DAC AUG 2024
LAB CONTENT - CORE JAVA

By- DEEPSHIKHA KURRE
PRN- 240850120042

DAY 1

Example- Hello.java

```
public class Hello
{
	public static void main(String[] args)
	{
	System.out.println("HELLO WORLD,THIS IS DEEPSHIKHA");
	}
}
```

Data type-

Example- Employee.java

```
class Employee
{
	int empid;
    String name;
    float sal;
}
 public class TestEmployee
{
    public static void main(String[] args)
    {
	Employee e1 = new Employee();

	e1.empid = 1001;
	e1.name = "nsnathan";
	e1.sal = 30000;
	System.out.println("First object");
```

```
        System.out.println(e1.empid);
        System.out.println(e1.name);
        System.out.println(e1.sal);

     System.out.println("second object");
        Employee e2 = new Employee();
        e2.empid = 1002;
        e2.name = "nathan";
        e2.sal = 40000;
        System.out.println(e2.empid);
        System.out.println(e2.name);
        System.out.println(e2.sal);
   }
```

Variable-

1) local variable- variable declared inside the method is called local variable.

Example- VariableExample.java

```
class VariableExample
{
public static void main(String[] args)
{
    int id;
    String name;
    String desig;
    float sal;
    char grade;
    boolean ispromoted;
   id =1001;
   name = "nsnathan";
   desig = "faculty";
   sal = 30000;
   grade = 'A';
   ispromoted = true;
   System.out.println("employee id " + id);
   System.out.println("name   " +name);
   System.out.println(desig);
```

```
   System.out.println(grade);
   System.out.println(sal);
   System.out.println(ispromoted);
}}
```

3) static variable

Example-

```
main()
{
  int age = 20;
  f1()
  printf(age)
}
f1()
{
  printf(age);
}
```

Example- Adding.java

```
public class Adding
{
      public static void main(String[] args)
      {
            int num1 = 50;
            int num2 = 20;
            int sum = num1 +num2;
            System.out.println("sum of "+ num1 + " + "+num2+" =" + sum);
            if(num1>num2) {
                System.out.println(num1 + "is greater");
            }
            else
                System.out.println(num2 + "is greater");
}
```

Core java Assignment 1

1. Basic Arithmetic Calculator: Write a method that performs addition, subtraction, multiplication, and division on two integers and returns the results.

2. **Sum of Even Numbers:** Write a method that takes an integer `n` and returns the sum of all even numbers from 1 to `n`.

3. **Character Count:** Implement a method to count the occurrences of a specific character in a given string.

4. **Multiplication Table:** Create a method to generate and print the multiplication table for a given integer up to 10.

5. **Sum of Digits:** Write a method that calculates the sum of all digits in a given integer.

6. Temperature Conversion: Create a method to convert a temperature from Celsius to Fahrenheit and return the result.

7. Grade Calculator: Develop a program that computes the average grade from the user and determines the corresponding letter grade.

8. Palindrome Checker: Create a method to determine if a given string is a palindrome.

9. Factorial Calculator: Write a method to calculate the factorial of a non-negative integer using a loop.

10. Find Maximum Value: Implement a method to find and return the maximum value from given three numbers

11.Write a Java method to find and return the maximum and minimum values from an integer array.

12.Implement a Java method to reverse the elements of a given integer array in-place.

13.Given an integer array where all elements appear twice except one, write a method to find the single element that appears once.

14.Write a Java method to merge two sorted integer arrays into a single sorted array.

15.Implement a Java method to find the Kth largest element in an unsorted integer array.

DAY 2-

1.Operator-

Example- OperatorExample.java

```
package assignment1;
import java.util.*;
public class ArithmaticOp
{
	public static void arith(int a,int b)
	{
		System.out.println("Addition is = " + (a+b));
		System.out.println("Substraction is = " + (a-b));
		System.out.println("Multiplication is = " + (a*b));
		System.out.println("Division is = " + (a/b));
		System.out.println("Modulas is = " + (a%b));
	}

  public static void main(String[] args)
  {
	Scanner s = new Scanner(System.in);
	System.out.println("ENTER FIRST NUMBER = ");
	int c=s.nextInt();
	System.out.println("ENTER SECOND NUMBER = ");
```

```
        int d=s.nextInt();
        arith(c,d);
        s.close();
    }
}
```

