

ZeeZee Bank

Account.java

```
public class Account {
    private long accountNumber;
    private double balanceAmount;

    public Account(long accountNumber, double balanceAmount) {
        this.accountNumber = accountNumber;
        this.balanceAmount = balanceAmount;
    }

    public long getAccountNumber() {
        return accountNumber;
    }

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

    public double getBalanceAmount() {
        return balanceAmount;
    }

    public void setBalanceAmount(double balanceAmount) {
        this.balanceAmount = balanceAmount;
    }

    public void deposit(double depositAmount) {
        balanceAmount += depositAmount;
    }

    public boolean withdraw(double withdrawAmount) {
        if (withdrawAmount <= balanceAmount) {
            balanceAmount -= withdrawAmount;
            return true;
        }

        return false;
    }
}
```

Main.java

```
import java.text.DecimalFormat;
```

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.00");

        System.out.println("Enter the account number:");
        long accountNumber = scanner.nextLong();

        System.out.println("Enter initial balance:");
        double balanceAmount = scanner.nextDouble();

        Account account = new Account(accountNumber, balanceAmount);

        System.out.println("Enter the amount to be deposited:");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);
        double availableBalance = account.getBalanceAmount();

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));

        System.out.println("Enter the amount to be withdrawn:");
        double withdrawAmount = scanner.nextDouble();
        boolean isWithdrawn = account.withdraw(withdrawAmount);
        availableBalance = account.getBalanceAmount();

        if (!isWithdrawn) {
            System.out.println("Insufficient balance");
        }

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));
    }
}

```

Numerology number

```

Main.java
import java.util.Scanner;

public class Main {
    private static int getSum(long num) {
        char[] chars = Long.toString(num).toCharArray();

```

```

    int sum = 0;

    for (char ch : chars) {
        sum += Character.digit(ch, 10);
    }

    return sum;
}

private static int getNumerology(long num) {
    String string = String.valueOf(num);

    while (string.length() != 1) {
        string = String.valueOf(getSum(Long.parseLong(string)));
    }

    return Integer.parseInt(string);
}

private static int getOddCount(long num) {
    int oddCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 != 0) {
            ++oddCount;
        }
    }

    return oddCount;
}

private static int getEvenCount(long num) {
    int evenCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 == 0) {
            ++evenCount;
        }
    }

    return evenCount;
}

public static void main(String[] args) {

```

```

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number");
long num = scanner.nextLong();

System.out.println("Sum of digits");
System.out.println(getSum(num));

System.out.println("Numerology number");
System.out.println(getNumerology(num));

System.out.println("Number of odd numbers");
System.out.println(getOddCount(num));

System.out.println("Number of even numbers");
System.out.println(getEvenCount(num));
    }
}

```

Substitution Cipher Technique

```

Main.java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        StringBuilder stringBuilder = new StringBuilder();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the encrypted text:");
        String text = scanner.nextLine();
        char[] chars = text.toCharArray();
        boolean flag = false;

        for (char ch : chars) {
            if (Character.isLetter(ch)) {
                flag = true;

                if (Character.isLowerCase(ch)) {
                    int sub = (int) ch - 7;

                    if (sub < 97) {
                        ch = (char) (122 - (97 - sub) + 1);
                    } else {

```

```

        ch = (char) sub;
    }
    } else if (Character.isUpperCase(ch)) {
        int sub = (int) ch - 7;

        if (sub < 65) {
            ch = (char) (90 - (65 - sub) + 1);
        } else {
            ch = (char) sub;
        }
    }
}

stringBuilder.append(ch);
} else if (Character.isWhitespace(ch)) {
    stringBuilder.append(ch);
}
}

if (flag) {
    System.out.println("Decrypted text:");
    System.out.println(stringBuilder.toString());
} else {
    System.out.println("No hidden message");
}
}
}

```

Bank Account - Interface

Account.java

```

public class Account {
    private String accountNumber;
    private String customerName;
    private double balance;

    public Account(String accountNumber, String customerName, double balance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }
}

```

```

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

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public double getBalance() {
        return balance;
    }

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

```

CurrentAccount.java

```

public class CurrentAccount extends Account implements MaintenanceCharge {
    public CurrentAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }

    @Override
    public float calculateMaintenanceCharge(float noOfYears) {
        return (100.0f + noOfYears) + 200.0f;
    }
}

```

MaintenanceCharge.java

```

public interface MaintenanceCharge {
    float calculateMaintenanceCharge(float noOfYears);
}

```

SavingsAccount.java

```

public class SavingsAccount extends Account implements MaintenanceCharge {
    public SavingsAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }
}

```

```

@Override
public float calculateMaintenanceCharge(float noOfYears) {
    return (50.0f * noOfYears) + 50.0f;
}
}

```

UserInterface.java

```

import java.text.DecimalFormat;
import java.util.Scanner;

```

```

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.0");

        System.out.println("1. Savings Account");
        System.out.println("2. Current Account");
        System.out.println("Enter your choice:");
        int choice = scanner.nextInt();

        System.out.println("Enter the Account number");
        String accountNumber = scanner.next();

        System.out.println("Enter the Customer Name");
        String customerName = scanner.next();

        System.out.println("Enter the Balance amount");
        double balance = scanner.nextDouble();

        System.out.println("Enter the number of years");
        int noOfYears = scanner.nextInt();

        System.out.println("Customer Name " + customerName);
        System.out.println("Account Number " + accountNumber);
        System.out.println("Account Balance " + decimalFormat.format(balance));

        switch (choice) {
            case 1: {
                SavingsAccount savingsAccount = new SavingsAccount(accountNumber,
customerName, balance);
                System.out.println("Maintenance Charge for Savings Account is Rs " +
decimalFormat.format(savingsAccount.calculateMaintenanceCharge(noOfYears)));
                break;
            }

```

