

1. Write a program that displays Welcome to Java, Learning Java Now and Programming is Fun.

Program/Solution:

```
public class Main{
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java, Learning Java Now and Programming is fun.");
    }
}
```

Output:

```
java -cp /tmp/N8d9ptqbnV/Main
Welcome to Java, Learning Java Now and Programming is fun.

=== Code Execution Successful ===
```

2. Write a java program to calculate simple interest.

Program/Solution:

```
public class Main {
    public static void main(String[] args) {
        SimpleInterest SI = new SimpleInterest(12, 13, 2);
        System.out.println("Calculated Interest:");
        System.out.println("Interest = " + SI.calculateInterest());
        System.out.println("*****");
    }
}

class SimpleInterest{
    double p,r,n; // p=prinipal interest, r=rate on amount, n=nuber of period
    SimpleInterest(){
        p=r=n=0;
    }
    SimpleInterest(double p, double r, double n)
    {
        this.p = p;
        this.r = r;
        this.n = n;
    }
}
```

```
public double calculateInterest(){  
    return (p*r*n)/100;  
}  
}
```

Output:

```
java -cp /tmp/jybBQd4bnS/Main  
Calculated Interest:  
Interest = 3.12  
*****  
  
=== Code Execution Successful ===
```

3. Write a java program to check leap year.

Program/Solution:

```
class Main{  
    public static void main(String[] args) {  
        LeapYear LY = new LeapYear(1990);  
        System.out.println("Leap Year?:");  
        System.out.println((LY.checkYear()?"It's LeapYear":"It's not a Leap Year"));  
    }  
}  
  
class LeapYear{  
    long year;  
    LeapYear(){  
        year = 2012;  
    }  
    LeapYear(long year)  
    {  
        this.year = year;  
    }  
    public boolean checkYear(){  
        return ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0);  
    }  
}
```

Output:

```
java -cp /tmp/1aSH0yp3Bu/Main  
Leap Year?:1990  
It's not a Leap Year  
  
=== Code Execution Successful ===
```

4. Write a java program to add two binary numbers.

Program/Solution:

```
import java.util.*; public  
class Add2binary{  
    public static void main(String[] args){ long  
        binary1, binary2;  
        int i=0, carry = 0;  
        int[] sum = new int[20];  
  
        Scanner in = new Scanner(System.in); System.out.print("Enter  
        first binary number: "); binary1 = in.nextLong();  
  
        System.out.print("Enter second binary number: "); binary2 =  
        in.nextLong();  
  
        in.close();  
  
        while(binary1 != 0 || binary2 != 0){  
  
            sum[i++] = (int)((binary1 % 10 + binary2 % 10 + carry) % 2); carry =  
            (int)((binary1 % 10 + binary2 % 10 + carry) / 2); binary1 = binary1 / 10;  
            binary2 = binary2 / 10;  
  
        }  
  
        if(carry != 0){  
  
            sum[i++] = carry;  
  
        }  
  
        --i;  
  
        System.out.print("Sum of two binary numbers: "); while(i >=  
        0)
```

```
        System.out.print(sum[i--]);  
        System.out.print("\n");  
    }  
}
```

Output:

```
java -cp /tmp/AzHEP602E9/Add2binary  
Enter first binary number: 1110  
Enter second binary number: 0111  
Sum of two binary numbers: 10101  
  
=== Code Execution Successful ===
```

1. Write a java program to check if a given number is Armstrong from given array.

Program/Solution:

```
public class Practical2_EquationSolver {
    public static void main(String[] args) {
        Armstrong am = new Armstrong(10);
        if (am.is_armstrong()) {
            System.out.println(am.number + " = Armstrong Number");
        }
        else{
            System.out.println(am.number + " = Not Armstrong Number");
        }
    }
}

class Armstrong {
    int number;
    String number_str;

    Armstrong() {
        number=0;
    }
    Armstrong(int number){
        this.number = number;
        this.number_str = String.valueOf(this.number);
    }
    boolean is_armstrong()
    {
        int power = this.number_str.length();
        int temp = this.number;
        int rem=0,arm_sum =0;
        while (temp > 0) {
            rem = temp % 10;
            arm_sum += Math.pow(rem, power);
            temp /= 10;
        }
        if (this.number == arm_sum) {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

Output:

```
java -cp /tmp/Ki5MQD1fKA/Practical2_EquationSolver
153 = Armstrong Number

=== Code Execution Successful ===
```

2. Write a java program to perform bubble sort on Strings.**Program/Solution:**

```
public class StrBubbleSort
{
    public static void main(String[] args)
    {
        String str[] = {"Hansil", "Alexnader", "Yuki", "Tony", "Peter", "Kane", "Steven", "Marc"};
        String temp;

        System.out.println("Strings in sorted order: ");
        for(int j = 0; j < str.length; j++)
        {
            for(int i = j+1; i < str.length; i++)
            {
                //Comparing adjacent strings

                if(str[i].compareTo(str[j]) < 0)
                {
                    temp = str[j]; str[j] = str[i]; str[i] = temp;
                }
            }
            System.out.println(str[j]);
        }
    }
}
```

Output:

```
java -cp /tmp/JaP3W2s5GP/StrBubbleSort
Strings in sorted order:
Alexnader
Hansil
Kane
Marc
Peter
Steven
Tony
Yuki

=== Code Execution Successful ===
```

3. Write a java program to perform different methods of string buffer class.

Program:

Program/Solution:

```
class StrBuffer{
    public void sbMethods(){
        StringBuffer sb = new StringBuffer(); sb.append("Hello");
        sb.append(" ");
        sb.append("world");
        String message = sb.toString(); System.out.println(message);
        System.out.println(sb.reverse());
    }
}

public class Test {
    public static void main(String[] args){
        StrBuffer s = new StrBuffer(); s.sbMethods();
    }
}
```

Output:

```
java -cp /tmp/an1GvjgS6M/Test
Hansil MG
GM lisnaH

=== Code Execution Successful ===
```

1. Create a class named Product with instance variables MRP and QUANTITY and methods display(), setdata(). In display() method, display the instance variables value (MRP and QUANTITY). And in setdata() method set the instance variable values (MRP and QUANTITY).

Program/Solution:

```
public class PR3_a{ private double mrp = 0;
private int quantity = 0;
    public void setData(double mrp , int quantity){
        this.mrp = mrp;
        this.quantity = quantity;
    }

    public void display(){
        System.out.println("MRP: "+ this.mrp);
        System.out.println("QUANTITY: " + this.quantity);
    }
    public static void main(String[] args){ PR3_a p1 = new PR3_a(); p1.setData(90, 100);
        p1.display();
    }
}
```

Output:

```
java -cp /tmp/rRKsqlRcvZ/PR3_a
MRP: 90.0
QUANTITY: 100

=== Code Execution Successful ===
```

2. Write a class named Account with instance variables Ac_No, Name and Balance and methods display(), setdata(), deposit(). In display() method display the instance variable values (Ac_No, Name and Balance). And in setdata() method set the instance variable values (Ac_No, Name and Balance) and in deposit() method the amount that the user want to deposit will be deposited.

Program/Solution:

```
public class PR3_b {
    private int acNo;
```



```
private String name;
private double balance;
public void display() {
    System.out.println("Account Number: " + acNo);
    System.out.println("Name: " + name);
    System.out.println("Balance: " + balance);
}
public void setData(int acNo, String name, double balance) {
    this.acNo = acNo;
    this.name = name;
    this.balance = balance;
}

public void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        System.out.println("Deposit successful. Updated balance: "
            + balance);
    } else {
        System.out.println("Invalid deposit amount. Please enter a positive amount.");
    }
}

public static void main(String[] args) {
    PR3_b account = new PR3_b();
    account.setData(123456, "John Doe", 1000.0);
    account.display();
    account.deposit(500.0);
    account.display();
}
```

Output:

```
java -cp /tmp/1fPKrEP925/PR3_b
Account Number: 123456
Name: John Doe
Balance: 1000.0
Deposit successful. Updated balance: 1500.0
Account Number: 123456
Name: John Doe
Balance: 1500.0

=== Code Execution Successful ===
```

3. Create a class named Student with static variable Enrollment no. and instance variable Name and methods display(), setdata(). In display() method, display the variables value (Enrollment no. and Name). And in setdata() method set the variable values (Enrollment no. and Name).

Program/Solution:

```
import java.util.Scanner;
public class Practical3_ {
    public static void main(String[] args)
    {
        Student s1 = new Student();
        s1.setData(11122345, "steve");
        s1.display();
    }
}

class Student{
    static long enr_no;
    String name;
    void display()
    {
        System.out.println("Name = " + this.name + "\nEnrollement = " + enr_no);
    }
    void setData(long enr_no, String name)
    {
        Student.enr_no = enr_no;
        this.name = name;
    }
}
```

Output:

```
java -cp /tmp/SNjBgD1uiD/Practical3_
Name = steve
Enrollement = 11122345

=== Code Execution Successful ===
```

1. Create a default constructor of class Product to print-“Welcome to Super-Market”. And Parameterized constructor to get the values of variables. (using Command Line Arguments)

Program/Solution:

```
public class Product {
    private String productName; private double price;
    public Product() {
        System.out.println("Welcome to Super-Market.");
    }

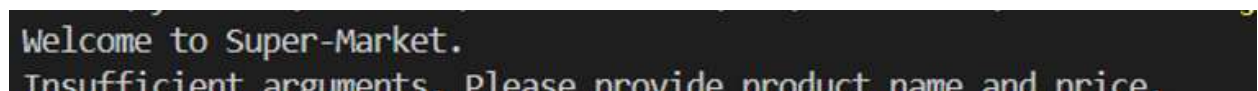
    // Parameterized constructor
    public Product(String productName, double price) { this.productName =
        productName;
        this.price = price;
    }

    // Method to display product details public void display() {
        System.out.println("Product Name: " + productName); System.out.println("Price:
        " + price);
    }

    // Main method to test the constructors public static void main(String[] args) {
        // Create an instance using default constructor Product defaultProduct = new
        Product();

        // Create an instance using parameterized constructor if (args.length >= 2) {
        String productName = args[0];
        double price = Double.parseDouble(args[1]);
        Product parameterizedProduct = new Product(productName,
        price);
        parameterizedProduct.display();
        }
    }
}
```

Output:



```
Welcome to Super-Market.
Insufficient arguments. Please provide product name and price.
```

2. Create a default constructor of class Account to print “Welcome to Bank”. And parameterized constructor to get the values of variables. (using command line arguments).

Program/Solution:

```
public class Account {
    private String name;
    private double amount;

    public Account() {
        System.out.println("Welcome to Bank");
    }

    public Account(String name, double amount) {
        this.name = name;

        this.amount = amount;
    }

    public void display() {
        System.out.println("Account Holder: " + name);
        System.out.println("Amount: " + amount);
    }

    public static void main(String[] args) {
        Account defaultAccount = new Account();

        if (args.length >= 2) {
            String name = args[0];
            double amount = Double.parseDouble(args[1]);
            Account parameterizedAccount = new Account(name, amount);
            parameterizedAccount.display();
        } else {
            System.out.println("Insufficient arguments. Please provide Account holder name and amount.");
        }
    }
}
```

Output:

```
java -cp /tmp/aWzjZDiBFX/Account
Welcome to Bank
Insufficient arguments. Please provide Account holder name and amount.

=== Code Execution Successful ===
```

3. Create a default constructor of class Student to print “Welcome to Student-Information system”. And parameterized constructor to get the value of variables (Enrollment no. and Name). (using Command Line Arguments)

Program/Solution:

```
public class Student {
    private String name;
    private long enrollment_no;

    public Student() {
        System.out.println("Welcome to Student-Information System");
    }

    public Student(String name, long enrollment_no) {
        this.name = name;
        this.enrollment_no = enrollment_no;
    }

    public void display() {
        System.out.println("Student Name : " + name);
        System.out.println("Enrollment No. : " + enrollment_no);
    }

    public static void main(String[] args) {
        Student defaultStudent = new Student();
        if (args.length >= 2) {
            String name = args[0];
            long enrollment_no = Long.parseLong(args[1]);
            Student parameterizedStudent = new Student(name,
                enrollment_no);
            parameterizedStudent.display();
        } else { System.out.println("Insufficient arguments. Please provide Account holder
name and amount.") } }
```

Output:

```
java -cp /tmp/D4IOvZc5mr/Student
Welcome to Student-Information System
Insufficient arguments. Please provide Account holder name and amount.

=== Code Execution Successful ===
```

1. **Create three inherited classes named-Crockery, Household, Food-items from Product super class and inherit the instance method (display() and setdata()) and variables (MRP and QUANTITY) of super class product**

Product.java

```
public class Product{
    protected double
    MRP; protected
    int QUANTITY;

    protected void display(){
        System.out.println("Price: " + MRP);
        System.out.println("Stock: " + QUANTITY);
    }
    protected void setdata(double MRP, int
    QUANTITY){ this.MRP = MRP;
    this.QUANTITY = QUANTITY;
    }
}
```

Crockery.java

```
public class Crockery extends Product{
    private String type_of_product = "Crockery";

    public void printType(){
        System.out.println("Type of Product: " + this.type_of_product);
    }
}
```

Food_items.java

```
public class Food_items extends Product{
```

```
private String type_of_product =  
    "Crockery";  
  
public void printType(){  
    System.out.println("Type of Product: " + this.type_of_product);  
}  
}
```

Household.java

```
public class Household extends Product{  
    private String type_of_product = "Household";  
  
    public void printType(){  
        System.out.println("Type of Product: " + this.type_of_product);  
    }  
}
```

Main.java

```
public class Main {  
    public static void main(String[]  
        args) { Crockery c = new  
        Crockery(); Household h = new  
        Household(); Food_items fi =  
        new Food_items();  
        c.printType();  
        c.setdata(999.99,  
        301); c.display();  
        h.printType();  
        h.setdata(89.99,  
        11); h.display();  
        fi.printType();  
    }  
}
```