Example- UniqueElementInArray.java

```
package assignment1;
public class UniqueElementInArray
{
        public static int findSingle(int[] nums)
        {
            int result = 0;
            for (int num : nums)
            {
                result ^= num; // XOR all elements
            }
            return result; // The single element that appears once
        }

        public static void main(String[] args)
        {
            int[] nums = {2, 2, 1, 4, 1, 5, 5};
            int singleElement = findSingle(nums);
            System.out.println("The single element is= " + singleElement);
        }
    }
```

1.7 Control statement

if

if else

if else if

1.8 method()- Example-

```
import java.util.Scanner;
public class MethodExampleFindMax
{
	public static int findmax(int c,int d)
	{
		int max=0;
		if(c>d)
		    max = c;
		else
			max = d;
		return max;
	}
	public static void main(String[] args)
	{
		Scanner s =new Scanner(System.in);
		int a;
		int b;
		a = s.nextInt();
		b = s.nextInt();
		int m=findmax(a,b);
		System.out.println("max "+ m);
	}
}
```

1.8.1. static method- if the method is static, we can call the method by the method name or can call the method by classname.method name.

Example- methodExample.java

```
class methodExample
{
	//user defined static method
	static void findmax()
	{
	syso("inside the findmax”);
```

```
    }
    //inbuilt static method
public static void main(String[] args)
   {
        syso("method example");
        methodExample.findmax();
    }
}
```

1.8.2. Non static method /instance method- To call non static method you require the object reference of the class.

Example-

```
class methodExample
{
    void findmax()
    {}
    public static void main()
    {
     methodExample m = new methodExample();
    m.findmax();
    }
}
```

1.9 How to read data from the user-

Example- ReadDataFromUser.java

```
import java.util.Scanner;
public class ReadDataFromUser
{
    public static char findGrade(int p )
    {
        char grade;
       if(p>=90)
          grade = 'A';
```

```
            else if(p>=80 && p<90)
                grade = 'B';
            else if(p>=60 && p<80)
                grade = 'C';
            else
                grade = 'D';
            return grade;
      }
public static void main(String[] args)
{
             Scanner s =new Scanner(System.in);
             System.out.println("enter the percentage");
             int per = s.nextInt();

              char g=findGrade(per);
             System.out.println("grade = "+g);
             System.out.println("end of main");
//how to read data from the user

      Scanner s =new Scanner(System.in);
       System.out.println("enter the empno"); int empno =s.nextInt();
       System.out.println("enter the sal"); float empsal = s.nextFloat();
       System.out.println("enter the name"); String name = s.next();
      System.out.println("empno = "+ empno); System.out.println("empsal =
" + empsal); System.out.println("empname = " + name );
      }
}
```

Assignment 2(oops)

1. Book Management System:

Create a Book class with attributes like title, author, price, and ISBN. Implement methods to set and get book details, and a method to apply a discount to the book price.

2. Bank Account Management:
Design a BankAccount class with properties such as accountNumber, accountHolderName, balance. Include methods to deposit, withdraw, and display the balance.

3.Student Grading System:
Develop a Student class with attributes like name, rollNumber, marks in different subjects. Create methods to calculate the total and average marks, and determine the grade based on the average.

4. Employee Management:
Create an Employee class with attributes like employeeID, name, designation, and salary. Write methods to increase salary, display employee details, and calculate the annual bonus.

5. Car Rental System:
Implement a Car class with properties such as carModel, registrationNumber, rentalRatePerDay, and availability. Write methods to rent a car, return a car, and display the rental charges.

6. Library System:
Design a Library class with attributes bookList (list of books available), memberList (list of library members), and methods to issue a book to a member, return a book, and search for books by title or author.

7. Shopping Cart System:
Develop a Product class with attributes like productID, productName, price, and quantity. Implement methods to add products to the cart, remove products from the cart, and calculate the total bill for all products in the cart.

8. Online Course Enrollment:
Create a Course class with properties such as courseID, courseName, instructorName, and fee. Implement methods for students to enroll in the course, view enrolled students, and calculate the total revenue generated by the course enrollments.

9.Inventory Management System:
Create a ProductInventory class with attributes like productID, productName, stockQuantity, and price. Write methods to add stock, remove stock, and display current stock levels for a product.