```

        case 2: {
            CurrentAccount currentAccount = new CurrentAccount(accountNumber,
customerName, balance);
            System.out.println("Maintenance Charge for Current Account is Rs " +
decimalFormat.format(currentAccount.calculateMaintenanceCharge(noOfYears)));
        }
    }
}
}
}

```

Batting Average

UserInterface.java

```
package com.ui;
```

```
import com.utility.Player;
```

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
```

```

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Player player = new Player();
        player.setScoreList(new ArrayList<>());
        boolean flag = true;

        while (flag) {
            System.out.println("1. Add Runs Scored");
            System.out.println("2. Calculate average runs scored");
            System.out.println("3. Exit");
            System.out.println("Enter your choice");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1: {
                    System.out.println("Enter the runs scored");
                    int score = scanner.nextInt();
                    player.addScoreDetails(score);
                    break;
                }
                case 2: {
                    System.out.println("Average runs secured");

```



```

        System.out.println(player.getAverageRunScored());
        break;
    }
    case 3: {
        System.out.println("Thank you for use the application");
        flag = false;
        break;
    }
}
}
}
}

```

Player.java

```
package com.utility;
```

```
import java.util.List;
```

```

public class Player {
    private List<Integer> scoreList;

    public List<Integer> getScoreList() {
        return scoreList;
    }

    public void setScoreList(List<Integer> scoreList) {
        this.scoreList = scoreList;
    }

    public double getAverageRunScored() {
        if (scoreList.isEmpty()) {
            return 0.0;
        }

        int size = scoreList.size();
        int totalScore = 0;

        for (int score : scoreList) {
            totalScore += score;
        }

        return (double) totalScore / (double) size;
    }
}

```

```

    public void addScoreDetails(int score) {
        scoreList.add(score);
    }
}

```

Grade Calculation

Main.java

```
import java.util.Scanner;
```

```

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

        System.out.println("Enter the number of Threads:");
        int n = scanner.nextInt();

        GradeCalculator[] gradeCalculators = new GradeCalculator[n];
        Thread[] threads = new Thread[n];

        for (int i = 0; i < n; ++i) {
            System.out.println("Enter the String:");
            String string = scanner.next();
            String[] strings = string.split(":");
            int[] marks = new int[5];

            String studName = strings[0];

            for (int j = 1; j < 6; ++j) {
                marks[j - 1] = Integer.parseInt(strings[j]);
            }

            gradeCalculators[i] = new GradeCalculator(studName, marks);
            threads[i] = new Thread(gradeCalculators[i]);
            threads[i].start();
            threads[i].interrupt();
        }

        for (int i = 0; i < n; ++i) {
            System.out.println(gradeCalculators[i].getStudName() + ":" +
gradeCalculators[i].getResult());
        }
    }
}

```

Gradecalculator.java

```
public class GradeCalculator extends Thread {
    private String studName;
    private char result;
    private int[] marks;

    public GradeCalculator(String studName, int[] marks) {
        this.studName = studName;
        this.marks = marks;
    }

    public String getStudName() {
        return studName;
    }

    public void setStudName(String studName) {
        this.studName = studName;
    }

    public char getResult() {
        return result;
    }

    public void setResult(char result) {
        this.result = result;
    }

    public int[] getMarks() {
        return marks;
    }

    public void setMarks(int[] marks) {
        this.marks = marks;
    }

    @Override
    public void run() {
        int totalMarks = 0;

        for (int mark : marks) {
            totalMarks += mark;
        }
    }
}
```

```

        if (totalMarks <= 500 && totalMarks >= 400) {
            result = 'A';
        } else if (totalMarks < 400 && totalMarks >= 300) {
            result = 'B';
        } else if (totalMarks < 300 && totalMarks >= 200) {
            result = 'C';
        } else if (totalMarks < 200 && totalMarks >= 0) {
            result = 'E';
        }
    }
}

```

Employees eligible for promotionCoding exercise

Main.java

```

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

class Employee implements Comparable<Employee> {
    private final String id;
    private final LocalDate joiningDate;
    private boolean isEligible;

    public Employee(String id, LocalDate joiningDate) {
        this.id = id;
        this.joiningDate = joiningDate;
    }

    public void setIsEligible(LocalDate now) {
        isEligible = joiningDate.until(now, ChronoUnit.YEARS) >= 5;
    }

    public boolean getIsEligible() {
        return isEligible;
    }

    public String getId() {
        return id;
    }
}

```

```

    }

    @Override
    public String toString() {
        return id;
    }

    @Override
    public int compareTo(Employee employee) {
        return this.id.compareTo(employee.getId());
    }
}

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate now = LocalDate.parse("01/01/2019", dateTimeFormatter);
        int n = scanner.nextInt();
        ArrayList<Employee> employees = new ArrayList<>();

        IntStream.rangeClosed(1, 4).forEach(i -> {
            String id = scanner.next();
            String joiningDateStr = scanner.next();

            try {
                LocalDate joiningDate = LocalDate.parse(joiningDateStr, dateTimeFormatter);
                Employee employee = new Employee(id, joiningDate);
                employee.setIsEligible(now);
                employees.add(employee);
            } catch (Exception ignore) {
                System.out.println("Invalid date format");
                System.exit(0);
            }
        });

        List<Employee> filteredEmployees =
employees.stream().filter(Employee::getIsEligible).collect(Collectors.toList());

        if (filteredEmployees.isEmpty()) {
            System.out.println("No one is eligible");
        } else {
            Collections.sort(filteredEmployees);
            filteredEmployees.forEach(System.out::println);
        }
    }
}

