JAVA ICT CODES

Rainfall Report Automation

AnnualRainfall

```
public class AnnualRainfall {


                                        private int cityPincode;
                                        private String cityName;
                                        private double averageAnnualRainfall;

                                        public int getCityPincode() {
                                                return cityPincode;
                                        }

                                        public void setCityPincode(int cityPincode) {
                                                this.cityPincode = cityPincode;
                                        }

                                        public String getCityName(){
                                                return cityName;
                                        }

                                        public void setCityName(String cityName){
                                                this.cityName = cityName;
                                        }
public double getAverageAnnualRainfall(){
        return averageAnnualRainfall;
}

public void
setAverageAnnualRainfall(double
averageAnnualRainfall){
        this.averageAnnualRainfall =
averageAnnualRainfall;
}

public void calculateAverageAnnualRainfall
(double monthlyRainfall [ ]){

        double average=0;
  for(int i=0;i<monthlyRainfall.length;i++)
  {
        average+=monthlyRainfall[i];
  }
  average/=12;
  this.averageAnnualRainfall=average;
}

}
```

DBHandler

```
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 DBHandler {

private static Connection con = null;

private static Properties props = new Properties();

//Write the required business logic as expected in the question description

public Connection establishConnection() throws ClassNotFoundException, SQLException {

        try{

                FileInputStream fis = null;

                fis = new FileInputStream("db.properties");

                props.load(fis);


                // load the Driver Class


                Class.forName(props.getProperty("db.classname"));


                // create the connection now

        con = DriverManager.getConnection(props.getProperty("db.url"),props.getProperty("db.username"),props.getProperty("db.password"));
```

```
                                                                      }
                                                                      catch(IOException e){
                                                                         e.printStackTrace();
                                                                      }
                                                                           return con;
                                                                    }
}
```

InvalidCityPincodeException

```
@SuppressWarnings("serial")
public class InvalidCityPincodeException extends Exception {
                                                                    public
                                                             InvalidCityPincodeException(String s)
                                                             {
                                                                    super(s);
}
}
```

Main

```
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class Main {

                                                             public static void main(String[] args)
                                                                           throws IOException,
                                                             InvalidCityPincodeException,
                                                             ClassNotFoundException, SQLException {
```

```
                                                RainfallReport rf = new
                                         RainfallReport();

                                                List<AnnualRainfall> avgli = new
                                         ArrayList<AnnualRainfall>();

                                                avgli =
                                         rf.generateRainfallReport("AllCityMonthlyR
                                         ainfall.txt");

                                                List<AnnualRainfall> maxli = new
                                         ArrayList<AnnualRainfall>();

                                                maxli =
                                         rf.findMaximumRainfallCities();

                                                for (int i = 0; i < maxli.size(); i++) {

                                                        AnnualRainfall ob =
                                         maxli.get(i);

                                                        System.out.println("City
                                         Pincode:" + ob.getCityPincode());

                                                        System.out.println("City
                                         Name:" + ob.getCityName());


                                                System.out.println("Average
                                         RainFall:" + ob.getAverageAnnualRainfall());

                                                }


                                         }


}
```

RainfallReport

```
import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.sql.Connection;
import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;


public class RainfallReport {



                                        public List<AnnualRainfall>
                                        generateRainfallReport(String filePath)
                                        throws IOException {


                                                List<AnnualRainfall> avgList=new
                                        ArrayList<>();

                                                FileReader fr = new FileReader(new
                                        File(filePath));

                                                BufferedReader br = new
                                        BufferedReader(fr);

                                                String l;

                                                while((l=br.readLine())!=null)

                                                {

                                                        String[] a=l.split(",");

                                                        String pincode=a[0];

                                                        try

                                                        {

                                                                if(validate(pincode))

                                                                {

                                                                        double[]
                                        monthlyRainFall=new double[12];

                                                                        for(int
                                        i=2;i<=13;i++)

                                                                        {
```

```
            monthlyRainFall[i-
2]=Double.parseDouble(a[i]);
                                                }

            AnnualRainfall ar=new
AnnualRainfall();

            ar.calculateAverageAnnualRainfall(
monthlyRainFall);

            ar.setCityName(a[1]);

            ar.setCityPincode(Integer.parseInt(p
incode));

            avgList.add(ar);
                                    }
                        }

            catch(InvalidCityPincodeException
e)
                        {

            System.out.println(e.getMessage());
                        }
            }
            br.close();
            return avgList;
}

public List<AnnualRainfall>
findMaximumRainfallCities() throws
SQLException, ClassNotFoundException,
IOException  {
            DBHandler d=new DBHandler();
```

```
		List<AnnualRainfall> finalList=new 
ArrayList<>();
		Connection 
c=d.establishConnection();
		Statement s=c.createStatement();
		String sql = "SELECT * FROM 
ANNUALRAINFALL WHERE 
AVERAGE_ANNUAL_RAINFALL IN (SELECT 
MAX(AVERAGE_ANNUAL_RAINFALL) FROM 
ANNUALRAINFALL)";
		ResultSet rs=s.executeQuery(sql);
		while(rs.next())
		{
			AnnualRainfall ar=new 
AnnualRainfall();

		ar.setCityName(rs.getString(2));

		ar.setCityPincode(Integer.parseInt(r
s.getString(1)));

		ar.setAverageAnnualRainfall(Double
.parseDouble(rs.getString(3)));
			finalList.add(ar);
		}
		return finalList;
}



public boolean validate(String cityPincode) 
throws InvalidCityPincodeException {
		if(cityPincode.length()==5)
		{
			return true;
		}
		else
```

```
            {
                        throw new
InvalidCityPincodeException("Invalid
sCityPincode Exception");
            }
}
}
```