10. Hotel Room Booking System:
Design a Room class with attributes such as roomNumber, roomType, ratePerNight, and availabilityStatus. Write methods to book a room, cancel a booking, and calculate the total amount for a stay based on the number of nights booked.

DAY 3-

1.static method- using method name or class name.method name
no need of object reference.
Example-

```
public class StaticMethodExample
{
	public static void findOddEven()
	{
		System.out.println("inside the oddeven method");
		int num = 30;
		if(num%2==0)
			System.out.println("even");
		else
			System.out.println("odd");
}
	public static void main(String[] args)
```

```
    {
        System.out.println("method example");
        findOddEven();
    }
}
```

2. Non static method- need object reference to make a call to the method.

Example-

```
public class NonstaticMethodExample
{
    public void findmax()
    {
        System.out.println("inside findmax");
    }
    public static void main(String[] args)
    {
        NonstaticMethodExample n = new NonstaticMethodExample();
        System.out.println("inside main");
        n.findmax();
    }
}
```

3. conditional statement-

3.1 switch case- 1.add

2.delete

3.update

4.retrieve

5.exit

Example- (A) - switch case-

```
user input ch= 3
do
{
switch(ch)
{
  case ch ==1:
  break;
  case ch == 2: syso(data deleted)
```

```
  default: syso(invaili input)
}
```

(B) while=

```
while(i!=5)
```

3.2 control and looping statements- break, continue, return

Example-(A)-

```
int i = 0;
while(i<10)
{
 i++
}
```

(B) for loop-

```
for(int i =0 ;i<10;i++)
{
//logic part
}
```

(C) do while loop-

```
i =0
do
{   i= i+1;
}while(i<5)
```

(D) ControlExampleFindOddEven.java

```
public class ControlExampleFindOddEven
{      // find  the given data is odd or even
       public static void main(String[] args)
       {        int num = 31;
                if(num%2 ==0)
                        System.out.println("the given number is even");
                else
                        System.out.println("the given number is odd");
       }
}
```

4.Type casting-
4.1) implicit casting(widening)

```
int i = 20;
float f = i;
```

4.2) explicit casting(narrowing)

```
float f = 40.5;
int i = (v int)f;
```

Example-

```
class jdbcmain
{
public static void main(String[] args)
{
    switch(ch)
   {
    case 1: jdbcManagement.create();
    delete();
    update();
    display();
}
class jdbcManagement
{
       public static void create()
       {        //logic part
       }
       public static void delete()
       {
       }
}
}
```

5. Reference data type-
5.1 Array- How to declare?, How to initialize?, How to retrieve?
int array, float array, char array, String array.

5.2 How to create array- int[] mark = new int[5];
5.3 How to initialize- mark[0]= 10;
mark[1]=20;
mark[2]=40;
mark[3]=50;
mark[4]=30;

Example- TO TAKE USER INPUT-
(A)

```
for(i=0;i<5;i++)
{
  mark[i]= sc.nextINt();
}
```

(B)

```
int[] mark= {10,20,30,40,50}
```

5.4 How to retrieve-

```
for(int i =0;i<5;i++)
{
 syso(mark[i);
}
for(int ele    :mark)
{
  syso(ele);
}
```

DAY 4-

1.Classes and Objects-
1.1 Object- object is a real word entity, object has properties(data) and behavior (method). object is a instance of the class. object takes memory.

1.2 Class- 1) class is a template or blueprint
2) class is used to create object
3) it is a representation of object
Example- Student.java

```
class Student
{
  int sid;
   String name;
   Int age;
   Int mark;
   String cname;
}
```

Example 2- Mobilephone.java

```
class Mobilephone
{
	Int price;
	String model;
	String color;
}
```

Example3 - Cdacblr.java

```
class Cdacblr
{
	int course_id;
	String coursename;
	int noofstudent;
	int noofmodule;
	float fees;
public static void main()
{
	Cdacblr c1 = new Cdacblr();
	c1.course_Id = 1001;
	c1.coursename = "DAC";
	c1.no_of_student =200;
	c1.modules = 8;
}
```

DAY 5-

2nd Assignment(Overloading)

1) Calculator Application: Design a simple calculator application that can perform various arithmetic operations (e.g., addition, subtraction, multiplication, division). Implement method overloading in a Calculator class where methods with the same name, calculate(), handle different types and numbers of parameters:

calculate(int a, int b) for basic integer arithmetic operations.
calculate(double a, double b) for operations involving double precision numbers.
calculate(int a, int b, int c) for operations involving three integers.

