FIXED DEPOSITE

```
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);
}
}
```

EXTRACT BOOK DETAILS

```
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();
}
}
```

BANK ACCOUNT DETAILS

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)));
}
}
}
}
```

ALTERNATIVE NUMBER DIFFERNCE

```
import java.util.Scanner;
public class Main{ public static void main (String[] args) {
Scanner sc = new Scanner(System.in);
int arr_size = sc.nextInt();
if(arr_size <= 3){
System.out.println("Invalid array size");
System.exit(0);
}
int[] arr_int = new int[arr_size];
int big_diff = 0;
int small_index = 0;
for(int i = 0; i < arr_size; i++)
```

```
arr_int[i] = sc.nextInt();
for(int j = 0; j < arr_size - 2; j++){
if(Math.abs(arr_int[j] - arr_int[j+2]) > big_diff){
big_diff = Math.abs(arr_int[j] - arr_int[j+2]);
if(arr_int[j] <= arr_int[j+2])
small_index = j;
else
small_index = j + 2;
}
 }
System.out.println(small_index);
}
 }
```

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]);
                }
            }
        }

}
```

PAYMENT INHERTIENCE

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(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;
}
}
```

PERFORM CALUCLATION

```
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);
}
```

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");
}
}
}
```

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");
}
}
}
```