ElectricityBill Automation

DBHandler

```
import java.sql.Connection;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.*;
import java.util.Properties;
import java.io.FileNotFoundException;

public class DBHandler {

public Connection establishConnection()
throws ClassNotFoundException,
SQLException, FileNotFoundException {

    Connection con = null;
```

```
   Properties props = new Properties();

   // this try block reads the db
Properties file and establishConnection.

   try{

      FileInputStream fis = new
FileInputStream("src/db.properties");

      props.load(fis);



Class.forName(props.getProperty("db.cl
assname"));



      con =
DriverManager.getConnection(props.get
Property("db.url"),props.getProperty("d
b.username"),props.getProperty("db.pa
ssword"));

   }

   catch(IOException e){

      e.printStackTrace();

   }


   return con;




        //fill code here



}

}
```

ElectricityBill

```
}

//This is the POJO/model class

public class ElectricityBill {

    private String consumerNumber;
    private String consumerName;
    private String consumerAddress;
    private int unitsConsumed;
    private double billAmount;

    public String getConsumerNumber() {
        return consumerNumber;
    }

    public void setConsumerNumber(String consumerNumber) {
        this.consumerNumber = consumerNumber;
    }

    public String getConsumerName() {
        return consumerName;
    }

    public void setConsumerName(String consumerName) {
        this.consumerName = consumerName;
    }
```

```
public String getConsumerAddress() {
	return consumerAddress;
}

public void setConsumerAddress(String
consumerAddress) {
	this.consumerAddress =
consumerAddress;
}

public int getUnitsConsumed() {
	return unitsConsumed;
}

public void setUnitsConsumed(int
unitsConsumed) {
	this.unitsConsumed =
unitsConsumed;
}

public double getBillAmount() {
	return billAmount;
}

public void setBillAmount(double
billAmount) {
	this.billAmount = billAmount;
}
```

```
//Write the required business logic as
expected in the question description

public void calculateBillAmount() {

   // method for calaculating the bill
amount.

   int units = unitsConsumed;

   double bill = 0;


   if(units <= 100){

      bill = 0;

   }

   if(units > 100 && units <= 300){

      bill = (units-100) * 1.5;

   }

   if(units > 300 && units <= 600){

      bill = 200 * 1.5 + (units-300) * 3.5;

   }

   if(units > 600 && units <= 1000){

      bill = 200 * 1.5 + 300 * 3.5 + (units-
600) * 5.5;

   }

   if(units > 1000){

      bill = 200 * 1.5 + 300 * 3.5 + 400 *
5.5 + (units-1000) * 7.5;

   }


   setBillAmount(bill);



//fill the code
```

```
    }

}
```

ElectricityBoard

```
import java.util.List;
import java.util.*;
import java.io.FileReader;
import java.io.File;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.regex.Pattern;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;

public class ElectricityBoard {

    //write the required business logic methods as expected in the question description
    public void addBill(List<ElectricityBill> billList) {

        DBHandler db =  new DBHandler();

        try(Connection con = db.establishConnection()){

            PreparedStatement stmt = con.prepareStatement("insert into ElectricityBill
                                                                        values(?,?,?,?,?);");
    // for loop to insert the values into the table
    for(ElectricityBill obj : billList){
        stmt.setString(1,obj.getConsumerNumber());
        stmt.setString(2,obj.getConsumerName());
        stmt.setString(3,obj.getConsumerAddress());
        stmt.setInt(4,obj.getUnitsConsumed());
        stmt.setDouble(5,obj.getBillAmount());

        stmt.execute();
    }
}
catch(ClassNotFoundException e){
    e.printStackTrace();
}

catch(FileNotFoundException e){
    e.printStackTrace();
}

catch(SQLException e){
    e.printStackTrace();
}




//fill the code
```

```
}

public List<ElectricityBill> generateBill(String filePath) {

    List <ElectricityBill> list = new ArrayList<>();
    File f = new File (filePath);

    // this try block is for opening and reading the file
    try(BufferedReader br = new BufferedReader(new FileReader(f)))
    {
        String line = null;

        while((line = br.readLine())!= null)
        {
            String records[] = null;
            String consumerNumber = "";
            String consumerName = "";
            String consumerAddress = "";
            int unitsConsumed = 0;

            records = line.split(",");
            consumerNumber = records[0];
            consumerName = records[1];
            consumerAddress = records[2];
            unitsConsumed = Integer.parseInt(records[3]);


        //this try block checks for the validated consumerNumber
            try{
```

```
            if(validate(consumerNumber)){
               ElectricityBill obj = new ElectricityBill();
               obj.setConsumerNumber(consumerNumber);
               obj.setConsumerName(consumerName);
               obj.setConsumerAddress(consumerAddress);
               obj.setUnitsConsumed(unitsConsumed);
               obj.calculateBillAmount();

               list.add(obj);
            }
         }
         catch(InvalidConsumerNumberException e){
            System.out.println(e.getMessage());
         }
      }
   }
   catch(FileNotFoundException e){
      e.printStackTrace();
   }
   catch(IOException e){
      e.printStackTrace();
   }

return list;
   //fill the code

}

public boolean validate(String consumerNumber) throws
                                                    InvalidConsumerNumberException {
```

```
        // method for validating the consumerNumber
          boolean isValid = Pattern.matches("^[0][0-9]{9}" , consumerNumber);

          if(!isValid){
             throw new InvalidConsumerNumberException("Invalid Consumer Number");
          }

          return true;

                                                            //fill the code

    }

}
```

InvalidConsumerNumberException

```
//make the required changes to this class so that InvalidConsumerNumberException is of
                                                          type exception.

public class InvalidConsumerNumberException extends Exception{

   public InvalidConsumerNumberException(String message)
   {
      super(message);
   }
```

```
    //fill the code


}
```