```

```

    }
}
}

```

Check Number Type

NumberType.java

```

public interface NumberType {
    boolean checkNumber(int num);
}

```

NumberTypeUtility.java

```

import java.util.Scanner;

public class NumberTypeUtility {
    public static NumberType idOdd() {
        return (num) -> num % 2 != 0;
    }

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

        if (idOdd().checkNumber(num)) {
            System.out.println(num + " is odd");
        } else {
            System.out.println(num + " is not odd");
        }
    }
}

```

Retrieve Flight details based on source and destination

Main.java

```

import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        sc.next();
        String source=sc.nextLine();
        System.out.println("Enter the destination");
        String dest=sc.nextLine();
        FlightManagementSystem obj=new FlightManagementSystem();
    }
}

```

```

        ArrayList<Flight> res=obj.viewFlightsBySourceDestination(source,dest);
        if(res!=null)
            System.out.println(res);
        else
            System.out.println("No flights available for the given source and destination");
    }
}

```

DB.java

```

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

```

```

public class DB {

```

```

    private static Connection con = null;
    private static Properties props = new Properties();

```

```

    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
    public static Connection getConnection() throws ClassNotFoundException,
    SQLException {
        try{

            FileInputStream fis = null;
            fis = new FileInputStream("database.properties");
            props.load(fis);

            // load the Driver Class
            Class.forName(props.getProperty("DB_DRIVER_CLASS"));

            // create the connection now

            con =
            DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNA
            ME"),props.getProperty("DB_PASSWORD"));
        }
        catch(IOException e){
            e.printStackTrace();
        }

        return con;
    }
}

```

```
}
```

FlightManagementSystem.java

```
import java.util.*;
import java.sql.*;
public class FlightManagementSystem{
    public ArrayList<Flight> viewFlightsBySourceDestination(String source, String destination){
        DB db=new DB();
        ArrayList<Flight> list=new ArrayList<Flight>();
        try{
            int f=0;
            Connection con=db.getConnection();
            Statement st=con.createStatement();
            String sql= "select * from Flight where source= '"+source+"' and destination=
 '"+destination+"'";
            ResultSet rs=st.executeQuery(sql);
            while(rs.next()){
                f=1;
                Flight x=new Flight(rs.getInt(1), rs.getString(2),rs.getString(3), rs.getInt(4),
rs.getDouble(5));
                list.add(x);
            }
            con.close();
            if(f==1)
                return list;
            else
                return null;
        }
        catch(SQLException e){
            System.out.println("SQL Error. Contact Administrator.");
            return null;
        }
        catch(Exception e){
            System.out.println("Exception. Contact Administrator.");
            return null;
        }
    }
}
```

Flight.java

```
public class Flight {

    private int flightId;
    private String source;
```



```

private String destination;
private int noOfSeats;
private double flightFare;
public int getFlightId() {
    return flightId;
}
public void setFlightId(int flightId) {
    this.flightId = flightId;
}
public String getSource() {
    return source;
}
public void setSource(String source) {
    this.source = source;
}
public String getDestination() {
    return destination;
}
public void setDestination(String destination) {
    this.destination = destination;
}
public int getNoOfSeats() {
    return noOfSeats;
}
public void setNoOfSeats(int noOfSeats) {
    this.noOfSeats = noOfSeats;
}
public double getFlightFare() {
    return flightFare;
}
public void setFlightFare(double flightFare) {
    this.flightFare = flightFare;
}
public Flight(int flightId, String source, String destination,
              int noOfSeats, double flightFare) {
    super();
    this.flightId = flightId;
    this.source = source;
    this.destination = destination;
    this.noOfSeats = noOfSeats;
    this.flightFare = flightFare;
}

public String toString(){

```

```

        return ("Flight ID : "+getFlightId());
    }
}

```

Perform Calculation

```

import java.util.Scanner;

public class Calculator {

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        int a = sc.nextInt();

        int b= sc.nextInt();

        Calculate Perform_addition = performAddition();

        Calculate Perform_subtraction = performSubtraction();

        Calculate Perform_product = performProduct();

        Calculate Perform_division = performDivision();

        System.out.println("The sum is "+Perform_addition.performCalculation(a,b));

        System.out.println("The difference is
        "+Perform_subtraction.performCalculation(a,b));

        System.out.println("The product is "+Perform_product.performCalculation(a,b));

        System.out.println("The division value is
        "+Perform_division.performCalculation(a,b));

    }

    public static Calculate performAddition(){

        Calculate Perform_calculation = (int a,int b)->a+b;

        return Perform_calculation;
    }
}

```

```

    }

    public static Calculate performSubtraction(){
        Calculate Perform_calculation = (int a,int b)->a-b;
        return Perform_calculation;
    }

    public static Calculate performProduct(){
        Calculate Perform_calculation = (int a,int b)->a*b;
        return Perform_calculation;
    }

    public static Calculate performDivision(){
        Calculate Perform_calculation = (int a,int b)->{
            float c = (float)a;
            float d = (float)b;
            return (c/d);
        };
        return Perform_calculation;
    }
}

public interface Calculate {
    float performCalculation(int a,int b);
}

```

```

public class InPatient extends Patient {

    InPatient(String patientId, String patientname, long mobileNumber,
String gender) {
        super(patientId, patientname, mobileNumber, gender);
    }
    InPatient()
    {

    }
    private double roomRent;
    public double getrent()
    {
        return roomRent;
    }
    public void setrent(double rent)
    {
        roomRent=rent;
    }
    public double calculateTotalBill1(int no,double medi)
    {
        return ((roomRent*no)+medi);
    }
}