```

        fi.setdata(90.92,
        78); fi.display();
    }
}

```

Output:

```

PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 5\1> javac Main.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 5\1> java Main
Type of Product: Crockery
Price: 999.99
Stock: 301
Type of Product: Household
Price: 89.99
Stock: 11
Type of Product: Crockery
Price: 90.92
Stock: 78

```

2. **Create two inherited classes named-Savings, Current from Account super class and inherit the instance method(display(), setdata() and deposit()) and variables(Ac No, Name and Balance) of super class Account.**

Account.java

```

class Account{
    protected long
    ac_no; protected
    String name;
    protected int
    balance;
    protected void display(){
        System.out.println("\nCustomer Name: " + this.name);
        System.out.println("Account Number: " + this.ac_no);
        System.out.println("Account Balance: " + this.balance);
    }

    protected void setdata(long ac_no, String name, int balance){

```



```
        this.ac_no = ac_no;
        this.name = name;
        this.balance =
        balance;
    }

    protected void deposit(long ac_no, int amount){
        if(amount > 0){
            System.out.println("\n\nCustomer Name: " + this.name);
            System.out.println("Account Number: " + this.ac_no);
            System.out.println("Account Balance after deposit: " +
            (this.balance + amount));
        }
        else
            System.out.println("Entered amount is negative!!!!");
    }
}
```

Current.java

```
public class Current extends Account{
    public String acc_type = "Current Account";
}
```

Savings.java

```
public class Savings extends Account{
    public String acc_type = "Savings Account";
}
```

Main.java

```
public class Main{
    public static void main(String[] args)
    {
```

```

        Savings S = new Savings();
        S.setdata(202019754929001L, "Steve", 32000);

        S.display();

        S.deposit(202019754929L,
        200);

        System.out.println("Account type: " + S.acc_type);
        Current C = new Current();

        C.setdata(202019754929000L, "Stark", 100020);

        C.display();

        C.deposit(202019754929000L,
        200);

        System.out.println("Account type: " + C.acc_type);
    }
}

```

Output:

```

PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 5\2> javac Main.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 5\2> java Main

Customer Name: Steve
Account Number: 202019754929001
Account Balance: 32000

Customer Name: Steve
Account Number: 202019754929001
Account Balance after deposit: 32200
Account type: Savings Account

Customer Name: Stark
Account Number: 202019754929000
Account Balance: 100020

Customer Name: Stark
Account Number: 202019754929000
Account Balance after deposit: 100220
Account type: Current Account

```

3. Create two inherited classes named- BE, ME from Student super class and

inherit the instance method (display() and setdata()) and variables (Enrollment no and Name) of super class Student and make an instance variable Entry_year and final variable duration for each of the above class.

Student.java

```
public class
    Student{ long
        enroll_no;
        String name;

        void display()
        {
            System.out.println("Enrollment number: " + this.enroll_no);
            System.out.println("Name: " + name);
        }

        void setData(long enroll_no, String name)
        {
            this.enroll_no = enroll_no;
            this.name = name;
        }
    }
```

BE.java

```
public class BE extends
    Student{ int entry_year;
        final int duration=4;

        BE(int entry_year){
            this.entry_year =
```

```
        entry_year;
    }
}
ME.java
public class ME extends
    Student{ int entry_year;
    final int duration=3;

    ME(int entry_year){
        this.entry_year =
            entry_year;
    }
}
```

Main.java

```
public class Main {
    public static void main(String[]
        args){ BE student_1 = new
        BE(2020);
        student_1.setData(200283116032L, "Alex");
        student_1.display();
        System.out.println("Entry year: " +
        student_1.entry_year); ME student_2 = new ME(2023);
        student_2.setData(230283123032L, "Rogers");
        student_2.display();
        System.out.println("Entry year: " + student_2.entry_year);
    }
}
```

Output:

```
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 5\3> javac Main.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 5\3> java Main
Enrollment number: 200283116032
Name: Alex
Entry year: 2020
Enrollment number: 230283123032
Name: Rogers
Entry year: 2023
```

1. Create an interface named place with method search() and variable BlockNo and implement it together on all of the above classes. Use binary I/O.

Pr_1.java

```
interface Place{
    int blockNO =
    10; public void
    search();
}

class College implements
    Place{ public void
    search(){
        System.out.println("its behind annexe building!!");
    }
}

public class Pr_1{

    public static void main(String[] args)
    {
        College c = new College();
        System.out.println("What is the block no ?\n"+ c.blockNO);
        c.search();
    }
}
```

Output:

```
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 6> javac Pr_1.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 6> java Pr_1
What is the block no ?
10
its behind annexe building!!
```

2. Create an interface named Branch with method search() and variable IFSC Code and implement it together on all the above classes.

Pr_2.java

```
interface Branch{
    int ifscCode =
    10; public void
    search();
}

class Bank implements
    Branch{ public void
    search(){
        System.out.println("Naroda Branch, Ahmedabad");
    }
}

public class Pr_2{
    public static void main(String[] args)
    {
        Bank c = new Bank();
        System.out.println("Where is the Branch located ?\n"+
c.ifscCode);
        c.search();
    }
}
```

Output:

```
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 6> javac Pr_2.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 6> java Pr_2
Where is the Branch located ?
10
Naroda Branch, Ahmedabad
```

3. Create a interface named Result with method getMarks() and variable percentage and implement it together on all the above classes in getMarks() get the marks of 3 subjects and calculate the average inside the method.

Pr_3.java

```
interface Result{
    public void getMarks(int a,int b,int c);
}

class Results implements Result{
    public void getMarks(int a,int b,int
        c){ int total = a+b+c;
        float per = total/3;
        System.out.println("You got "+ per
            +"%");
    }
}

public class Pr_3{
    public static void main(String[] args)
    {
        Results c = new Results();
        c.getMarks(92,81,71);
    }
}
```

Output:

```
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 6> javac Pr_3.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 6> java Pr_3
You got 81.0%
```


1. Define three objects for all the classes named-Crockery, Household, Food- items and store the initial values for all the objects in arraylist or collection.

Example: Class Crockery extends Product, implements place; Define objects of crockery -plates, cups, jug;

Pr_1.java

```
import
java.util.*;
class Pr_1
{
    public static void main(String[] args)
    {
        ArrayList<Crockery> crock=new
        ArrayList<Crockery>(); Crockery cro1=new
        Crockery();
        Crockery cro2=new
        Crockery(); Crockery
        cro3=new Crockery();
        cro1.plates();
        cro1.details(1000,30);
        cro2.cups();
        cro2.details(400,2);
        cro3.jug();
        cro3.details(2000,4);
        crock.add(cro1);
        crock.add(cro2);
        crock.add(cro3);
        Iterator itr=crock.iterator();
        while(itr.hasNext())
```

```
{

    Crockery ck=(Crockery)itr.next();

    System.out.println("Price:"+ck.price+ " Quantity:
"+ck.quantity);

}

ArrayList<Household> house=new ArrayList<Household>();

Household ho1=new Household();