2) Library Management System: Develop a library management system where you can add books to the library database. Implement method overloading in an AddBook class to handle different scenarios of adding books:

addBook(String title, String author) to add a book with a title and author.
addBook(String title, String author, int year) to include the publication year.
addBook(String title, String author, int year, String genre) to include the publication year and genre of the book.

Assignment(Overriding)

1)Employee Salary Calculation: Design an employee management system where different types of employees (e.g., FullTimeEmployee, PartTimeEmployee, ContractEmployee) have different salary calculation rules. Create a base class Employee with a method calculateSalary(), and derive subclasses that override this method to implement specific salary calculations based on the type of employee.

2)Bank Account Management: Develop a banking system that handles various types of bank accounts (e.g., SavingsAccount, CheckingAccount,

BusinessAccount). Each account type has a different method for calculating interest or fees. Define a base class BankAccount with a method calculateInterest(), and have each subclass override this method to provide account-type-specific calculations.

3)Electricity Bill Calculation: Implement a billing system for electricity where different types of customers (e.g., ResidentialCustomer, CommercialCustomer, IndustrialCustomer) have distinct billing rates and rules. Create a base class ElectricityBill with a method calculateBill(), and derive subclasses that override this method to calculate the bill based on the customer's specific requirements and usage patterns.

DAY 6-

1.1 Encapsulation- Process of bundling data and the method which operates on the data together is called encapsulation.
Example 1 - Student.java

```
import java.util.Scanner;
public class Student
{
	private int id;
	private String name;
	public int getId() {
		return id;
	}
	public String getName() {
		return name;
	}
	public void setId(int id) {
		this.id = id;
	}
	public void setName(String name) {
		this.name = name;
	}
```

```
	public void display()
	{
		System.out.println(id+name);
	}
	public void readstudent()
	{
		Scanner s = new Scanner(System.in);
		id = s.nextInt();
		name = s.next();
	}
}
public class TestSTudent
{
	public static void main(String[] args)
	{
		Student s = new Student();
		s.setId(1001);
		s.setName("nathan");
		System.out.println(s.getId() +" "+ s.getName()) ;
	}
}
```

1.2 Inheritance- Inheritance a process of acquiring the properties(data) and the behavior(method) from one class to the other class.creating new class from the existing class so that the new class get the properties and the behavior of existing class.creating sub class object accessing the sub class features(properties and behavior) and super class features(properties and behavior) this is called inheritance.

1.3 Advantage- time, code reusability ,achieving runtime polymorphism.

Syntax-

```
class A
{
}
class B extend A
{
```

```
}
```
Example-
```
public class A
{
    int a;
    void printA()
    {
        System.out.println("A class method "+a);
    }
}
 class B extends A
{
    int b;
    void printB()
    {
        System.out.println("B class method "+b);
    }
}
 class C extends B
 {
     int c;
     void printC()
     {
        System.out.println("B class method "+c);
     }
 }
 class Testinheritance
 {
          public static void main(String[] args)
          {
                  B b = new B();
                  b.b = 20;
                  b.printB();
                  b.a = 10;
                  b.printA();
```

```
                C c = new C();
                c.a = 10;
                c.b = 20;
                c.c = 30;

                c.printA();
                c.printB();
                c.printC();
        }
 }
```