```

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class Main {
    public static void main(String [] args)throws IOException {
        Scanner sc=new Scanner(System.in);
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        OutPatient o1=new OutPatient();
        InPatient o2=new InPatient();
        System.out.println("1.In Patient\n2.Out Patient");
        System.out.println("Enter the choice");
        int a=sc.nextInt();
        //sc.hasNextLine();
        System.out.println("Enter the details\nPatient Id");
        String pid=br.readLine();
        System.out.println("Patient Name");
        String pname=br.readLine();
        System.out.println("Phone Number");
        long mob=sc.nextLong();
        System.out.println("Gender");
        String gen=br.readLine();
        if(a==1)
        {
            System.out.println("Room Rent");
            double rent=sc.nextDouble();

```

```

        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Number of Days of Stay");
        int no=sc.nextInt();
        c2.setrent(rent);
        System.out.println("Amount to be paid
"+c2.calculateTotalBill1(no,med));
    }
    else
    {
        System.out.println("Consultancy Fee");
        double con=sc.nextDouble();
        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Scan Pay");
        int scan=sc.nextInt();
        c1.setcon(con);
        System.out.println("Amount to be paid
"+c1.calculateTotalBill1(scan,med));
    }
}
}

```

```

public class OutPatient extends Patient {

    /*OutPatient(String patientId, String patientname, long
mobileNumber, String gender) {
    super(patientId, patientname, mobileNumber, gender);
    // TODO Auto-generated constructor stub
}*/
OutPatient()
{
    super();
}
private double consultingFee;
public double getcon()
{
    return consultingFee;
}
public void setcon(double con)
{
    consultingFee=con;
}
public double calculateTotalBill1(int scan,double medi)
{
    return (consultingFee+scan+medi);
}
}

```

```

public class Patient {

    private String patientId,patientname,gender;
    private long mobileNumber;
    Patient(String patientId,String patientname,long
mobileNumber,String gender)
    {
        this.patientId=patientId;
        this.patientname=patientname;
        this.gender=gender;
        this.mobileNumber=mobileNumber;
    }
    Patient()
    {
    }
    public String getpaid(){
        return patientId;
    }
    public String getpaname(){
        return patientname;
    }
    public String getpacer(){
        return gender;
    }
    public long getpanob(){
        return mobileNumber;
    }

    public void setpaid(String id){
        patientId=id;
    }
    public void setpaname(String name){
        patientname=name;
    }
    public void setpacer(String gen){
        gender=gen;
    }
    public void setpanob(long mob){
        mobileNumber=mob;
    }
}

```

Payment Inheritance

Bill.java

```
public class Bill {
```

```
    public String processPayment(Payment obj) {
```

```
        String message = "Payment not done and your due amount is "+obj.getDueAmount();
```

```

if(obj instanceof Cheque ) {

Cheque cheque = (Cheque) obj;

if(cheque.payAmount())

message = "Payment done succesfully via cheque";

}

else if(obj instanceof Cash ) {

Cash cash = (Cash) obj;

if(cash.payAmount())

message = "Payment done succesfully via cash";

}

else if(obj instanceof Credit ) {

Credit card = (Credit) obj;

if(card.payAmount())

message = "Payment done succesfully via creditcard. Remainig amount in your
"+card.getCardType()+" card is "+card.getCreditCardAmount();

}

return message;

}

}

```

Cash.java

```

public class Cash extends Payment{

    private int cashAmount;

    public int getCashAmount() {

```

```

        return cashAmount;

    }

    public void setCashAmount(int cashAmount) {

        this.cashAmount = cashAmount;

    }

    @Override

    public boolean payAmount() {

        return getCashAmount() >= getDueAmount();

    }

}

```

Cheque.java

```

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.GregorianCalendar;

public class Cheque extends Payment {

    private String chequeNo;

    private int chequeAmount;

    private Date dateOfIssue;

    public String getChequeNo() {
        return chequeNo;
    }
}

```



```
public void setChequeNo(String chequeNo) {  
    this.chequeNo = chequeNo;  
}
```

```
public int getChequeAmount() {  
    return chequeAmount;  
}
```

```
public void setChequeAmount(int chequeAmount) {  
    this.chequeAmount = chequeAmount;  
}
```

```
public Date getDateOfIssue() {  
    return dateOfIssue;  
}
```

```
public void setDateOfIssue(Date dateOfIssue) {  
  
    this.dateOfIssue = dateOfIssue;  
  
}
```

@Override

```
public boolean payAmount() {  
    int months = findDifference(getDateOfIssue());  
    return (getChequeAmount() >= getDueAmount() & months <= 6);  
}
```

```
private int findDifference(Date date) {  
    Calendar myDate = new GregorianCalendar();  
    myDate.setTime(date);  
    return (2020 - myDate.get(Calendar.YEAR)) * 12 + (0-myDate.get(Calendar.MONTH));  
}
```

```
public void generateDate(String date) {  
  
    try {  
        Date issueDate = new SimpleDateFormat("dd-MM-yyyy").parse(date);  
        setDateOfIssue(issueDate);  
    }  
    catch (ParseException e) {  
        e.printStackTrace();  
    }  
}
```

```
}  
}
```

Credit.java

```
public class Credit extends Payment {  
  
    private int creditCardNo;  
  
    private String cardType;  
  
    private int creditCardAmount;  
  
    public int getCreditCardNo(){  
  
        return creditCardNo;  
  
    }  
  
    public void setCreditCardNo(int creditCardNo) {  
  
        this.creditCardNo = creditCardNo;  
  
    }  
  
    public String getCardType() {  
  
        return cardType;  
  
    }  
  
    public void setCardType(String cardType) {  
  
        this.cardType = cardType;  
  
    }  
  
    public int getCreditCardAmount() {  
  
        return creditCardAmount;  
  