Household ho2=new
Household(); Household
ho3=new Household();

ho1.spoons();
ho1.details(400,12);
ho2.knives();
ho2.details(500,2);
ho3.stoves();
ho3.details(10000,2);

house.add(ho1);
house.add(ho2);
house.add(ho3);

Iterator i=house.iterator();
while(i.hasNext())
{

    Household ho=(Household)i.next();

    System.out.println("Price:"+ho.price+ " Quantity:
"+ho.quantity);

}

ArrayList<Fooditems> food=new
ArrayList<Fooditems>(); Fooditems fo1=new
```

```

        Fooditems();
        Fooditems fo2=new
        Fooditems(); Fooditems
        fo3=new Fooditems();
        fo1.butter();
        fo1.details(70,1);
        fo2.sunfloweroil();
        fo2.details(2300,1);
        fo3.bread();
        fo3.details(100,4);
        food.add(fo1);

        food.add(f
        o2);
        food.add(f
        o3);
        Iterator ir=food.iterator();
        while(ir.hasNext())
        {

            Fooditems it=(Fooditems)ir.next();
            System.out.println("Price:"+it.price+ " Quantity:
            "+it.quantity);
        }
    }

    class Product
    {
        float
        price;

```

```
        int
        quantity
    ;
    void details( float price, int quantity)
    {
        this.price=price;
        this.quantity=quantity;
    }
}
class Crockery extends Product
{
    void plates()
    {
        System.out.println("Plates");
    }
    void cups()
    {
        System.out.println("Cups");
    }
    void jug()
    {
        System.out.println("Jug");
    }
}
class Household extends Product
{
    void spoons()
    {
```

```
        System.out.println("Spoons");
    }
    void knives()
    {
        System.out.println("Knives");
    }
    void stoves()
    {
        System.out.println("Stoves");
    }
}
class Fooditems extends
    Product { void butter()
    {
        System.out.println("Butter");
    }
    void sunfloweroil()
    {
        System.out.println("Sunfloweroil");
    }
    void bread()
    {
        System.out.println("Bread");
    }
}
```

Output:

```

PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 7> javac Pr_1.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 7> java Pr_1
Plates
Cups
Jug
Price:1000.0 Quantity: 30
Price:400.0 Quantity: 2
Price:2000.0 Quantity: 4
Spoons
Knives
Stoves
Price:400.0 Quantity: 12
Price:500.0 Quantity: 2
Price:10000.0 Quantity: 2
Butter
Sunfloweroil
Bread
Price:70.0 Quantity: 1
Price:2300.0 Quantity: 1
Price:100.0 Quantity: 4

```

2. Define three objects for all the classes named-Savings, Current and store the initial values for all the objects in arraylist or collection.

Example: Class Savings extends Account implements Branch; Define objects.

Pr_2.java

```

import java.util.*;

public class Pr_2{

    public static void main(String[] args){

        ArrayList<Saving> sav=new
        ArrayList<Saving>(); Saving a1=new
        Saving();
        a1.details(1654354,"Moksha",866.22f,556f,8.7
        f); sav.add(a1);
        Iterator itr=sav.iterator();
        while(itr.hasNext()){

```

```

        Saving sa=(Saving)itr.next();

        System.out.println("\nINFORMATION OF SAVING ACCOUNT");

        System.out.println("\nACCOUNT NUMBER:"+sa.acno+
"\nNAME OF ACCOUNT HOLDER: "+sa.name+"\nBALANCE
:"+sa.balance+"\nDEPOSITE : " +
sa.deposite + "\nINTEREST:"+sa.interest);
    }

    a1.Deposite()

;

a1.interestra
te();

ArrayList<Current> cur=new
ArrayList<Current>(); Current a2=new
Current();

a2.details(154534,"XYZ",874.12f,964f,6.6f);

cur.add(a2);

Iterator it=cur.iterator();

while(it.hasNext()){

    Current cu=(Current)it.next();

    System.out.println("\nINFORMATION OF CURRENT ACCOUNT");

    System.out.println("\nACCOUNT NUMBER:"+cu.acno+
"\nNAME OF ACCOUNT HOLDER: "+cu.name+"\nBALANCE
:"+cu.balance+"\nDEPOSITE : " +
cu.deposite + "\nINTEREST:" + cu.interest);

}

a2.Deposite()

;

a2.interestra
te();

}

```

```
}

class Account{
    void details(int acno,String name,float balance,float deposit,float
interest){
        this.acno=acno;
        this.name=name;
        this.balance=balance
        ;
        this.deposit=deposi
te;
        this.interest=intere
st;
    }
}

class Saving extends
Account{ void
Deposit(){
    balance= balance+deposit;
    System.out.println("Balance after
deposit:"+balance);
}

void interestrate(){
    interest=(balance*interest)/percent;
    System.out.println("Amount of
interest:"+interest);}
}

class Current extends Account{
void Deposit(){
    balance= balance+deposit;
```



```

        System.out.println("Balance after
        deposit:"+balance);
    }
    void interestrate(){
        interest=(balance*interest)/percent;
        System.out.println("Amount of
        interest:"+interest);}

```

Output:

```

PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 7> javac Pr_2.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 7> java Pr_2

INFORMATION OF SAVING ACCOUNT

ACCOUNT NUMBER:1654354
NAME OF ACCOUNT HOLDER: Moksha
BALANCE :866.22
DEPOSIT :556.0
INTEREST:8.7
Balance after deposit:1422.22
Amount of interest:123.73313

INFORMATION OF CURRENT ACCOUNT

ACCOUNT NUMBER:154534
NAME OF ACCOUNT HOLDER: XYZ
BALANCE :874.12
DEPOSIT :964.0
INTEREST:6.6
Balance after deposit:1838.12
Amount of interest:121.31592

```

3. Define two objects for all the classes named BE, ME and store the initial values for all the objects in arraylist or collection.
Example: Class BE extends Student, implements Result; Define objects of BE.

Pr_3.java

```

import java.util.*;

public class Pr_3
{
    public static void main(String[] args)
    {

```

```
ArrayList<BE> ba=new
    ArrayList<BE>(); BE be1=new
    BE();
    be1.course();
    be1.details("Steve",230283116033L,"IT");
    ba.add(be1);
    Iterator itr = ba.iterator();
    while(itr.hasNext())
    {
        BE be2=(BE)itr.next();
        System.out.println("NAME :"+be2.name+"\nENROLLMENT
NUMBER:"+be2.ennno+"\nBRANCH:"+be2.branch);
    }
    ArrayList<ME> ma=new
    ArrayList<ME>(); ME me1=new ME();
    me1.course();
    me1.details("Tony",230280110034L,"CS");
    ma.add(me1);
    Iterator it = ma.iterator();
    while(it.hasNext())
    {
        ME me2=(ME)it.next();
        System.out.println("NAME :"+me2.name+"\nENROLLMENT
NUMBER:"+me2.ennno+"\nBRANCH:"+me2.branch);
    }
}

class Student
{
```

```
void details(String name,long enno,String branch)
{
    this.name=name;
    this.enno=enno;
    this.branch=branch;
}
}
class BE extends Student
{
    void course(){
        System.out.println("\nBE STUDENT INFORMATION");
    }
}
class ME extends Student{
    void course()
    {
        System.out.println("\nME STUDENT INFORMATION");
    }
}
```

Output:

```
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 7> javac Pr_3.java
PS G:\My Drive\BE Sem 4\LD Practicals\OOP\Practicals\Practical 7> java Pr_3

BE STUDENT INFORMATION
NAME :Steve
ENROLLMENT NUMBER:230283116033
BRANCH:IT

ME STUDENT INFORMATION
NAME :Tony
ENROLLMENT NUMBER:230280110034
BRANCH:CS
```

1. Create a Package and put all the classes mentioned above in package.**Current.java**

```
package Bank;

public class
    Current{ public
        void display(){
            System.out.println("Current Account\nBank: HDFC\nBranch: Surat");
        }
    }
```

Savings.java

```
package Bank;

public class
    Savings{ public
        void display(){
            System.out.println("Savings Account\nBank:
            AXIS\nBranch: Ahmedabad");
        }
    }
```

Practical8b.java

```
//File:
Practical18b.java

import Bank.*;

import
java.util.Scanner;

class Practical8b
{
```

```
public static void main(String args[])
{
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter 1 for
    Savings"); System.out.println("Enter
    2 for Current");
    System.out.print("Enter your choice:
    "); int ch = sc.nextInt();
    System.out.print("\n");
    switch(ch)
    {
        case 1:
            Bank.Savings cr=new Bank.Savings();
            cr.display(); case 2:
            Bank.Current hr=new Bank.Current();
            hr.display();
            break;
            default
            :
            System.out.println("Invalid Choice");
        }
    }
}
```

Output:

```
C:\JAVA\BE_PRACTICALS>javac Practical8b.java

C:\JAVA\BE_PRACTICALS>java Practical8b
Enter 1 for Savings
Enter 2 for Current
Enter your choice: 1

Savings Account
Bank: AXIS
Branch: Ahmedabad

C:\JAVA\BE_PRACTICALS>|
```

1. Create a method named buy () in the main function performing exception handling.

Example: Banana = 10; Banana_Buy = 12

Here, Banana_Buy > Banana (throw exception).

Program/Solution:

```
public class Main {
    private int availableQuantity;

    public Main(int availableQuantity) {
        this.availableQuantity = availableQuantity;
    }

    // Method to perform buying with exception handling
    public void buy(int quantityToBuy) {
        try {
            if (quantityToBuy > availableQuantity) {
                throw new IllegalArgumentException("Quantity to buy exceeds
available quantity");
            } else {
                // Perform the buying process here
                availableQuantity -= quantityToBuy;
                System.out.println("Purchase successful! Remaining quantity:
" + availableQuantity);
            }
        } catch (IllegalArgumentException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        Main item = new Main(10); // Assuming there are initially 10 items
        available
        int quantityToBuy = 12; // Quantity user wants to buy

        // Attempt to buy the specified quantity
        item.buy(quantityToBuy);
    }
}
```

Output:

PROBLEMS TERMINAL OUTPUT PORTS DEBUG CONSOLE

```
PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS> cd "e:\MH-SEM-4-DEGREE(BE)\OOP-1\F
ava } ; if ($?) { java Main }
Exception caught: Quantity to buy exceeds available quantity
```

2. Create a method named Withdraw () in the main function performing exception handling.

Example: Balance = 1000; Withdraw = 12000

Here, Balance<Withdraw (throw exception)

Program/Solution:

```
public class Main {
    private double balance;

    public Main(double balance) {
        this.balance = balance;
    }

    // Method to perform withdrawal with exception handling
    public void withdraw(double withdrawalAmount) {
        try {
            if (withdrawalAmount > balance) {
                throw new IllegalArgumentException("Withdrawal amount exceeds
available balance");
            } else {
                // Perform the withdrawal process here
                balance -= withdrawalAmount;
                System.out.println("Withdrawal successful! Remaining balance:
" + balance);
            }
        } catch (IllegalArgumentException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        Main account = new Main(1000); // Assuming there is initially $1000
        balance
        double withdrawalAmount = 12000; // Amount user wants to withdraw

        // Attempt to withdraw the specified amount
        account.withdraw(withdrawalAmount);
    }
}
```


Output:

```
PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS> cd "e:\MH-SEM-4-DEGREE(BE)\OOP-1\PR
ava } ; if ($?) { java Main }
Exception caught: Withdrawal amount exceeds available balance
```

3. Create a method named `searchStudent()` in the `main ()` Function performing exception handling.

Example: if we search the student name and if it is present in the list then it will represent the details else it will throw an exception.

Program/Solution:

```
import java.util.HashMap;

public class Main {
    private HashMap<String, String> studentDetails;

    public Main() {
        // Initialize student details hashmap
        studentDetails = new HashMap<>();
        // Populate with sample data (student name as key, details as value)
        studentDetails.put("John", "Age: 20, Grade: A");
        studentDetails.put("Alice", "Age: 22, Grade: B");
        studentDetails.put("Bob", "Age: 21, Grade: A-");
    }

    // Method to search for student details with exception handling
    public void searchStudent(String studentName) {
        try {
            String details = studentDetails.get(studentName);
            if (details == null) {
                throw new IllegalArgumentException("Student not found");
            } else {
                System.out.println("Student details for " + studentName + ":
" + details);
            }
        } catch (IllegalArgumentException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
```

```
Main mainObj = new Main(); // Create an instance of Main class
String studentName = "Alice"; // Name of the student to search

// Attempt to search for the student details
mainObj.searchStudent(studentName);
    }
}
```

Output:

```
Exception caught: Withdrawal amount exceeds available balance
PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS> cd "e:\MH-SEM-4-DEGREE(BE)\OO
ava } ; if ($?) { java Main }
Student details for Alice: Age: 22, Grade: B
```

1. Save object data in the file using File Writer class, using parameterized constructor.

Program/Solution:

```
import java.io.*;

// Serializable class representing student details
class Student implements Serializable {
    private String name;
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getters and setters
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Age: " + age;
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a Student object
        Student student = new Student("John", 20);
    }
}
```

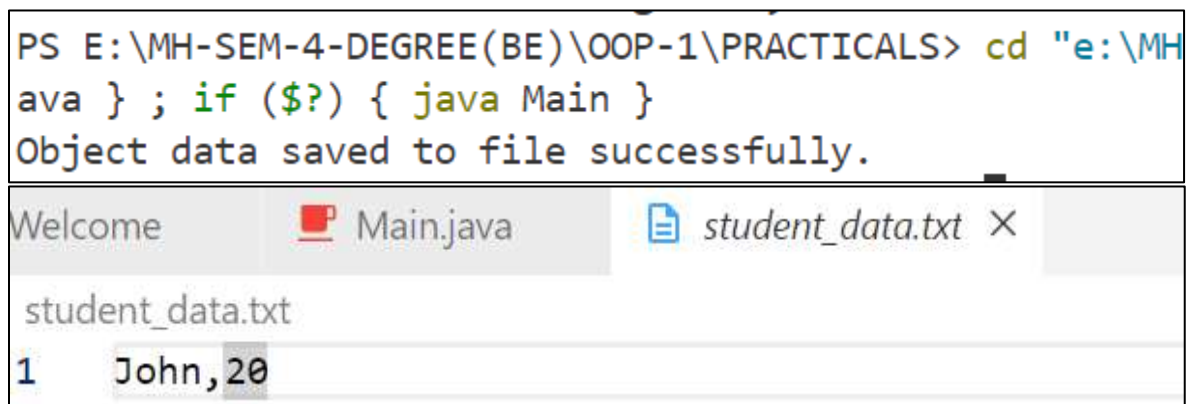
```
// File path
String filePath = "student_data.txt";

try {
    // Create ObjectOutputStream to write object to file
    ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(filePath));