1.4 Types of inheritance-

1.4.1 is a relation

1.4.2 has a relation

1.5 Varieties of inheritance-

Example-

```
class Bank
{
        int aid;
        String aname;
        float balance;
        public Bank(int aid, String aname, float balance)
        {
                this.aid = aid;
                this.aname = aname;
                this.balance = balance;
        }
        void dispBankDetails()
        {
                System.out.print(aid+aname+balance+ " ");
        }
        void withdraw(int amt)
        {
```

```
                balance =balance -amt;
                System.out.println(balance);
        }
}
class Savings extends Bank
{       float ri;

          Savings(int aid,String aname,float balance,float ri)
          {
                  super(aid,aname,balance);
                  this.ri = ri;
          }
         void printsavingDetails()
         {
                 super.dispBankDetails();
                 System.out.println(ri);
         }
          void findInterest()
          {
                  float interest = balance*ri;
                  System.out.println(interest);
          }
}
class Current extends Bank
{
        int overduefees;
        public Current(int aid, String aname, float balance,int overduefees)
        {
                super(aid, aname, balance);
                this.overduefees = overduefees;
        }
        void printCurrentDetails()
        {
                super.dispBankDetails();
                System.out.println(overduefees);
```

```
        }
          void findoverdraft(int amt)
          {
                if(balance<amt)
                      balance= balance-500;
                System.out.println(balance);
          }
}
public class InheritanceMain
{
        public static void main(String[] args)
        {
              Savings s = new Savings(1001,"nsnathan",40000,0.02f);
              s.printsavingDetails();
              s.withdraw(2000);
              s.findInterest();
              System.out.println("------------------------");
              Current c = new Current(2001,"shan",20000,500);
              c.printCurrentDetails();
              c.findoverdraft(3000);
        }
}
```

DAY 7-

1. Constructor behavior during inheritance-

Example-

```
class A
{
  int a;
  A(int a)
  {   this.a = a; }
  }
  void dispA(){   }
class B extends A
```

```
{
   int b;
  B(int a,int b )
  { super(a)
this .b = b}
}
main()
{
  B b = new B(10,20)
  b.dispA();
}
```

2. Polymorphism- poly = many, morphism = form(method). More than one method can have same name with different functionality.

2.1 Method overloading- in single class, single class has two different method with same name. method name can be same at least the no of parameters and the type of parameters should be different.

Example-

```
class soring
{
static void sort(int[] n)
{ - for sorting integer array
}
static void sort(float[] f)
{ - for sorting float array
}
static void sort(String  s)
{ - for
sorting string array
}
}
main()
{
  String[]  = {'c++',"java",'python'}
  sortSting(subject)
```

```
}
class A
{
   void findtax(){}
   void dispEmp();
}
class B extends A
{
  void findtax(){}
}
```

3. Method overriding- During inheritance, method name can be same the no of parameters ,type of parameters also be same.

Example1 -

```
class A
        {
        void print(String name){ }
        }
class B extends A
        {
        void print(String name);
                {       super.print();
                }
        }
public class  C
{
        public static void main(String[] args)
        B b = new B();
        b.print();
}}
```

Example 2-

```
Bank
{
void findinterest()
{
```

```
    interest = balance*0.02f;
}
}
class sbi extends Bank
{
void findinterest()
{
    interest = balance*0.03f;
}
}
class icici extends Bank
{
void findinterest()
{
    interest = balance*0.01f;
}
}
public static void main()
{
  sbi s = new sbi();
  s.findInterest()
  Icici i = new Icici();
  i.Interest():
}
```

DAY8-

2nd Assignment 2 -HAS A RELATION)

Problem Statement 1: Car and Engine

Scenario: You are designing a system to manage cars, and each car has a single engine. The engine, however, does not need to know about the car that it's in.

Task: Create two classes in Java: Car and Engine.

The Car class should have attributes like model, make, and a reference to an Engine object.
The Engine class should have attributes like engineNumber, type, and power.

Problem Statement 2: Library and Book

Scenario: You are creating a system to manage libraries and the books they contain. Each library has a list of books, but each book does not need to reference back to the library.

Task: Create two classes in Java: Library and Book.

The Library class should have attributes such as name, location, and a list of Book objects.
The Book class should have attributes such as title, author, and isbn.

Abstraction

Problem Statement: 1 Payment Processing System

You are developing a payment processing system that supports various payment methods. Each payment method has a specific way of processing payments, but they all follow a common set of operations.

Create an abstract class PaymentMethod with an abstract method processPayment(double amount). Then, create two concrete subclasses: CreditCard and PayPal.

The PaymentMethod class should have the abstract method processPayment(double amount).
The CreditCard class should implement processPayment(double amount) to handle credit card payments.

The PayPal class should implement processPayment(double amount) to handle PayPal payments.

Problem Statement:2 Employee and Task Management System
You are creating a system to manage employees and the tasks they perform. Each employee can have different types of tasks assigned, but each task type has its own way of tracking progress.