    }  
  
    public void setCreditCardAmount(int creditCardAmount) {
```

```
this.creditCardAmount = creditCardAmount;

}

@Override

public boolean payAmount() {

    int tax = 0;

    boolean isDeducted = false;

    switch(cardType) {

        case "silver":

            setCreditCardAmount(10000);

            tax = (int) (0.02*getDueAmount()+getDueAmount());

            if(tax <= getCreditCardAmount()) {

                setCreditCardAmount(getCreditCardAmount()-tax);

                isDeducted = true;

            }

            break;

        case "gold":

            setCreditCardAmount(50000);

            tax = (int) (0.05*getDueAmount()+getDueAmount());

            if(tax <= getCreditCardAmount()) {

                setCreditCardAmount(getCreditCardAmount()-tax);

                isDeducted = true;

            }

    }
```

```
break;

case "platinum":

setCreditCardAmount(100000);

tax = (int) (0.1*getDueAmount()+getDueAmount());

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

}

return isDeducted;

}

}
```

Main.java

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Bill bill = new Bill();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the due amount:");

        int dueAmount = sc.nextInt();

        System.out.println("Enter the mode of payment(cheque/cash/credit:");
```

```
String mode = sc.next();

switch (mode) {

case "cash":

System.out.println("Enter the cash amount:");

int cashAmount = sc.nextInt();

Cash cash = new Cash();

cash.setCashAmount(cashAmount);

cash.setDueAmount(dueAmount);

System.out.println(bill.processPayment(cash));

break;

case "cheque":

System.out.println("Enter the cheque number:");

String number = sc.next();

System.out.println("Enter the cheque amount:");

int chequeAmount = sc.nextInt();

System.out.println("Enter the date of issue:");

String date = sc.next();

Cheque cheque = new Cheque();

cheque.setChequeAmount(chequeAmount);

cheque.setChequeNo(number);

cheque.generateDate(date);

cheque.setDueAmount(dueAmount);
```

```
System.out.println(bill.processPayment(ch cheque));
break;

case "credit":

System.out.println("Enter the credit card number.");

int creditNumber = sc.nextInt();

System.out.println("Enter the card type(silver,gold,platinum)");

String cardType = sc.next();

Credit credit = new Credit();

credit.setCardType(cardType);

credit.setCreditCardNo(creditNumber);

credit.setDueAmount(dueAmount);

System.out.println(bill.processPayment(credit));

default:

break;

}

sc.close();

}

}
```

Payment.java

```
public class Payment {

    private int dueAmount;

    public int getDueAmount() {

        return dueAmount;

    }

}
```

```

    }

    public void setDueAmount(int dueAmount) {

        this.dueAmount = dueAmount;

    }

    public boolean payAmount() {

        return false;

    }

}

```

HUNGER EATS

```

package com.utility;
import java.util.*;
import com.bean.FoodProduct;
public class Order{
    private double discountPercentage;
    private List<FoodProduct> foodList=new ArrayList<FoodProduct>();

    public double getDiscountPercentage() {
        return discountPercentage;
    }
    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }
    public List<FoodProduct> getFoodList() {
        return foodList;
    }
    public void setFoodList(List<FoodProduct> foodList) {
        this.foodList = foodList;
    }

    public void findDiscount(String bankName)
    {
        if(bankName.equals("HDFC")) {
            discountPercentage=15.0;
        }
        else if(bankName.equals("ICICI")) {

```

```

        discountPercentage=25.0;
    }
    else if(bankName.equals("CUB")) {
        discountPercentage=30.0;
    }
    else if(bankName.equals("SBI")) {
        discountPercentage=50.0;
    }
    else if(bankName.equals("OTHERS")) {
        discountPercentage=0.0;
    }
}

public void addToCart(FoodProduct foodProductObject)
{
    List<FoodProduct> f=getFoodList();
    f.add(foodProductObject);
    setFoodList(f);
}

public double calculateTotalBill()
{
    double bill = 0;
    List<FoodProduct> f=getFoodList();
    for(int i=0;i<f.size();i++)
    {
        //      System.out.println(f.get(i).getCostPerUnit());
        //      System.out.println(f.get(i).getQuantity());
        bill+=f.get(i).getQuantity()*f.get(i).getCostPerUnit()*1.0;
    }
    //      System.out.println(bill);
    //      System.out.println(dis);
    bill=bill-((bill*discountPercentage)/100);
    return bill;
}
}
package com.ui;

import java.util.Scanner;

```



```

import com.utility.Order;
import com.bean.FoodProduct;

public class UserInterface{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int itemno;
        String bank;

        System.out.println("Enter the number of items");
        itemno=sc.nextInt();
        System.out.println("Enter the item details");

        Order o=new Order();

        for(int i=0;i<itemno;i++)
        {
            FoodProduct fd=new FoodProduct();
            System.out.println("Enter the item id");
            fd.setFoodId(sc.nextInt());
            System.out.println("Enter the item name");
            fd.setFoodName(sc.next());
            System.out.println("Enter the cost per unit");
            fd.setCostPerUnit(sc.nextDouble());
            System.out.println("Enter the quantity");
            fd.setQuantity(sc.nextInt());
            o.addToCart(fd);

        }

        System.out.println("Enter the bank name to avail offer");
        bank=sc.next();
        o.findDiscount(bank);

        System.out.println("Calculated Bill Amount:"+o.calculateTotalBill());

    }
}

```

```

}
package com.bean;

public class FoodProduct {

    private int foodId;
    private String foodName;
    private double costPerUnit;
    private int quantity;

    public int getFoodId() {
        return foodId;
    }
    public void setFoodId(int foodId) {
        this.foodId = foodId;
    }
    public String getFoodName() {
        return foodName;
    }
    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }
    public double getCostPerUnit() {
        return costPerUnit;
    }
    public void setCostPerUnit(double costPerUnit) {
        this.costPerUnit = costPerUnit;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}