    // Write the object to the file
    outputStream.writeObject(student);

    // Close the stream
    outputStream.close();

    System.out.println("Object data saved to file successfully.");
} catch (IOException e) {
    System.out.println("Error occurred: " + e.getMessage());
}
}
```

Output:

The screenshot shows a Windows command prompt window with the following text:

```
PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS> cd "e:\MH
ava } ; if ($?) { java Main }
Object data saved to file successfully.
```

Below the command prompt is a Java IDE window titled "Welcome" with tabs for "Main.java" and "student_data.txt". The "student_data.txt" tab is active, showing the following text:

```
1 John,20
```

1. Create customer class which extends thread class and contains two instance variables name, ProductName and static variable Product Quantity. For example if two customers are trying to buy the same product at once then follow

Program/Solution:

```
public class Customer extends Thread {
    private String name;
    private String productName;
    private static int productQuantity = 10; // Initial quantity of the
    product

    public Customer(String name, String productName) {
        this.name = name;
        this.productName = productName;
    }

    @Override
    public void run() {
        synchronized (Customer.class) { // Synchronize on the class object to
            ensure atomic access to shared variable
                if (productQuantity > 0) { // Check if product is available
                    System.out.println(name + " is buying " + productName);
                    productQuantity--; // Decrease product quantity
                    System.out.println(productName + " purchased by " + name + ".
                    Remaining quantity: " + productQuantity);
                } else {
                    System.out.println("Sorry, " + productName + " is out of
                    stock for " + name);
                }
            }
        }

    public static void main(String[] args) {
        // Create two customers trying to buy the same product
        Customer customer1 = new Customer("Customer1", "ProductA");
        Customer customer2 = new Customer("Customer2", "ProductA");

        // Start the threads
        customer1.start();
        customer2.start();
    }
}
```

Output:

```
PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS\ChatGPTPrg> cd ..\e:\MH-SEM-4-DEGREE(BE)\OO
omer.java } ; if ($?) { java Customer }
Customer1 is buying ProductA
ProductA purchased by Customer1. Remaining quantity: 9
Customer2 is buying ProductA
ProductA purchased by Customer2. Remaining quantity: 8
```

2. Create Customer class which extends thread class and contains two instance variables name, BankName and static variable Product Balance. if two customers are trying to Withdraw from the same Account at once then then follow synchronization of two customers extending thread class.

Program/Solution:

```
public class Customer extends Thread {
    private String name;
    private String bankName;
    private static double accountBalance = 1000.0; // Initial balance in the
    account

    public Customer(String name, String bankName) {
        this.name = name;
        this.bankName = bankName;
    }

    @Override
    public void run() {
        synchronized (Customer.class) { // Synchronize on the class object to
        ensure atomic access to shared variable
            double withdrawAmount = 500.0; // Amount to withdraw
            if (withdrawAmount > accountBalance) {
                System.out.println("Sorry, " + bankName + " has insufficient
                balance for " + name + " to withdraw.");
            } else {
                System.out.println(name + " is withdrawing " + withdrawAmount
                + " from " + bankName);
                accountBalance -= withdrawAmount; // Decrease account balance
                System.out.println("Withdrawal successful! New balance for "
                + bankName + ": " + accountBalance);
            }
        }
    }

    public static void main(String[] args) {
        // Create two customers trying to withdraw from the same bank account
        Customer customer1 = new Customer("Customer1", "BankA");
```

```

        Customer customer2 = new Customer("Customer2", "BankA");

        // Start the threads
        customer1.start();
        customer2.start();
    }
}

```

Output:

```

PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS\ChatGPTPrg> cd "e:
omer.java } ; if ($?) { java Customer }
Customer1 is withdrawing 500.0 from BankA
Withdrawal successful! New balance for BankA: 500.0
Customer2 is withdrawing 500.0 from BankA
Withdrawal successful! New balance for BankA: 0.0

```

3. Create Number class which extends thread class and create two objects a1, a2 displaying the number having even Enrollment no. and odd Enrollment no.

Program/Solution:

```

public class Number extends Thread {
    private int enrollmentNo;

    public Number(int enrollmentNo) {
        this.enrollmentNo = enrollmentNo;
    }

    @Override
    public void run() {
        if (enrollmentNo % 2 == 0) {
            // Display even numbers
            for (int i = 2; i <= 10; i += 2) {
                System.out.println("Even Number: " + i);
                try {
                    Thread.sleep(1000); // Sleep for 1 second
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        } else {
            // Display odd numbers
            for (int i = 1; i <= 10; i += 2) {

```

```

        System.out.println("Odd Number: " + i);
        try {
            Thread.sleep(1000); // Sleep for 1 second
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

public static void main(String[] args) {
    // Create two Number objects with even and odd enrollment numbers
    Number a1 = new Number(1001); // Odd enrollment number
    Number a2 = new Number(1002); // Even enrollment number

    // Start the threads
    a1.start();
    a2.start();
}
}

```

Output:

```

PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS\ChatGPTPr> cd "e:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS\ChatGPTPr" & java er.java } ; if ($?) { java Number }
Odd Number: 1
Even Number: 2
Odd Number: 3
Even Number: 4
Even Number: 6
Odd Number: 5
Odd Number: 7
Even Number: 8
Odd Number: 9
Even Number: 10
PS E:\MH-SEM-4-DEGREE(BE)\OOP-1\PRACTICALS\ChatGPTPr> █

```


1. Create the above system using a menu and implement it using switch statement. For example, in the above system implement the menu driven like.
 1. For setting a value of MRP and QUANTITY for the product.
 2. For buying a product.
 3. For searching a product.
 4. To delete a product from storage.
 5. To show stored object data

Program/Solution:

```
import java.util.Scanner;

class Product {
    private String name;
    private double price;
    private int quantity;

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    // Getters and setters
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
```

```
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Price: " + price + ", Quantity: " +
quantity;
    }
}

public class ProductManagementSystem {
    private static Product product;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Set MRP and Quantity for the product");
            System.out.println("2. Buy a product");
            System.out.println("3. Search for a product");
            System.out.println("4. Delete a product from storage");
            System.out.println("5. Show stored object data");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    setProductDetails(scanner);
                    break;
                case 2:
                    buyProduct(scanner);
                    break;
                case 3:
                    searchProduct(scanner);
                    break;
                case 4:
                    deleteProduct();
                    break;
                case 5:
                    showStoredData();
                    break;
                case 6:

```