Create an abstract class Task with an abstract method trackProgress(). Then, create two concrete subclasses: DevelopmentTask and DesignTask.

The Task class should have the abstract method trackProgress().
The DevelopmentTask class should implement trackProgress() to track the progress of software development tasks.
The DesignTask class should implement trackProgress() to track the progress of design-related tasks.

Problem Statement:3

Design a Java application for a vehicle rental service that supports different types of vehicles, such as cars and motorcycles.

Create an abstract class Vehicle with properties licensePlate and rentalPricePerDay, and an abstract method calculateRentalCost(int days).
Implement two concrete subclasses: Car with additional properties numberOfDoors and isAutomatic, and Motorcycle with properties engineDisplacement and hasSidecar. Override calculateRentalCost() in both subclasses to compute the rental cost based on the number of days and specific vehicle attributes.

Develop a RentalService class that maintains a list of Vehicle objects, with methods to add vehicles and calculate the total rental cost for a given number of days for each vehicle.

1. Inheritance-
   1.1 is a relation
   1.2 has a relation

Example-

```
public class Employee
{
   int empno;
   String name;
   float sal;
   Address add;
public Employee(int empno, String name, float sal, Address add)
        {
                this.empno = empno;
                this.name = name;
                this.sal = sal;
                this.add = add;
        }
void dispEmp()
        {
        System.out.print(empno+name+sal+add.city+add.state+add.pin);
                //add.dispAddress();
        }
public static void main(String[] args)
        {
                Address add = new Address("Bangalore","Karnataka",560038);
                Employee e = new Employee(1001,"shan",30000,add);
                e.dispEmp();
        }
}
public class Address
{
```

```
        String city;
    String state;
    int pin;
        public Address(String city, String state, int pin) {

                this.city = city;
                this.state = state;
                this.pin = pin;
        }
        void dispAddress()
        {
                System.out.println(city+state+pin);
        {
}
```

2. Abstraction- abstraction a process of hiding the implementation details showing only the functionality to the user.

2.1 abstract class

2.2 Interface

Format-

```
abstract class emp
{
  abstract void diplay();
  void print()
  {}
}
contract extends emp
{
  void display()
  {}
}
```

Example-

```
public class NetBankingPayment
{
        void payment(float amount)
        {
```

```
            System.out.println("Processing payment  "+ amount +" via net
Baning");
        }
}
class SBI extends NetBankingPayment
{
        void payment(float amount)
        {
            System.out.println("Processing  payment   "+ amount + " via
SBI");
        }
}
class ICICI extends NetBankingPayment
{
        void payment(float amount)
        {
            System.out.println("Processing  payment   "+ amount + " via
ICICI");
        }
}
class HDFC extends NetBankingPayment
{
        void payment(float amount)
        {
            System.out.println("Processing  payment   "+ amount + " via
HDFC");
        }
}
class testRuntimePolymorphism
{
        public static NetBankingPayment createbankobject(String bname)
        {
            if(bname.equals("SBI"))
            {
              return   new SBI();
```

```
        }
    else if(bname.equals("ICICI"))
    {
        return new ICICI();
    }
    else if(bname.equals("HDFC"))
    {
        return  new HDFC();
    }
    else
        return null;
    }
    public static void main(String[] args)
    {
        NetBankingPayment n;
        Scanner s  = new Scanner(System.in);
        System.out.println("enter the bank name to be processed");
        String bname=s.next();
         n =createbankobject(bname);
         n.payment(1000);
    }
}
```

DAY9-

Collection- 1.Stack

Example-

```
import java.util.Stack;
public class StackExample
{
    public static void main(String[] args)
    {
        Stack<Integer> s = new Stack<>();
        s.push(20);
        s.push(30);
        s.push(40);
```

```
s.push(10);
s.push(60);
for(Integer ele :s)
{
        System.out.println(ele);
}
System.out.println("element poped");
 Integer  n =s.pop();
System.out.println(n);
System.out.println("after pop");
for(Integer ele :s)
{
        System.out.println(ele);
}
System.out.println("peek");
n =s.peek();
System.out.println(n);
System.out.println("after peek");
for(Integer ele :s)
{
        System.out.println(ele);
}
s.remove(2);
System.out.println("after remove");
for(Integer ele :s)
{
        System.out.println(ele);
}
s.set(2, 50);
System.out.println("after update");
for(Integer ele :s)
{
        System.out.println(ele);
}
s.add(2,40);
```

```
                System.out.println("after insert");
                for(Integer ele :s)
                {
                        System.out.println(ele);
                }

        }
}
```

2.Queue

Example-

```
import java.util.PriorityQueue;
public class QueueExample
{
        public static void main(String[] args)
        {
        PriorityQueue<Integer> pq = new PriorityQueue<>();
                pq.add(10);
                pq.add(20);
                pq.add(30);
                pq.add(40);
                for(Integer ele :pq)
                {
                        System.out.println(ele);
                }
                System.out.println("after poll");
                Integer n=pq.poll();
                System.out.println(n);
                System.out.println("--------------------");
                for(Integer ele :pq)
                {
                        System.out.println(ele);
                }
                System.out.println("after peek");
```

```
                n=pq.peek();
                System.out.println(n);
                System.out.println("--------------------");
                for(Integer ele :pq)
                {
                        System.out.println(ele);
                }
                System.out.println("after remove");
boolean rm=pq.remove(40);
                System.out.println(rm);
                System.out.println("--------------------");
                for(Integer ele :pq)
                {
                        System.out.println(ele);
                }
        }
}
```

3. Hash Map

Example-

```
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class HashMapExample
{
        public static void main(String[] args)
        {
        HashMap<Integer ,Integer > hm = new HashMap<>();
                hm.put(1, 10);
                hm.put(1, 20);
                hm.put(3, 30);
                hm.put(4, 40);
                Set<Entry<Integer, Integer>> s= hm.entrySet();
```

```
            for(Map.Entry<Integer, Integer> h: s )
            {
            System.out.println(h.getKey() + " "+ h.getValue());
            }
            System.out.println("after given the key value 2");
            Integer value =hm.get(2);
            System.out.println(value);
            System.out.println("after given the key value 1");
            hm.remove(1);
            for(Map.Entry<Integer, Integer> h:  hm.entrySet())
            {
            System.out.println(h.getKey() + " "+ h.getValue());
            }
            System.out.println("after update");
            hm.put(3,100);
            for(Map.Entry<Integer, Integer> h:  hm.entrySet())
            {
            System.out.println(h.getKey() + " "+ h.getValue());
            }
            System.out.println("after insert");
            hm.put(5, 50);
        for(Map.Entry<Integer, Integer> h:  hm.entrySet())
        {
            System.out.println(h.getKey() + " "+ h.getValue());
        }
}
}
```

DAY10-

File Handling

Example- Read Write into file

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.util.Scanner;
public class ReadWritePremitive
{
        public static void main(String[] args)
        {
             Scanner sc=new Scanner(System.in);
            String filePath = "primitives.dat";
            // WRITING PREMITIVE DATA TYPES TO FILE
            try (DataOutputStream dos = new DataOutputStream(new
FileOutputStream(filePath)))
            {
                System.out.println("ENTER ANY INTEGER VALUE= ");
             int intValue = sc.nextInt();
             System.out.println("ENTER AND DOUBLE VALUE= ");
                double doubleValue = sc.nextDouble();
                System.out.println("SET THE BOOLEAN VALUE AS
TRUE OR FALSE = ");
                boolean booleanValue = sc.nextBoolean();
                // Writing data
                dos.writeInt(intValue);
                dos.writeDouble(doubleValue);
                dos.writeBoolean(booleanValue);
                System.out.println("Data written to " + filePath);
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
            try(DataInputStream dis = new DataInputStream(new
FileInputStream(filePath)))
            {
                // READING PREMITIVE DATA
                int intValue = dis.readInt();
                double doubleValue = dis.readDouble();
```

```
                boolean booleanValue = dis.readBoolean();

                // DISPLAYING
             System.out.println("READING FROM" + filePath + ":");
                System.out.println("Int= " + intValue);
                System.out.println("Double= " + doubleValue);
                System.out.println("Boolean= " + booleanValue);
              } catch (IOException e) {
                e.printStackTrace();
              }
            sc.close();
            }
    }
```

Example-Append file

```
    import java.io.FileWriter;
    import java.io.IOException;
    import java.io.FileReader;
    import java.io.BufferedReader;
    import java.util.Scanner;
    public class AppendFile
    {
            public static void main(String[] args)
            {
               Scanner sc = new Scanner(System.in);
               String toAppend;
               System.out.println("ENTER FILE NAME TO APPEND= ");
               String fileName=sc.next();
               try (FileWriter writer = new FileWriter(fileName, true))
               {
                  System.out.println("ENTER STRINGS TO APPEND TO
THE FILE. TYPE 'EXIT' TO STOP.");
                  while (true)
                  {
                        toAppend = sc.nextLine();
```

```
                    if (toAppend.equalsIgnoreCase("EXIT"))
                    {
                        break;
                    }
                    // APPENDING
                    writer.write(toAppend + "\n");
                }
            } catch (IOException e) {
                System.out.println("ERROR " + e.getMessage());
            }
            // READING FROM FILE
            try (BufferedReader reader = new BufferedReader(new
FileReader(fileName)))
            {
                System.out.println("\nCONTENTS OFTHE FILE= ");
                String line;
                while ((line = reader.readLine()) != null)
                {
                    System.out.println(line);
                }
                System.out.println(fileName + " FILE SUCCESSFULLY
APPENDED.");
            } catch (IOException e)
            {
                System.out.println("ERROR IN READING " +
e.getMessage());
            }
            sc.close();
        }
    }
```

DAY11-

Socket Programming

Example- Client.java-

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintStream;
import java.net.Socket;
public class Client1
{
	public static void main(String[] args) throws IOException
	{
		Socket s= new Socket("192.168.0.216",222);
		//read from console
		InputStreamReader isr=new
InputStreamReader(System.in);
		BufferedReader br=new BufferedReader(isr);
		System.out.println("ENTER MSG FOR SERVER-");
		String cmg=br.readLine();

		//write to the socket
		OutputStream os=s.getOutputStream();
		PrintStream ps=new PrintStream(os);
		ps.println(cmg);

		//read from socket
	    InputStream os1 =s.getInputStream();
		InputStreamReader ir1 = new InputStreamReader(os1);
		BufferedReader br1 = new BufferedReader(ir1);
		String smg=br1.readLine();

		//write to the console
		System.out.println("message from the server is "+smg);
		os1.close();
		os.close();
		ps.close();
		ir1.close();
```

```
                br1.close();
                s.close();
        }
    }
```

Server.java-

```
    import java.io.BufferedReader;
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.InputStreamReader;
    import java.io.OutputStream;
    import java.io.PrintStream;
    import java.net.ServerSocket;
    import java.net.Socket;
    public class Server1
    {
        public static void main(String[] args) throws IOException
        {
            //to establish connection
            ServerSocket ss=new ServerSocket(222);
            Socket s=ss.accept();
            System.out.println("GOT CONNECTION");
            //to read data from socket
            InputStream os=s.getInputStream();
            InputStreamReader ir=new InputStreamReader(os);
            BufferedReader br=new BufferedReader(ir);
            String cmg=br.readLine();
            //write to the console
            System.out.println("MSG RECIEVED FROM CLIENT." +
cmg);
            //read from console
                InputStreamReader ir1 = new
InputStreamReader(System.in);
                BufferedReader br1 = new BufferedReader(ir1);
```

```
                System.out.println("enter the message for client");
                String smg=br1.readLine();

            //write to the socket
                OutputStream os1 =s.getOutputStream();
                PrintStream ps = new PrintStream(os1);
                ps.println(smg);
            ss.close();
            s.close();
            ir.close();
            br.close();
            os.close();
        }
    }
```

DAY12-

JDBC CONNECTION

Example-

```
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.sql.Statement;
    public class JdbcExample
    {
        public static void main(String[] args) throws
ClassNotFoundException, SQLException
        {
            //load the driver
            Class.forName("com.mysql.cj.jdbc.Driver");
            //get connection
            Connection con;
        con=DriverManager.getConnection("jdbc:mysql://localhost:3306
/dac_dbt_2024","root","DEEPSHIKHA");
```

```
                System.out.println("WELCOME TO JDBC CONNECTION
");

                Statement stm=con.createStatement();
                String query="select * from emp";
                ResultSet rs=stm.executeQuery(query);
                while(rs.next())
                {
                System.out.println(rs.getInt(1)+""+ rs.getString(2));
                }
                rs.close();
                stm.close();
            }
        }
```

.....