```

Singapore

```

import java.util.*;
public class tourism {
    static String name;
    static String place;
    static int days;
    static int tickets;
}

```

```

static double price = 0.00;
static double total = 0.00;
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the passenger name");
    name = in.nextLine();
    System.out.println("Enter the place name");
    place=in.nextLine();
    if(place.equalsIgnoreCase("beach")

||place.equalsIgnoreCase("pilgrimage")||place.equalsIgnoreCase("heritage")||place.equalsIgno
reCase("Hills")||place.equalsIgnoreCase("palls")||place.equalsIgnoreCase("adventure")){
        System.out.println("Enter the number of days");
        days = in.nextInt();
        if(days>0){
            System.out.println("Enter the number of Tickets");
            tickets = in.nextInt();
            if(tickets>0){
                if(place.equalsIgnoreCase("beach")){
                    price = tickets*270;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price: %.2f",total);
                    }
                }
                else {
                    System.out.printf("Price: %.2f",price);
                }
            }
        }
        else if(place.equalsIgnoreCase("pilgrimage")){
            price = tickets*350;
            if(price>1000){
                total = 85*price/100;
                System.out.printf("Price: %.2f",total);
            }
        }
        else {
            System.out.printf("Price: %.2f",price);
        }
    }
    else if(place.equalsIgnoreCase("heritage")){
        price = tickets*430;
        if(price>1000){
            total = 85*price/100;
            System.out.printf("Price: %.2f",total);
        }
    }
}

```

```

else {
    System.out.printf("Price: %.2f", price);
}
}
else if(place.equalsIgnoreCase("hills")){
    price = tickets*780;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price: %.2f", total);
    }
    else {
        System.out.printf("Price: %.2f", price);
    }
}
else if(place.equalsIgnoreCase("palls")){
    price = tickets*1200;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price: %.2f", total);
    }
    else {
        System.out.printf("Price: %.2f", price);
    }
}
else {
    price = tickets*4500;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price: %.2f", total);
    }
    else {
        System.out.printf("Price: %.2f", price);
    }
}

}
else{
    System.out.println(tickets+" is an Invalid no. of tickets");
}
}
else{
    System.out.println(days+" is an Invalid no. of days");
}
}
}

```

```

    else {
    System.out.println(place+" is an Invalid place");
    }
}
}

```

Prime no ending

```

import java.util.*;
public class Main
{
    public static void main (String[] args) {
        int flag=0, k=0, z=0;
        Scanner sc =new Scanner(System.in );
        System.out.println("Enter the first number");
        int f=sc.nextInt();
        System.out.println("Enter the last number");
        int l=sc.nextInt();
        for(int i=f; i<=l; i++)
        {
            for(int j=2; j<i; j++)// this loop increments flag if i is divisible by j
            {
                if(i%j==0)
                {
                    flag++;
                }
            }
            if(i==l && (flag!=0 || i%10!=1))//when last number is not a prime
            {
                while(z==0)
                {
                    for(int a=2; a<i; a++)
                    {
                        if(i%a==0)
                        {
                            flag++;
                        }
                    }
                    if(i%10==1 && flag==0)
                    {
                        System.out.print(", "+i);
                        z++;
                    }
                }
                flag=0;
                i++;
            }
        }
    }
}

```

```

    }
}
if(i%10==1 && flag==0)//to check for last digit 1 and prime
{
    if(k==0)
    {
        System.out.print(i);
        k++;
    }
    else
    {
        System.out.print(", "+i);
    }
}
flag=0;
}
}
}

```

Query Set

```

public class Query {

    private class DataSet{
        private String theatreId;
        private String theatreName;
        private String location;
        private int noOfScreen;
        private double ticketCost;
        public String getTheatreId() {
            return theatreId;
        }
        public void setTheatreId(String theatreId) {
            this.theatreId = theatreId;
        }
        public String getTheatreName() {
            return theatreName;
        }
        public void setTheatreName(String theatreName) {
            this.theatreName = theatreName;
        }
        public String getLocation() {
            return location;
        }
    }
}

```

```

public void setLocation(String location) {
this.location = location;
}
public int getNoOfScreen() {
return noOfScreen;
}
public void setNoOfScreen(int noOfScreen) {
this.noOfScreen = noOfScreen;
}
public double getTicketCost() {
return ticketCost;
}
public void setTicketCost(double ticketCost) {
this.ticketCost = ticketCost;
}
@Override
public String toString() {
return "Theatre id: " + theatreId + "\nTheatre name: " + theatreName + "\nLocation: " + location
+ "\nNo of Screen: " + noOfScreen + "\nTicket Cost: " + ticketCost+"\n";
}
}

```

```

private String queryId;
private String queryCategory;
private DataSet primaryDataset;
private DataSet secondaryDataSet;
public String getQueryId() {
return queryId;
}
public void setQueryId(String queryId) {
this.queryId = queryId;
}
public String getQueryCategory() {
return queryCategory;
}
public void setQueryCategory(String queryCategory) {
this.queryCategory = queryCategory;
}
public DataSet getPrimaryDataset() {
return primaryDataset;
}
public void setPrimaryDataset(DataSet primaryDataset) {
this.primaryDataset = primaryDataset;
}
}

```

```

public DataSet getSecondaryDataSet() {
    return secondaryDataSet;
}
public void setSecondaryDataSet(DataSet secondaryDataSet) {
    this.secondaryDataSet = secondaryDataSet;
}
@Override
public String toString() {
    return "Primary data set\n" + primaryDataset
    + "Secondary data set\n" + secondaryDataSet + "Query id: " + queryId + "\nQuery category=" +
    queryCategory;
}

}