```
        System.out.println("Exiting...");
        break;
    default:
        System.out.println("Invalid choice. Please try again.");
    }
} while (choice != 6);
}
```

```
private static void setProductDetails(Scanner scanner) {
    System.out.print("Enter product name: ");
    String name = scanner.next();
    System.out.print("Enter product price: ");
    double price = scanner.nextDouble();
    System.out.print("Enter product quantity: ");
    int quantity = scanner.nextInt();
    product = new Product(name, price, quantity);
    System.out.println("Product details set successfully.");
}
```

```
private static void buyProduct(Scanner scanner) {
    if (product == null) {
        System.out.println("Please set product details first.");
        return;
    }
    System.out.print("Enter quantity to buy: ");
    int quantityToBuy = scanner.nextInt();
    if (quantityToBuy > product.getQuantity()) {
        System.out.println("Insufficient quantity available.");
    } else {
        product.setQuantity(product.getQuantity() - quantityToBuy);
        System.out.println("Product bought successfully.");
    }
}
```

```
private static void searchProduct(Scanner scanner) {
    if (product == null) {
        System.out.println("Please set product details first.");
        return;
    }
    System.out.print("Enter product name to search: ");
    String name = scanner.next();
    if (name.equals(product.getName())) {
        System.out.println("Product found: " + product);
    } else {
        System.out.println("Product not found.");
    }
}
```

```
    }  
}  
  
private static void deleteProduct() {  
    product = null;  
    System.out.println("Product deleted from storage.");  
}  
  
private static void showStoredData() {  
    if (product == null) {  
        System.out.println("No product stored.");  
    } else {  
        System.out.println("Stored product data: " + product);  
    }  
}  
}
```

Output:

Menu:

1. Set MRP and Quantity for the product
2. Buy a product
3. Search for a product
4. Delete a product from storage
5. Show stored object data
6. Exit

Enter your choice: 1

Enter product name: KoreeanNoodles

Enter product price: 100

Enter product quantity: 3

Product details set successfully.

Menu:

1. Set MRP and Quantity for the product
2. Buy a product
3. Search for a product
4. Delete a product from storage
5. Show stored object data
6. Exit

Enter your choice: 5

Stored product data: Name: KoreeanNoodles, Price: 100.0, Quantity: 3

2. Create the above system using a menu and implement it using switch statement. For example, in the above system implement the menu driven like.
 1. For Setting a value of Ac_No, Name and Balance for the Bank
 2. For Withdraw from a Account.
 3. For Searching a Account.
 4. To delete an Account from Bank.
 5. To show stored object data

Program/Solution:

```
import java.util.*;

class BankAccount {
    private int accountNumber;
    private String name;
    private double balance;

    public BankAccount(int accountNumber, String name, double balance) {
        this.accountNumber = accountNumber;
        this.name = name;
        this.balance = balance;
    }

    // Getters and setters
    public int getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(int accountNumber) {
        this.accountNumber = accountNumber;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }
}
```

```
    }

    @Override
    public String toString() {
        return "Account Number: " + accountNumber + ", Name: " + name + ",
Balance: " + balance;
    }
}

public class BankManagementSystem {
    private static List<BankAccount> accounts = new ArrayList<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Set Account Number, Name, and Balance");
            System.out.println("2. Withdraw from an Account");
            System.out.println("3. Search for an Account");
            System.out.println("4. Delete an Account from Bank");
            System.out.println("5. Show stored object data");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    setAccountDetails(scanner);
                    break;
                case 2:
                    withdrawFromAccount(scanner);
                    break;
                case 3:
                    searchAccount(scanner);
                    break;
                case 4:
                    deleteAccount(scanner);
                    break;
                case 5:
                    showStoredData();
                    break;
                case 6:
                    System.out.println("Exiting...");
            }
        }
    }
}
```

```
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    } while (choice != 6);
}

private static void setAccountDetails(Scanner scanner) {
    System.out.print("Enter account number: ");
    int accountNumber = scanner.nextInt();
    System.out.print("Enter name: ");
    String name = scanner.next();
    System.out.print("Enter balance: ");
    double balance = scanner.nextDouble();
    accounts.add(new BankAccount(accountNumber, name, balance));
    System.out.println("Account details set successfully.");
}

private static void withdrawFromAccount(Scanner scanner) {
    System.out.print("Enter account number: ");
    int accountNumber = scanner.nextInt();
    BankAccount account = findAccount(accountNumber);
    if (account != null) {
        System.out.print("Enter amount to withdraw: ");
        double amount = scanner.nextDouble();
        if (amount <= account.getBalance()) {
            account.setBalance(account.getBalance() - amount);
            System.out.println("Withdrawal successful! New balance: " +
account.getBalance());
        } else {
            System.out.println("Insufficient balance.");
        }
    } else {
        System.out.println("Account not found.");
    }
}

private static void searchAccount(Scanner scanner) {
    System.out.print("Enter account number to search: ");
    int accountNumber = scanner.nextInt();
    BankAccount account = findAccount(accountNumber);
    if (account != null) {
        System.out.println("Account found: " + account);
    } else {
        System.out.println("Account not found.");
    }
}
```

```
    }  
}  
  
private static void deleteAccount(Scanner scanner) {  
    System.out.print("Enter account number to delete: ");  
    int accountNumber = scanner.nextInt();  
    BankAccount account = findAccount(accountNumber);  
    if (account != null) {  
        accounts.remove(account);  
        System.out.println("Account deleted successfully.");  
    } else {  
        System.out.println("Account not found.");  
    }  
}  
  
private static BankAccount findAccount(int accountNumber) {  
    for (BankAccount account : accounts) {  
        if (account.getAccountNumber() == accountNumber) {  
            return account;  
        }  
    }  
    return null;  
}  
  
private static void showStoredData() {  
    if (accounts.isEmpty()) {  
        System.out.println("No accounts stored.");  
    } else {  
        System.out.println("Stored account data:");  
        for (BankAccount account : accounts) {  
            System.out.println(account);  
        }  
    }  
}  
}
```

Output:


```
Menu:
1. Set Account Number, Name, and Balance
2. Withdraw from an Account
3. Search for an Account
4. Delete an Account from Bank
5. Show stored object data
6. Exit
Enter your choice: 4
Enter account number to delete: 0
Account not found.