Main

```
import java.util.*;
import java.util.List;
import java.util.ArrayList;
public class Main {

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

       String filePath = "src/ElectricityBill.txt";

       List<ElectricityBill> list = new ArrayList<>();

       ElectricityBoard eb = new ElectricityBoard();
       list = eb.generateBill(filePath);

       for(ElectricityBill obj: list){
           System.out.println(obj.getConsumerNumber() + " " + obj.getConsumerName() + " " +
                                                                       obj.getBillAmount());

       }

       eb.addBill(list);
```

```
        System.out.println("Successfully Inserted");

        sc.close();
                                                                     //fill your code here

    }

}
```

CreditCardAdminSystem

CreditCardDAO

```
/*****************************************************************************
                                                  ***********************************

 * This class CreditCardDAO is used to persist or retrieve data from database.

 *

 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                  EXCEPTION CLAUSES, RETURN TYPES

 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS

 * DO TEST YOUR CODE USING MAIN METHOD

 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC

 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                  NEED BE,

 * YOU CAN CATCH THEM AND THROW ONLY

 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

 *
```

```
****************************************************************************
                                                  ********************************
                                                  /

package com.cts.creditcard.dao;


import java.sql.Connection;

import java.util.List;


import com.cts.creditcard.exception.CreditCardAdminSystemException;

import com.cts.creditcard.vo.CreditCard;



public class CreditCardDAO {


                                                  private static Connection conn = null;


                                                  public Boolean
                                                  addCreditCardDetails(List<CreditCard>
                                                  cards) throws
                                                  CreditCardAdminSystemException {

                                                   //TODO add your code here

                                                              return false;    //TODO CHANGE
                                                  THIS RETURN TYPE


                                                  }

}
```

CreditCardAdminSystemException

```
/***************************************************************************
                                                  ********************************
```

```
 * This class CreditCardAdminSystemException is as exception class.
 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                  EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                  NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY
 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE
 *
************************************************************************
                                                  ********************************
                                                  /
package com.cts.creditcard.exception;

public class CreditCardAdminSystemException  extends Exception {

                                                  private static final long serialVersionUID
                                                  = -6349759544203601561L;

                                                  //TODO add your constructors here
}
```

MainApp

```
/************************************************************************
                                                  ********************************
 * This class MainApp is used to run the service methods and to test the database.
 *
 * YOU CAN INVOKE THE METHODS AS REQUIRED FROM HERE TO TEST THE APP
 * DO NOT USE VIA COMMAND LINE ARGUMENTS FROM MAIN METHOD, UNLESS SPECIFIED
 *
****************************************************************************
                                                    *********************************
                                                    /
package com.cts.creditcard.main;


public class MainApp {
                                                    public static void main(String ag[]) {
                                                            //TODO add your code here
                                                    }
}
```

CreditCardAdminService

```
/***************************************************************************
                                                    *********************************
 * This class CreditCardAdminService is used to handle business logic for the proposed
                                                    system.
 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                    EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                    NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                                  EXCEPTION CLAUSE
 *
*****************************************************************************
                                                                  ***********************************
                                                                  /
package com.cts.creditcard.service;


import java.util.List;


import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;


public class CreditCardAdminService {


                                                                  /**
                                                                   * @param records
                                                                   * @return List<Customer>
                                                                   */
                                                                  public static List<CreditCard>
                                                                  buildMasterCreditCardList(List<String>
                                                                  records) {
                                                                           // TODO Add your logic here


                                                                           return null; // TODO change this
                                                                  return value
                                                                  }


                                                                  /**
                                                                   * @param billAmount
                                                                   * @return Double
 */
public static Double
getBillAmountWithLatePaymentCharges
(Double billAmount) {
            // TODO add your logic here
            return 0.00; // TODO change this
return value
}


/**
 * @param inputFeed
 * @return Boolean
 * @throws
CreditCardAdminSystemException
 */
public Boolean
addCreditCardDetails(String inputFeed)
throws
CreditCardAdminSystemException {
            // TODO add your logic here


            return null; //TODO change this
return value
}


}
```

ApplicationUtil

```
/***************************************************************************
                                 ********************************
 * This class ApplicationUtil is used for any utility methods needed for service or dao classes
```

```
*
* DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                        EXCEPTION CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* DO TEST YOUR CODE USING MAIN METHOD
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                        NEED BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                        EXCEPTION CLAUSE
*
***************************************************************************
                                                        ********************************
                                                        /

package com.cts.creditcard.util;


import java.util.Date;
import java.util.List;


import com.cts.creditcard.exception.CreditCardAdminSystemException;


public class ApplicationUtil {


                                                        /**
                                                         * @param fileName
                                                         * @return List<String>
                                                         * @throws
                                                        CreditCardAdminSystemException
                                                         */
```

```
public static List<String> readFile(String
fileName) throws
CreditCardAdminSystemException {

        // TODO Add your logic here


        return null; // TODO change this
return value

}


public static Date
geDateWithoutTime(Date date) {

        // TODO Add your logic here

        return null; // TODO change this
return value

}



/**

 * @param util

 *         Date

 * @return sql Date

 */

public static java.sql.Date
convertUtilToSqlDate(java.util.Date
uDate) {

        // TODO Add your logic here

        return null; // TODO change this
return value

}


/**

 * @param inDate

 * @return Date
 */
public static Date
convertStringToDate(String inDate) {
        // TODO Add your logic here
        return null; // TODO change this
return value
}


}
```