```

```

import java.util.Scanner;
public class TestApplication {
    public static void main(String[] args) {
        Query query = new Query();
        Scanner sc = new Scanner(System.in);
        Query.DataSet primary = query.new DataSet();
        Query.DataSet secondary = query.new DataSet();
        System.out.println("Enter the Details of primary data set");
        System.out.println("Enter the theatre id");
        String theatreid = sc.next();
        primary.setTheatreid(theatreid);
        sc.nextLine();
        System.out.println("Enter the theatre name");
        String theatrename = sc.next();
        primary.setTheatreName(theatrename);
        sc.nextLine();
        System.out.println("Enter the location");
        String location = sc.next();
        primary.setLocation(location);
        sc.nextLine();
        System.out.println("Entrer the no of screens");
        int screens = sc.nextInt();
        primary.setNoOfScreen(screens);
        System.out.println("Ente the ticket cost");
        double cost = sc.nextDouble();
        primary.setTicketCost(cost);
    }
}

```



```

System.out.println("ENter the details of secondary data set");
System.out.println("Enter the theatre id");
theatreid = sc.next();
secondary.setTheatreId(theatreid);
sc.nextLine();
System.out.println("Enter the theatre name");
theatrename = sc.next();
secondary.setTheatreName(theatrename);
sc.nextLine();
System.out.println("Enter the location");
location = sc.next();
secondary.setLocation(location);
sc.nextLine();
System.out.println("Entrer the no of screens");
screens = sc.nextInt();
secondary.setNoOfScreen(screens);
System.out.println("Ente the ticket cost");
cost = sc.nextDouble();
secondary.setTicketCost(cost);
System.out.println("Enter the query id");
String queryid = sc.next();
query.setQueryId(queryid);
sc.nextLine();
System.out.println("Enter the query category");
String querycategory = sc.next();
query.setQueryCategory(querycategory);
sc.nextLine();
query.setPrimaryDataset(primary);
query.setSecondaryDataSet(secondary);

System.out.println(query);
}
}

```

Extract book

```

import java.util.Scanner;

class ExtractBook {

    public static int extractDepartmentCode(String input) {
        return Integer.parseInt(input.substring(0, 3));
    }
}

```

```

public static String extractDepartmentName(int code) {

    switch (code) {
    case 101:
        return "Accounting";
    case 102:
        return "Economics";
    case 103:
        return "Engineering";
    }

    throw new Error(code + " is invalid department code");
}

```

```

public static int extractDate(String input) {
    String yearStr = input.substring(3, 7);
    try {
        int year = Integer.parseInt(yearStr);
        if (year > 2020 || year < 1900) {
            throw new NumberFormatException();
        }
        return year;
    } catch (NumberFormatException e) {
        throw new Error(yearStr + " is invalid year");
    }
}

```

```

public static int extractNumberOfPages(String input) {
    String pagesStr = input.substring(7, 12);
    try {
        int pages = Integer.parseInt(pagesStr);
        if (pages < 10) {
            throw new NumberFormatException();
        }
        return pages;
    } catch (NumberFormatException e) {
        throw new Error(pagesStr + " are invalid pages");
    }
}

```

```

public static String extractBookId(String input) {
    String id = input.substring(12, 18);
    if (!Character.isAlphabetic(id.charAt(0)))

```

```

        throw new NumberFormatException();
    try {
        Integer.parseInt(id.substring(1));
    } catch (NumberFormatException e) {
        throw new Error(id + " is invalid book id");
    }
    return id;
}

public static void parseAndPrint(String str) {
    if (str.length() != 18) {
        System.out.println(str + " is an invalid input");
        return;
    }

    try {
        int dCode = extractDepartmentCode(str);
        String dString = extractDepartmentName(dCode);
        int year = extractDate(str);
        int pages = extractNumberOfPages(str);
        String bookId = extractBookId(str);

        System.out.println("Department Code: " + dCode);
        System.out.println("Department Name: " + dString);
        System.out.println("Year of Publication: " + year);
        System.out.println("Number of Pages: " + pages);
        System.out.println("Book Id: " + bookId);

    } catch (Error e) {
        System.out.println(e.getMessage());
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String input = sc.nextLine();
    parseAndPrint(input);
    sc.close();
}
}

```

Fixed deposit

```

import java.util.*;
class FDScheme {

    private int schemeNo;
    private double depositAmt;
    private int period;
    private float rate;
    public FDScheme(int schemeNo, double depositAmt, int period) {
        super();
        this.schemeNo = schemeNo;
        this.depositAmt = depositAmt;
        this.period = period;
        calculateInterestRate();
    }
    public int getSchemeNo() {
        return schemeNo;
    }
    public void setSchemeNo(int schemeNo) {
        this.schemeNo = schemeNo;
    }
    public double getDepositAmt() {
        return depositAmt;
    }
    public void setDepositAmt(double depositAmt) {
        this.depositAmt = depositAmt;
    }
    public int getPeriod() {
        return period;
    }
    public void setPeriod(int period) {
        this.period = period;
    }
    public float getRate() {
        return rate;
    }
    public void setRate(float rate) {
        this.rate = rate;
    }

    public void calculateInterestRate()
    {
        if(period>=1 && period<=90)
        {
            this.rate=(float) 5.5;

```

```

    }
    else if(period>=91 && period<=180)
    {
        this.rate=(float) 6.25;
    }
    else if(period>=181 && period<=365)
    {
        this.rate=(float) 7.5;
    }
    System.out.println("Interest rate for "+period+" days is "+this.rate);
}
}
public class Main{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Scheme no");
        int no=sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Deposit amount");
        double amt=sc.nextDouble();
        System.out.println("enter period of deposit");
        int prd=sc.nextInt();
        FDScheme obj=new FDScheme(no,amt,prd);
    }
}

```

Annual Salary

```

import java.io.*;
public class Main
{
    public static void main(String[] args)throws IOException
    {
        // Scanner sc=new Scanner(System.in);
        //Fill the code
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the Employee Name");
        String name=br.readLine();
        System.out.println("Enter percentage of salary");
        double percent=Double.parseDouble(br.readLine());
        if(percent>0&&percent<20)
        {

```

```

System.out.println("Enter the Year of Experience");
int time=Integer.parseInt(br.readLine());

if(time>0&&time<15)
{
    double permonth=12000+(2000*(time));
    double dayshift=permonth*6;
    double nightshift=((permonth*percent)/100)+permonth)*6;
    double annualIncome=dayshift+nightshift;

    String str="The annual salary of "+name+" is";
    System.out.println(str+" "+annualIncome);

}
else{
    System.out.println((int)time+" is an invalid year of experience");}

}
else
    System.out.println((int)percent+" is an invalid percentage");
}
}

```

Amity Passenger

```

import java.util.*;
public class PassengerAmenity {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number of passengers");
        int no=sc.nextInt();
        sc.nextLine();
        int count=0;

        if(no>0)
        {
            String name[]=new String[no];
            String seat[]=new String[no];
            String arr[]=new String[no];

            for(int i=0;i<no;i++)

```

```

{
    System.out.println("Enter the name of the passenger "+(i+1));
    String str=sc.nextLine();

    name[i]=str.toUpperCase();

    System.out.println("Enter the seat details of the passenger "+(i+1));
    seat[i]=sc.nextLine();

    if(seat[i].charAt(0)>='A' && seat[i].charAt(0)<='S')
    {

        int r=Integer.parseInt(seat[i].substring(1,seat[i].length()));

        if(r>=10 && r<=99)
        {
            count++;
        }

        else
        {
            System.out.println(r+" is invalid seat number");
            break;
        }
    }

    else
    {
        System.out.println(seat[i].charAt(0)+" is invalid coach");
        break;
    }

    arr[i]=name[i]+" "+seat[i];
}

if(count==seat.length)
{

    Arrays.sort(seat);

    for(int i=seat.length-1;i>=0;i--)
    {
        for(int j=0;j<arr.length;j++)
        {

```

```

        if(arr[j].contains(seat[i]))
        {
            System.out.println(arr[j]);
        }
    }
}

else
{
    System.out.println(no+" is invalid input");
}
}
}

```

Change the Case

```

import java.util.*;

public class ChangeTheCase {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String a = sc.next();
        if(a.length() < 3) {
            System.out.println("String length of " + a + " is too short");
            return;
        }
        else if(a.length() > 10) {
            System.out.println("String length of " + a + " is too long");
            return;
        }

        char[] arr = a.toCharArray();
        char[] arr1 = new char[arr.length];
        int j = 0;
        for(int i = 0; i < a.length(); i++) {
            if((arr[i]<65 || ((arr[i]>90) && (arr[i]<97)) || arr[i]>122)) {
                arr1[j++] = arr[i];
            }
        }
    }
}

```



```

    }
    if(j!=0) {
        System.out.print("String should not contain ");
        for(int i = 0; i<=j; i++) {
            System.out.print(arr1[i]);
        }
        return;
    }
    char b = sc.next().charAt(0);
    int present = 0;
    for(int i = 0; i<a.length(); i++) {
        if(arr[i] == Character.toUpperCase(b)) {
            arr[i] = Character.toLowerCase(b);
            present = 1;
        }
        else if(arr[i] == Character.toLowerCase(b)) {
            arr[i] = Character.toUpperCase(b);
            present = 1;
        }
    }
    if(present == 0) {
        System.out.println("Character " + b + " is not found");
    }
    else {
        for(int i = 0; i <a.length(); i++) {
            System.out.print(arr[i]);
        }
    }
}
}

```

Club Member

```
import java.util.Scanner;
```

```

public class ClubMember {
    private int memberId;
    private String memberName;
    private String memberType;
    private double membershipFees;

```

```

    public ClubMember(int memberId, String memberName, String memberType) {

```

```

super();
this.memberId = memberId;
this.memberName = memberName;
this.memberType = memberType;
calculateMembershipFees();
}
public int getMemberId() {
    return memberId;
}
public void setMemberId(int memberId) {
    this.memberId = memberId;
}
public String getMemberName() {
    return memberName;
}
public void setMemberName(String memberName) {
    this.memberName = memberName;
}
public String getMemberType() {
    return memberType;
}
public void setMemberType(String memberType) {
    this.memberType = memberType;
}
public double getMembershipFees() {
    return membershipFees;
}
public void setMembershipFees(double membershipFees) {
    this.membershipFees = membershipFees;
}

public void calculateMembershipFees() {
    if(!(memberType == "Gold"))
    {

        this.membershipFees=(double) 50000.0;
    }
    else if(!(memberType=="Premium"))
    {
        this.membershipFees=(double) 75000.0;
    }
    System.out.println("Member Id is "+this.memberId);
    System.out.println("Member Name is "+this.memberName);
    System.out.println("Member Type is "+this.memberType);
}

```

```
        System.out.println("Membership Fees is "+this.membershipFees);  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter Member Id");  
        int id=sc.nextInt();  
        sc.nextLine();  
        System.out.println("Enter Name");  
        String name=sc.next();  
        System.out.println("Enter Member Type");  
        String type=sc.next();  
        ClubMember club=new ClubMember(id, name, type);  
        //club.calculateMembershipFees();  
    }  
}
```