Menu:
1. Set Account Number, Name, and Balance
2. Withdraw from an Account
3. Search for an Account
4. Delete an Account from Bank
5. Show stored object data
6. Exit
Enter your choice: 5
Stored account data:
Account Number: 12345, Name: Hansil, Balance: 1.23456789E12
```

3. Create the above system using a menu and implement it using switch statement. For example, in the above system implement the menu driven like.
1. Add Student details.
 2. Update student Details.
 3. For Searching a Student.
 4. Delete Student Details.

Program/solution:

```
import java.util.*;

class Student {
    private int rollNumber;
    private String name;
    private int age;

    public Student(int rollNumber, String name, int age) {
        this.rollNumber = rollNumber;
        this.name = name;
    }
}
```

```

        this.age = age;
    }

    // Getters and setters
    public int getRollNumber() {
        return rollNumber;
    }

    public void setRollNumber(int rollNumber) {
        this.rollNumber = rollNumber;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Roll Number: " + rollNumber + ", Name: " + name + ", Age: " +
age;
    }
}

public class StudentManagementSystem {
    private static List<Student> students = new ArrayList<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add Student details");

```

```
System.out.println("2. Update Student details");
System.out.println("3. Search for a Student");
System.out.println("4. Delete Student details");
System.out.println("5. Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();

switch (choice) {
    case 1:
        addStudentDetails(scanner);
        break;
    case 2:
        updateStudentDetails(scanner);
        break;
    case 3:
        searchStudent(scanner);
        break;
    case 4:
        deleteStudentDetails(scanner);
        break;
    case 5:
        System.out.println("Exiting...");
        break;
    default:
        System.out.println("Invalid choice. Please try again.");
}
} while (choice != 5);
}

private static void addStudentDetails(Scanner scanner) {
    System.out.print("Enter Roll Number: ");
    int rollNumber = scanner.nextInt();
    System.out.print("Enter Name: ");
    String name = scanner.next();
    System.out.print("Enter Age: ");
    int age = scanner.nextInt();
    students.add(new Student(rollNumber, name, age));
    System.out.println("Student details added successfully.");
}

private static void updateStudentDetails(Scanner scanner) {
    System.out.print("Enter Roll Number to update: ");
    int rollNumber = scanner.nextInt();
    Student student = findStudent(rollNumber);
    if (student != null) {
```

```
        System.out.println("Enter updated details:");
        System.out.print("Enter Name: ");
        String name = scanner.next();
        System.out.print("Enter Age: ");
        int age = scanner.nextInt();
        student.setName(name);
        student.setAge(age);
        System.out.println("Student details updated successfully.");
    } else {
        System.out.println("Student not found.");
    }
}

private static void searchStudent(Scanner scanner) {
    System.out.print("Enter Roll Number to search: ");
    int rollNumber = scanner.nextInt();
    Student student = findStudent(rollNumber);
    if (student != null) {
        System.out.println("Student found: " + student);
    } else {
        System.out.println("Student not found.");
    }
}

private static void deleteStudentDetails(Scanner scanner) {
    System.out.print("Enter Roll Number to delete: ");
    int rollNumber = scanner.nextInt();
    Student student = findStudent(rollNumber);
    if (student != null) {
        students.remove(student);
        System.out.println("Student details deleted successfully.");
    } else {
        System.out.println("Student not found.");
    }
}

private static Student findStudent(int rollNumber) {
    for (Student student : students) {
        if (student.getRollNumber() == rollNumber) {
            return student;
        }
    }
    return null;
}
}
```

Output:

```
Menu:
1. Add Student details
2. Update Student details
3. Search for a Student
4. Delete Student details
5. Exit
Enter your choice: 1
Enter Roll Number: 005
Enter Name: Hansil
Enter Age: 19
Student details added successfully.

Menu:
1. Add Student details
2. Update Student details
3. Search for a Student
4. Delete Student details
5. Exit
Enter your choice: 3
Enter Roll Number to search: 005
Student found: Roll Number: 5, Name: Hansil, Age: 19
```

1. Create the above system user interface of menu using JAVAFX**Program/Solution:**

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.util.ArrayList;
import java.util.List;

class Student {
    private int rollNumber;
    private String name;
    private int age;

    public Student(int rollNumber, String name, int age) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.age = age;
    }

    // Getters and setters
    public int getRollNumber() {
        return rollNumber;
    }

    public void setRollNumber(int rollNumber) {
        this.rollNumber = rollNumber;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }
}
```

```
}

public void setAge(int age) {
    this.age = age;
}

@Override
public String toString() {
    return "Roll Number: " + rollNumber + ", Name: " + name + ", Age: "
+ age;
}

}

public class StudentManagementSystem extends Application {
    private List<Student> students = new ArrayList<>();

    @Override
    public void start(Stage primaryStage) {
        GridPane gridPane = new GridPane();
        gridPane.setAlignment(Pos.CENTER);
        gridPane.setHgap(10);
        gridPane.setVgap(10);
        gridPane.setPadding(new Insets(25, 25, 25, 25));

        Label menuLabel = new Label("Menu:");
        gridPane.add(menuLabel, 0, 0);

        Button addBtn = new Button("Add Student details");
        addBtn.setOnAction(e -> addStudentDetails(primaryStage));
        gridPane.add(addBtn, 0, 1);

        Button updateBtn = new Button("Update Student details");
        updateBtn.setOnAction(e -> updateStudentDetails(primaryStage));
        gridPane.add(updateBtn, 0, 2);

        Button searchBtn = new Button("Search for a Student");
        searchBtn.setOnAction(e -> searchStudent(primaryStage));
        gridPane.add(searchBtn, 0, 3);

        Button deleteBtn = new Button("Delete Student details");
        deleteBtn.setOnAction(e -> deleteStudentDetails(primaryStage));
        gridPane.add(deleteBtn, 0, 4);

        Button exitBtn = new Button("Exit");
        exitBtn.setOnAction(e -> primaryStage.close());
    }
}
```

```

        gridPane.add(exitBtn, 0, 5);

        Scene scene = new Scene(gridPane, 400, 300);
        primaryStage.setTitle("Student Management System");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private void addStudentDetails(Stage primaryStage) {
        GridPane addGridPane = new GridPane();
        addGridPane.setAlignment(Pos.CENTER);
        addGridPane.setHgap(10);
        addGridPane.setVgap(10);
        addGridPane.setPadding(new Insets(25, 25, 25, 25));

        Label rollNumberLabel = new Label("Roll Number:");
        addGridPane.add(rollNumberLabel, 0, 0);
        TextField rollNumberField = new TextField();
        addGridPane.add(rollNumberField, 1, 0);

        Label nameLabel = new Label("Name:");
        addGridPane.add(nameLabel, 0, 1);
        TextField nameField = new TextField();
        addGridPane.add(nameField, 1, 1);

        Label ageLabel = new Label("Age:");
        addGridPane.add(ageLabel, 0, 2);
        TextField ageField = new TextField();
        addGridPane.add(ageField, 1, 2);

        Button addDetailsBtn = new Button("Add Details");
        addDetailsBtn.setOnAction(e -> {
            int rollNumber = Integer.parseInt(rollNumberField.getText());
            String name = nameField.getText();
            int age = Integer.parseInt(ageField.getText());
            students.add(new Student(rollNumber, name, age));
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Success");
            alert.setHeaderText(null);
            alert.setContentText("Student details added successfully.");
            alert.showAndWait();
        });
        addGridPane.add(addDetailsBtn, 1, 3);

        Scene scene = new Scene(addGridPane, 400, 300);
    }

```



```
        primaryStage.setScene(scene);  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

Output: