scanner;

```
class Minimum
{ public static void main (String[] args) {
  int a, b;
  Scanner sc = new Scanner (System.in);
  System.out.printfln ("Enter two values");
  a= sc.nextInt();
  b= sc.nextInt();
  if (a>b)
   {
    system.out.println("b is small");
    System.out.println(b)
   }
  else if(a<b)
    {
     system.out.println("a is small");
     system.out.println(a)
    }
  }
}
```

O/P:— Enter two values

60 61

a is small

60.

3) Method :- Method is a collection of statements (or) a block (or) set of code used to perform certain task (or) operation.

Syntax :- return type methodname() {

// code

}

Constructor :- Constructor is instance class. it is automatically invoked when an object is created, the name of constructor is same as class name.

Syntax :- class <classname>

{

classname () // Default

{

//code

}

}

Ex:-

Ex:- Class Example

```
{
int a;    // taken two integer, one double
int b;
double c;

Example (int a, int b, double c)
{
 a=a;          // argument (parameterized)
 b=b;
 c=c;
}
double avarage ()
{
 return ((a+b)/2);
}
}
class EXMain ()
{
public static void main (String [] args)
{
 Example A = new Example (10, 20, 60.61);
 double Avg;
 Avg = A. avarage ();
System.out.println (Avg);
}
}
```

O/p:- 15

write the object name as the argument name while calling the function the same way we do it for other variables

Syntax :- function_name (object_name);

2(a) object is an Argument it use to establish communication between two or more objects of same class as well as different class.

It is use easy to user to process data of two same or different objects within function.

Ex :-

```
class Add
{
  int a;
  int b;
  Add( int a, int b)
   {
    a=x;
    b=y;
   }
  void sum( Add A1) // object passed as argument
  {
    int sum1 = A1.a + A1.b // sum1 = new object
    system.out.println( sum1);
  }
}
  class Final Add()
  {
    public static void main( String[] args) {
      Add A2 = new Add(10,20);
      A2.sum(A2)
  } }
```

OP :- 30

3b) An array is a container object that holds a fixed number of values of single type.
The length of an array is established when the array is created. After creation, length fixed.

Ex :— // To sreach an element from array

```
import Java.util.Scanner;
  Class Array
  { public static void main (String[] args) {
  int i, n, flag, sreach;
  Scanner sc = new Scanner (System.in)
  System.out.println("Enter ~~n=elemen~~ how many elmt");
    n = sc.nextInt();
Int[] ~~Array~~ a = new ~~Array~~ Int[n];
  System.out.println("Enter elements")
  for (i=0; i<n; i++)
  {
    a[i]= sc.nextInt();
  }
  System.out.println("Element to be sreached");
   sreach = sc.nextInt();
  for (i=0; i<n; i++)
  {
    if (a[i]==sreach)
     {
      System.out.print("Element" + sreach + "at
                        position" + i);
      flag=1
      break;
3 }
```

```
if (flag==0)
  {
   system.out.println("element not found");
  }
 }
}
```

Op :- Enter n value =3
Enter elements
60
61
69
Enter element to be sreached
60
60 found at position 0.

Q(a) The two most modes of passing arguments to methods are pass by value and call by reference.

1. pass by value :-

In pass by value concept, the method is called by passing a value. so, it is called pass by value. It doesn't affect the original parameter

Ex:-
```
class PBV
{
 int a=100;
 void change(int a)
 {
  a=a+100;
 }
 public static void main(String[] args)
 {
```

```
PBV b = new PBV();
System.out.print("Before passing value " + b.a);
b.change(1000); //(passing value)
System.out.println("after passing " + b.a);
}
}
```

O/P :— 100
100

(ii) call by Reference :— By call by Reference original value is changed if we made changes in method.

Ex :—

```
class CBR
{
  int a=100;
  CBR
  void change(CBR A)
  {
    a = a+100
    A.a = A.a +100;
  }
  public void static main(String[] args)
  {
    CBR A = new CBR()
    A.change(A)
System.out.println(A.a)
```

O/P :— 200

(45) The Return Keyword finish the execution of method, and can be used to return value from the method

```
class Test {
int a;
Test(int i)
{
 a=i;
}
Test IncrByTen()
{
Test temp= new Test (a+10);
 return temp;
}
}
class Final
{
public static void main (String [] args)
{
  Test obj = new Test (10);
  Test obj2;
   obj2 = obj. IncrByTen ();
  System.out.println( obj.a);
  System.out.println (obj2.a);
   obj2 = obj2. IncrByTen ();
  system.out.println( obj2.a);
}
}
```

1) abstraction
Encapsulation
Inheritance
polymorphism
class
object

2) Jvm extends for Java virtual machine. It loads, verifies and executes Java bytecode.

Javac :- Javac does the Java compilation. It is used to check the errors at complie time

syntax :- Javac <programname>.java.

Java :- it is use to run the sucessfully compiled Java program

syntax :- Java <programname>

3. Default constructor :- A constructor is an Instance of class. constructor name is same as class name. Default constructor is a constructor which doesn't have any argument

syntax :- class <classname>
{
- - -
<classname> () // Default constructor
{
//code
}
}

4) Ternary operator provides an abbreviated syntax to evaluate a true or false condition, and return a value based on the Boolean result.

new object is a instance of class by allocating memory for a new object and returning a reference to that memory.

```
classname<object> = new classname();
```

5) Java applications obtain objects in memory as needed. It is the task of garbage collection in the JVM to automatically determine what memory is no longer being used by a Java application and to recycle this memory for other users.

2nd unit

1. Interface :—

An interface is an abstract "class" which doesn't have any body.

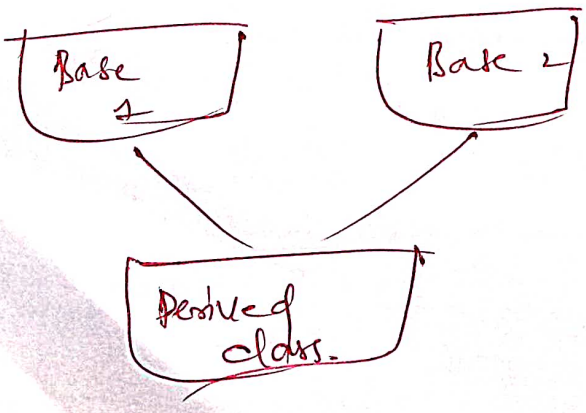
* It is used to group related method with "empty" bodies

* To access Interface ~~use~~ method, the interface must be "implemented" by another class we use keyword implements

Syntax :—

```
class A
{
  ---
  ---
}
class B implements A
{
  ---
  ---
}
```

Multiple Inheritance :— A derived class is derived from more than one base class is called as multiple Inheritance



Ex:-

```
Interface AnimalEat  // Base class
{
void eat();
}
Interface AnimalTravel  //Base class
{
void travel();
}
class Animal implements AnimalEat, AnimalTravel
                                  //Derived class
{
public void eat()
{
System.out.println("Animal is eating");
}
public void travel()
{
system.out.println("Animal is travelling");
}
}
```

b) In java, method overriding occurs when a subclass (child class) has the same method as the parent class.

That means child class implemented parent class.

* It is used for runtime polymorphism.
* Is-A Relation ship

Eg:-

```
class vehicle
{
void run()
{
System.out.println("vehicle is running");
}
}

class bike extends vehicle

{
public static void main( String[] args)
  //creating child class Instance
{
  Bike obj = new Bike (); //calling method with child class
  obj.run ();

}
}
```

24) package is a collection of predefine classes and Interface. Java provides different types of Inbuilt packages for different task.

programmer can also create own packages

There are many packages

Ex:- lang package,
io package,
util package
sql package etc.,

steps to create a package :-

1. First create a directory within name of package
2. create a java file in newly created directory
3. In this java file you must specify the package name with the help of package keyword.
4. Save the file with same name of public class
5. Now you can use these package in program.

2) Interface :- Interface is an abstract [illegible] which doesn't have any body

* used keycoord = implements
* It is blue print of class. It has static constants and abstract methods
* Is-A Relationship.

Ex :-

```
Interface shape
{
 void area ();
}
class Triangle implements shape
{
 int b = 10;
 int h = 20;
 double ar =0;
 public void area()
 {
  ar = 0.5 * b * h
  System.out.println(ar);
 }
}
class Rectangle implements shape
{
 int l = 10;
 int br = 20;
 double are = 0;
public void area()
{
```

```
        area = l * br ;
        System.out.println (a);
        }
    }
    class Demo
    {
    public static void main (String [] args)
    {
      Shape s ;
      S = new Triangle ();
      s.area()
      S = new Rectangle ();
      s.area();
    }
}
```

3a) super keyword in java is a reference variable which is used to refer the immediate parent class object.

Ex:-

```
class Animal
{
  void eat()
  {
   system.out.println("Animal is eating");
  }
}
class cat extends Animal
{
 void eat()
 {
  system.out.println("cat milk");
  super.eat();
 }
```

```
class finalone
{
public static void main (String[] args)
{
   Cat A = new Cat();
   A.eat();
}
}
```

35) A class which is declared with Abstract keyword is known as an Abstract class in java.

* It can have abstract and non-abstract method.
* It needs to be extended and its method implemented.
* It cannot be Instantiated.
* It can have Constructors and static methods, final methods.

Ex :-

```
Abstract class Bike
{
abstract void run();
}
class pulsar extends bike
{
void run()
{
System.out.println("Drive slowly");
}
public static void main(String[] args)
{
  Bike obj = new pulsar();
obj.run();
}
```

40) Java nested Interfaces:-

An Interface is accessed with another Interface is called as nested Interface

* The nested Interface must be public. if it is declared inside the interface, but it can have any access modifier if declared within the class.
* Nested interfaces are declared static.

Syntax:-

```
interface interface-name
{
- - - -
interface nested-interface-name
{
- - - -
}
}
```

Ex:-

```
Interface Whatsapp {
    void show();
    interface Message {
    void msg();
    }
}
Class chatting implements whatsapp.message
{
public void msg()
{
System.out.println ("Hello");
}
public static void main (String [] args) {
whatsapp.message M = new Chatting();
M.msg();
}
```

45) Final Keyword :—

The final keyword in java is used to restrict the user.

The final keyword can be used in many context

1. variable
2. method
3. class.

1. variable :— if final keyword used for variable u can't modify the value of variable.

Ex :—
```
class A
{
final int a=10;
void add()
{
a=10;
}
public static void main(String args)
{
A obj = new A();
obj.A();
}
}
```

op :— Compile Time error

2. Method :— If final keyword used for method you can't write another method in the program.

Ex :—
```
class A
{
int a=10;
final void add()
{ a=10;
```

```
}}
class B extends A {
    void add()
    {
    System.out.println(a);
    }
    public static void main(String[] args)
    {
      A obj = new A();
      A.add();
      A.add();
    }
}
```

op :- compile time error

2. class :- if final keyword is used for class then we can't write another class in program

```
Final class A
{
  int a=10;
  void run()
  {System.out.println(a);
  }
}
class B extends A
  {
  void run()
  {
    System.out.println(a);
  }
  public static void main(String[] args)
  {
    A obj = new A();
    A.run()
```

op :- compile-time error.

1) Dynamic method dispatch :—

The mechanism in which a call to an overridden method is resolved at runtime instead of compile time.

It is Run-time polymorphism.

2) Final keyword once assigned it can't change its value

* if Final used on variable then no updating
* if Final used on method then no more method
* if Final used on class then no more class

3) use :—

* To access the data of parent class to child class if the class name is same.
* To explicitly call default constructor / parameterized from sub class constructor.

4)

interface	class
1. Interface have only abstract method.	1. Class have both abstract and concrete methods
2. Multiple Inheritance supports	2. multiple Inheritance not support
3. Interface Keyword for Interface.	3. class Keyword for class
Interface <name> { - - - }	4. class <class-name> { - - - }
5. If doesn't have constructor.	5. If has constructor

5) package :- package is a collection of predefined classes and Interface.

Java provides different types of inbuilt Java package

Ex :- io package,
util package,
sql package.
lang package etc.)