DBConnectionManager

```
/****************************************************************************
                                  ********************************
 * This class DBConnectionManager is used acquire database connection
 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                  EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                  NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                  EXCEPTION CLAUSE
 *
****************************************************************************
                                  ********************************
                                  /
package com.cts.creditcard.util;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {

                                        /**
                                         * @throws
                                        CreditCardAdminSystemException
                                         */
                                        private DBConnectionManager() throws
                                        CreditCardAdminSystemException {


                                        }


                                        /**
                                         * @return DBConnectionManager
                                         * @throws
                                        CreditCardAdminSystemException
                                         */
                                        public static DBConnectionManager
                                        getInstance() throws
                                        CreditCardAdminSystemException {
                                                // TODO Add your logic here


                                                return null; // TODO change this
                                        return value
                                        }


}
```

CreditCard

```
/**************************************************************************
                                                    ********************************
 * This class CreditCard is a value object for data transfer between Service and DAO layers
 *
 * DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY OF THE BELOW MEMBER
                                                    VARIABLES
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                    EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                    NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                    EXCEPTION CLAUSE
 *
**************************************************************************
                                                    ********************************
                                                    /
package com.cts.creditcard.vo;

import java.util.Date;

public class CreditCard {

                                                    //DO NOT CHANGE THE NAMES OR
                                                    DATA TYPES OR VISIBILITY OF THE
                                                    BELOW MEMBER VARIABLES
                                                    public Long creditCardNum;
                                                    public String customerName;
public String customerEmail;

public Long customerPhone;

public Double billAmount;

public Date dueDate;

public Date paymentDate;

//TODO add your code here

}
```

CreditCardAdminSystem

CreditCardDAO

```
/*************************************************************************
                                        *********************************

 * This class CreditCardDAO is used to persist or retrieve data from database.

 *

 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                        EXCEPTION CLAUSES, RETURN TYPES

 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS

 * DO TEST YOUR CODE USING MAIN METHOD

 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC

 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                        NEED BE,

 * YOU CAN CATCH THEM AND THROW ONLY

 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

 *
```

```
*************************************************************************
                                        ********************************
                                        /
package com.cts.creditcard.dao;


import java.sql.Connection;
import java.util.List;


import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;



public class CreditCardDAO {


                                        private static Connection conn = null;


                                        public Boolean
                                        addCreditCardDetails(List<CreditCard>
                                        cards) throws
                                        CreditCardAdminSystemException {
                                         //TODO add your code here
                                                return false;    //TODO CHANGE
                                        THIS RETURN TYPE


                                        }
}
```

<u>CreditCardAdminSystemException</u>

```
/************************************************************************
                                        ********************************
```

```
 * This class CreditCardAdminSystemException is as exception class.
 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                        EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                        NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY
 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE
 *
*************************************************************************
                                                        ********************************
                                                        /
package com.cts.creditcard.exception;


public class CreditCardAdminSystemException  extends Exception {


                                                        private static final long serialVersionUID
                                                        = -6349759544203601561L;


                                                        //TODO add your constructors here
}
```

MainApp

```
/*************************************************************************
                                                        ********************************
 * This class MainApp is used to run the service methods and to test the database.
 *
 * YOU CAN INVOKE THE METHODS AS REQUIRED FROM HERE TO TEST THE APP
 * DO NOT USE VIA COMMAND LINE ARGUMENTS FROM MAIN METHOD, UNLESS SPECIFIED
 *
*************************************************************************
                                                  *******************************
                                                  /
package com.cts.creditcard.main;


public class MainApp {
                                                  public static void main(String ag[]) {
                                                          //TODO add your code here
                                                  }
}
```

CreditCardAdminService

```
/*************************************************************************
                                                  *******************************
 * This class CreditCardAdminService is used to handle business logic for the proposed
                                                  system.
 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                  EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                  NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                  EXCEPTION CLAUSE
 *
*************************************************************************
                                            ********************************
                                            /

package com.cts.creditcard.service;


import java.util.List;


import com.cts.creditcard.exception.CreditCardAdminSystemException;

import com.cts.creditcard.vo.CreditCard;


public class CreditCardAdminService {


                                            /**
                                             * @param records
                                             * @return List<Customer>
                                             */
                                            public static List<CreditCard>
                                            buildMasterCreditCardList(List<String>
                                            records) {
                                                  // TODO Add your logic here


                                                  return null; // TODO change this
                                            return value
                                            }


                                            /**
                                             * @param billAmount
                                             * @return Double
                                             */
public static Double
getBillAmountWithLatePaymentCharges
(Double billAmount) {

        // TODO add your logic here

        return 0.00; // TODO change this
return value

}


/**

 * @param inputFeed

 * @return Boolean

 * @throws
CreditCardAdminSystemException

 */

public Boolean
addCreditCardDetails(String inputFeed)
throws
CreditCardAdminSystemException {

        // TODO add your logic here


        return null; //TODO change this
return value

}


}
```

<u>ApplicationUtil</u>

```
/*************************************************************************
                                                               **********************************

 * This class ApplicationUtil is used for any utility methods needed for service or dao classes

 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                    EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                    NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                    EXCEPTION CLAUSE
 *
************************************************************************
                                                    ********************************
                                                    /
package com.cts.creditcard.util;


import java.util.Date;
import java.util.List;


import com.cts.creditcard.exception.CreditCardAdminSystemException;


public class ApplicationUtil {


                                                    /**
                                                     * @param fileName
                                                     * @return List<String>
                                                     * @throws
                                                    CreditCardAdminSystemException
                                                     */
                                                    public static List<String> readFile(String
                                                    fileName) throws
                                                    CreditCardAdminSystemException {
```

```
            // TODO Add your logic here


            return null; // TODO change this
return value

}


public static Date
geDateWithoutTime(Date date) {

            // TODO Add your logic here

            return null; // TODO change this
return value

}



/**

 * @param util

 *          Date

 * @return sql Date

 */

public static java.sql.Date
convertUtilToSqlDate(java.util.Date
uDate) {

            // TODO Add your logic here

            return null; // TODO change this
return value

}


/**

 * @param inDate

 * @return Date

 */

public static Date
convertStringToDate(String inDate) {
```

```
                                                            // TODO Add your logic here
                                                            return null; // TODO change this
                                                      return value
                                                      }



}

/*************************************************************************
                                                      ********************************
 * This class DBConnectionManager is used acquire database connection
 *
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                      EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                      NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                      EXCEPTION CLAUSE
 *
*************************************************************************
                                                      ********************************
                                                      /
package com.cts.creditcard.util;

import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {
```

```
/**
 * @throws
CreditCardAdminSystemException
 */
private DBConnectionManager() throws
CreditCardAdminSystemException {


}


/**
 * @return DBConnectionManager
 * @throws
CreditCardAdminSystemException
 */
public static DBConnectionManager
getInstance() throws
CreditCardAdminSystemException {
            // TODO Add your logic here


            return null; // TODO change this
return value
}


}
```

CreditCard

```
/************************************************************************
                                                           ********************************
 * This class CreditCard is a value object for data transfer between Service and DAO layers
 *
 * DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY OF THE BELOW MEMBER
                                                                   VARIABLES
 * DO NOT CHANGE THE CLASS NAME,  PUBLIC METHODS, SIGNATURE OF METHODS,
                                                                   EXCEPTION CLAUSES, RETURN TYPES
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 * DO TEST YOUR CODE USING MAIN METHOD
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
                                                                   NEED BE,
 * YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
                                                                   EXCEPTION CLAUSE
 *
************************************************************************
                                                                   ********************************
                                                                   /
package com.cts.creditcard.vo;


import java.util.Date;


public class CreditCard {


                                                                   //DO NOT CHANGE THE NAMES OR
                                                                   DATA TYPES OR VISIBILITY OF THE
                                                                   BELOW MEMBER VARIABLES
                                                                   public Long creditCardNum;
                                                                   public String customerName;
                                                                   public String customerEmail;
                                                                   public Long customerPhone;
                                                                   public Double billAmount;
                                                                   public Date dueDate;
                                                                   public Date paymentDate;
//TODO add your code here

}
```

UNOAdmission_

StudentAdmissionDAO

```
package com.cts.unoadm.dao;

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

import com.cts.unoadm.exception.StudentAdmissionException;
import com.cts.unoadm.vo.StudentAdmission;

public class StudentAdmissionDAO {

	public boolean addStudentAdmissionDetails(List<StudentAdmission> stdAdmissions) throws StudentAdmissionException {
		boolean recordsAdded = false;

		//code here

		return recordsAdded;
	}
public List<StudentAdmission>
getAllStudentAdmissionDetails() throws
StudentAdmissionException {

        List<StudentAdmission>
stdAdmissions = new
ArrayList<StudentAdmission>();

        //code here

        return stdAdmissions;

}
}
```

StudentAdmissionException

```
package com.cts.unoadm.exception;

public class StudentAdmissionException extends Exception {

private static final long serialVersionUID
= -1105431869622052445L;

/**
 * @param message
 * @param cause
 */
public
StudentAdmissionException(String
message, Throwable cause) {
```

```
                    super(message, cause);
          }
}
```

MainApp

```
package com.cts.unoadm.main;

import com.cts.unoadm.skeletonvalidator.SkeletonValidator;

public class MainApp {

          public static void main(String[] args) {
                    //Don't delete this code
                    //Skeletonvalidaton starts
                    new SkeletonValidator();
                    //Skeletonvalidation ends

                    //Write your code here..


          }

}
```

StudentAdmissionService

```
package com.cts.unoadm.service;
```

```
import java.util.ArrayList;

import java.util.List;


import com.cts.unoadm.exception.StudentAdmissionException;

import com.cts.unoadm.vo.StudentAdmission;


public class StudentAdmissionService {

	/**
	 * @return List<StudentAdmission>
	 */
	public static List<StudentAdmission> buildStudentAdmissionsList(List<String> studentAdmissionRecords) {
		List<StudentAdmission> studentAdmissionList = new ArrayList<StudentAdmission>();

		//Code here

		return studentAdmissionList;
	}


	public boolean addStudentAdmissionDetails(String inputFeed) throws StudentAdmissionException {

		//Code here

		return false;
		}


		public static double[]
		calculateTotalCollegeFee(String
		preferCollegeHostel, String
		firstGraduate, String departmentName)
		{
			double[] studentAdmissionCosts
		= new double[4];


			//Code here..


			return studentAdmissionCosts;
		}


		public boolean
		searchStudentAdmission(String
		admissionId) throws
		StudentAdmissionException {
			boolean status = false;


			//Code here..


			return status;
		}
}
```

SkeletonValidator

```
package com.cts.unoadm.skeletonvalidator;
```

```
import java.lang.reflect.Array;

import java.lang.reflect.Method;

import java.util.logging.Level;

import java.util.logging.Logger;


/**
 * @author t-aarti3
 *       This class is used to verify if the Code Skeleton is intact and not
 *       modified by participants thereby ensuring smooth auto evaluation
 * */

public class SkeletonValidator {
                                        public SkeletonValidator() {

                                                validateClassName("com.cts.uno
                                        adm.util.DBConnectionManager");

                                                validateClassName("com.cts.uno
                                        adm.util.ApplicationUtil");

                                                validateClassName("com.cts.uno
                                        adm.service.StudentAdmissionService");

                                                validateClassName("com.cts.uno
                                        adm.dao.StudentAdmissionDAO");

                                                validateClassName("com.cts.uno
                                        adm.vo.StudentAdmission");

                                                validateClassName("com.cts.uno
                                        adm.exception.StudentAdmissionExcept
                                        ion");
```

```
		validateMethodSignature(

		"addStudentAdmissionDetails:bo
olean,getAllStudentAdmissionDetails:Lis
t",

		"com.cts.unoadm.dao.StudentAd
missionDAO");
		validateMethodSignature(

		"buildStudentAdmissionsList:List,
addStudentAdmissionDetails:boolean,ca
lculateTotalCollegeFee:double[],searchS
tudentAdmission:boolean",

		"com.cts.unoadm.service.Studen
tAdmissionService");
		validateMethodSignature(

		"readFile:List,convertUtilToSqlDa
te:Date,convertStringToDate:Date,check
IfValidAdmission:boolean",

		"com.cts.unoadm.util.Applicatio
nUtil");
		validateMethodSignature(

		"getConnection:Connection,getIn
stance:DBConnectionManager",

		"com.cts.unoadm.util.DBConnect
ionManager");




}
```

```
private static final Logger LOG =
Logger.getLogger("SkeletonValidator");

protected final boolean
validateClassName(String className) {


        boolean iscorrect = false;

        try {


        Class.forName(className);

                iscorrect = true;

                LOG.info("Class Name " +
className + " is correct");


        } catch (ClassNotFoundException
e) {

                LOG.log(Level.SEVERE,
"You have changed either the " + "class
name/package. Use the correct package
"

                                + "and
class name as provided in the
skeleton");


        } catch (Exception e) {

                LOG.log(Level.SEVERE,

                                "There is
an error in validating the " + "Class
Name. Please manually verify that the "


        + "Class name is same as
skeleton before uploading");

        }

        return iscorrect;

}
```

```
protected final void
validateMethodSignature(String
methodWithExcptn, String className) {

        Class cls = null;

        try {


                String[] actualmethods =
methodWithExcptn.split(",");

                boolean errorFlag = false;

                String[] methodSignature;

                String methodName =
null;

                String returnType = null;


                for (String singleMethod :
actualmethods) {

                        boolean
foundMethod = false;

                        methodSignature
= singleMethod.split(":");


                        methodName =
methodSignature[0];

                        returnType =
methodSignature[1];

                        cls =
Class.forName(className);

                        Method[]
methods = cls.getMethods();

                        for (Method
findMethod : methods) {
```

```
                                        if
(methodName.equals(findMethod.getN
ame())) {

          foundMethod = true;

                                                  if
(!(findMethod.getReturnType().getSimpl
eName().equals(returnType))) {

          errorFlag = true;

          LOG.log(Level.SEVERE, " You
have changed the " + "return type in '" +
methodName

                              + "' method.
Please stick to the " + "skeleton
provided");


                                                  }
else {

          LOG.info("Method signature of "
+ methodName + " is valid");
                                                  }


                                        }
                              }
                              if (!foundMethod)
{
                                        errorFlag =
true;

          LOG.log(Level.SEVERE, " Unable
to find the given public method " +
methodName
```

```
                + ". Do not change the " + "given
public method name. " + "Verify it with
the skeleton");

                                }


                        }

                        if (!errorFlag) {

                                LOG.info("Method
signature is valid");

                        }


                } catch (Exception e) {

                        LOG.log(Level.SEVERE,

                                        " There is
an error in validating the " + "method
structure. Please manually verify that
the "


                + "Method signature is same as
the skeleton before uploading");

                }

        }


}
```

ApplicationUtil

```
package com.cts.unoadm.util;



import java.util.ArrayList;
import java.util.Date;

import java.util.List;


import com.cts.unoadm.exception.StudentAdmissionException;


public class ApplicationUtil {

	/**
	 * @param fileName
	 * @return List<String>
	 * @throws StudentAdmissionException
	 */
	public static List<String> readFile(String fileName) throws StudentAdmissionException {
		List<String> studentAdmissionList = new ArrayList<String>();
		  //Code here..


		return studentAdmissionList;
	}


	/**
	 * @param util
	 *          Date
	 * @return sql Date
	 */
	public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate) {
		java.sql.Date sDate = null;

		//Code here..

		return sDate;
	}

	/**
	 * @param inDate
	 * @return Date
	 */
	public static Date
	convertStringToDate(String inDate) {

		//Code here..

		return new Date();//TODO
	change this return value
	}

	public static boolean
	checkIfValidAdmission(Date
	dtOfCounseling, Date dtOfAdmission,
	String manager) {
		boolean admissionValidity =
	false;

		//Code here..

		return admissionValidity;
```

```
                                        }
}
```

DBConnectionManager

```
/**
 * Don't change this code
 */
package com.cts.unoadm.util;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.cts.unoadm.exception.StudentAdmissionException;


public class DBConnectionManager {

                                        public static final String PROPERTY_FILE
                                        = "database.properties";
                                        public static final String DRIVER =
                                        "drivername";
                                        public static final String URL = "url";
                                        public static final String USER_NAME =
                                        "username";
```

```
public static final String PASSWORD =
"password";


private static Connection connection =
null;

private static Properties props = null;


/**

 * @throws StudentAdmissionException

 */

private DBConnectionManager() throws
StudentAdmissionException {

        loadProperties();

        try {


        Class.forName(props.getProperty
(DRIVER));

                this.connection =
DriverManager.getConnection(props.get
Property(URL),
props.getProperty(USER_NAME),


        props.getProperty(PASSWORD));

        } catch (ClassNotFoundException
ex) {


                throw new
StudentAdmissionException("Could not
find Driver class ", ex.getCause());

        } catch (SQLException e) {

                throw new
StudentAdmissionException("Database
Connection Creation Failed",
e.getCause());

        }
```

```
}

/**
 * @return Connection
 */
public Connection getConnection() {
	return connection;
}

/**
 * @return DBConnectionManager
 * @throws StudentAdmissionException
 */
public static DBConnectionManager
getInstance() throws
StudentAdmissionException {

	// Code here

	return null;
}

/**
 * @throws StudentAdmissionException
 */
private void loadProperties() throws
StudentAdmissionException {
	FileInputStream inputStream =
null;
	try {
			inputStream = new
FileInputStream(PROPERTY_FILE);

			props = new Properties();

			props.load(inputStream);

		} catch (FileNotFoundException
e) {


			throw new
StudentAdmissionException("Database
Property File Not Found", e.getCause());

		} catch (IOException e) {

			throw new
StudentAdmissionException("Exception
during property file I/O", e.getCause());

		} finally {

			if (inputStream != null) {

				try {


		inputStream.close();

				} catch
(IOException e) {

					throw new
StudentAdmissionException("Exception
during property file I/O", e.getCause());

				}

			}

		}

	}

}
```

StudentAdmission

```
/*
 * Don't change this code
 */
package com.cts.unoadm.vo;

import java.util.Date;

public class StudentAdmission {
                                        String admissionId;
                                        String studentCode;
                                        Date dateOfCounseling;
                                        String departmentName;
                                        Date dateOfAdmission;
                                        String preferCollegeHostel;
                                        String firstGraduate;
                                        String managerApproval;
                                        double admissionFee;
                                        double tuitionFee;
                                        double hostelFee;
                                        double totalCollegeFee;
                                        String finalStatusOfAdmission;

                                        public StudentAdmission() {
                                                super();
                                        }

                                        public StudentAdmission(String
                                        admissionId, String studentCode, Date
dateOfCounseling, String
departmentName,

                    Date dateOfAdmission,
String preferCollegeHostel, String
firstGraduate, String managerApproval,

                    double admissionFee,
double tuitionFee, double hostelFee,
double totalCollegeFee,

                    String
finalStatusOfAdmission) {

          super();

          this.admissionId = admissionId;

          this.studentCode = studentCode;

          this.dateOfCounseling =
dateOfCounseling;

          this.departmentName =
departmentName;

          this.dateOfAdmission =
dateOfAdmission;

          this.preferCollegeHostel =
preferCollegeHostel;

          this.firstGraduate =
firstGraduate;

          this.managerApproval =
managerApproval;

          this.admissionFee =
admissionFee;

          this.tuitionFee = tuitionFee;

          this.hostelFee = hostelFee;

          this.totalCollegeFee =
totalCollegeFee;

          this.finalStatusOfAdmission =
finalStatusOfAdmission;

}
```

```
public String getAdmissionId() {
        return admissionId;
}

public void setAdmissionId(String
admissionId) {
        this.admissionId = admissionId;
}

public String getStudentCode() {
        return studentCode;
}

public void setStudentCode(String
studentCode) {
        this.studentCode = studentCode;
}

public Date getDateOfCounseling() {
        return dateOfCounseling;
}

public void setDateOfCounseling(Date
dateOfCounseling) {
        this.dateOfCounseling =
dateOfCounseling;
}

public String getDepartmentName() {
        return departmentName;
}

public void setDepartmentName(String
departmentName) {
            this.departmentName =
departmentName;
}

public Date getDateOfAdmission() {
            return dateOfAdmission;
}

public void setDateOfAdmission(Date
dateOfAdmission) {
            this.dateOfAdmission =
dateOfAdmission;
}

public String getPreferCollegeHostel() {
            return preferCollegeHostel;
}

public void
setPreferCollegeHostel(String
preferCollegeHostel) {
            this.preferCollegeHostel =
preferCollegeHostel;
}

public String getFirstGraduate() {
            return firstGraduate;
}
```

```
}

public void setFirstGraduate(String
firstGraduate) {
            this.firstGraduate =
firstGraduate;
}

public String getManagerApproval() {
            return managerApproval;
}

public void setManagerApproval(String
managerApproval) {
            this.managerApproval =
managerApproval;
}

public double getAdmissionFee() {
            return admissionFee;
}

public void setAdmissionFee(double
admissionFee) {
            this.admissionFee =
admissionFee;
}

public double getTuitionFee() {
            return tuitionFee;
}
```

```
public void setTuitionFee(double
tuitionFee) {
	this.tuitionFee = tuitionFee;
}

public double getHostelFee() {
	return hostelFee;
}

public void setHostelFee(double
hostelFee) {
	this.hostelFee = hostelFee;
}

public double getTotalCollegeFee() {
	return totalCollegeFee;
}

public void setTotalCollegeFee(double
totalCollegeFee) {
	this.totalCollegeFee =
totalCollegeFee;
}

public String
getFinalStatusOfAdmission() {
	return finalStatusOfAdmission;
}

public void
setFinalStatusOfAdmission(String
finalStatusOfAdmission) {
		this.finalStatusOfAdmission = finalStatusOfAdmission;
	}

	@Override
	public String toString() {
		return "Student Admission Details: [admissionId=" + admissionId + ", studentCode=" + studentCode + ", dateOfCounseling="
				+ dateOfCounseling + ", departmentName=" + departmentName + ", dateOfAdmission=" + dateOfAdmission + ", preferCollegeHostel="
				+ preferCollegeHostel + ", firstGraduate=" + firstGraduate + ", managerApproval=" + managerApproval
				+ ", admissionFee=" + admissionFee + ", tuitionFee=" + tuitionFee + ", hostelFee=" + hostelFee + ", totalCollegeFee=" + totalCollegeFee
				+ ", finalStatusOfAdmission=" + finalStatusOfAdmission + "]";
	}

}
```

ConstructionCostTimeEstimate

CostAndTimeEstimation

```
package com.cts.conctes.client;

import com.cts.conctes.service.ConstructionProjectEstimationService;

public class CostAndTimeEstimation {

                                        public static void main(String[] args)
                                        {

                                                ConstructionProjectEstimationSe
                                        rvice cpeService = new
                                        ConstructionProjectEstimationService();
                                                //WRITE YOUR CODE HERE

                                        }
}
```

CostAndTimeEstDAO

```
package com.cts.conctes.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import com.cts.conctes.exception.ConstructionEstimationException;

import com.cts.conctes.model.ConstructionProject;

import com.cts.conctes.util.ApplicationUtil;




public class CostAndTimeEstDAO {

        public static Connection connection =
        null;


        public boolean
        insertConstructionProject(ArrayList
        <ConstructionProject> constProjects)
        throws
        ConstructionEstimationException {

                boolean recordsAdded = false;


                //WRITE YOUR CODE HERE


                return recordsAdded;

        }

        public ArrayList <ConstructionProject>
        getConstructionProjectsData()

        {

                ArrayList <ConstructionProject>
        consApplicants = new
        ArrayList<ConstructionProject>();


                //WRITE YOUR CODE HERE


                return consApplicants;
```

```
                                            }


}
```

DBConnectionManager

```
package com.cts.conctes.dao;


import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;


import com.cts.conctes.exception.ConstructionEstimationException;




public class DBConnectionManager {

                                             private static Connection con = null;

                                             private static DBConnectionManager
                                            instance;


                                            public  DBConnectionManager()  throws
                                            ConstructionEstimationException

                                            {
```

```
                                        //WRITE YOUR CODE HERE
    //return con;
                                    }
                                    public static DBConnectionManager
                                    getInstance() throws
                                    ConstructionEstimationException
                                    {

                                        //WRITE YOUR CODE HERE
                                        return instance;
                                    }
                                    public Connection getConnection()
                                    {
                                        //WRITE YOUR CODE HERE
                                        return con;
                                    }
}
```

ConstructionEstimationException

```
package com.cts.conctes.exception;

public class ConstructionEstimationException extends Exception{

                                    String strMsg1;
                                    Throwable strMsg2;
```

```
                                        public
                                        ConstructionEstimationException() {
                                                    super();
                                        }

}
```

ConstructionProject

```
package com.cts.conctes.model;

import java.util.Date;

public class ConstructionProject {

                                        String projectId;
                                        Date plannedDOStart;
                                        String typeOfProject;
                                        String structure;
                                        double areaInSqFt;
                                        double estimatedCostInlac;
                                        double estimatedTimeInMonths;

                                        public ConstructionProject() {
                                                    super();
                                        }
```

```
public ConstructionProject(String
projectId, Date plannedDOStart, String
typeOfProject, String structure,

                    double areaInSqFt,
double estimatedCostInlac, double
estimatedTimeInMonths) {

          super();

          this.projectId = projectId;

          this.plannedDOStart =
plannedDOStart;

          this.typeOfProject =
typeOfProject;

          this.structure = structure;

          this.areaInSqFt = areaInSqFt;

          this.estimatedCostInlac =
estimatedCostInlac;

          this.estimatedTimeInMonths =
estimatedTimeInMonths;

}


public String getProjectId() {

          return projectId;

}


public void setProjectId(String projectId)
{

          this.projectId = projectId;

}


public Date getPlannedDOStart() {

          return plannedDOStart;
}

public void setPlannedDOStart(Date
plannedDOStart) {
            this.plannedDOStart =
plannedDOStart;
}

public String getTypeOfProject() {
            return typeOfProject;
}

public void setTypeOfProject(String
typeOfProject) {
            this.typeOfProject =
typeOfProject;
}

public String getStructure() {
            return structure;
}

public void setStructure(String structure)
{
            this.structure = structure;
}

public double getAreaInSqFt() {
            return areaInSqFt;
}
```

```
public void setAreaInSqFt(double
areaInSqFt) {
            this.areaInSqFt = areaInSqFt;
}


public double getEstimatedCostInlac() {
            return estimatedCostInlac;
}


public void
setEstimatedCostInlac(double
estimatedCostInlac) {
            this.estimatedCostInlac =
estimatedCostInlac;
}


public double
getEstimatedTimeInMonths() {
            return estimatedTimeInMonths;
}


public void
setEstimatedTimeInMonths(double
estimatedTimeInMonths) {
            this.estimatedTimeInMonths =
estimatedTimeInMonths;
}


@Override
public String toString() {
            return "ConstructionProject
[projectId=" + projectId + ",
                                                        plannedDOStart=" + plannedDOStart + ",
                                                        typeOfProject="

                                                                          + typeOfProject +
                                                        ", structure=" + structure + ",
                                                        areaInSqFt=" + areaInSqFt + ",
                                                        estimatedCostInlac="

                                                                          +
                                                        estimatedCostInlac + ",
                                                        estimatedTimeInMonths=" +
                                                        estimatedTimeInMonths + "]";

                                                        }




}
```

ConstructionProjectEstimationService

```
package com.cts.conctes.service;


import java.util.ArrayList;

import java.util.Date;

import java.util.List;


import com.cts.conctes.dao.CostAndTimeEstDAO;

import com.cts.conctes.exception.ConstructionEstimationException;

import com.cts.conctes.model.ConstructionProject;

import com.cts.conctes.util.ApplicationUtil;
```

```
public class ConstructionProjectEstimationService {

public static ArrayList
<ConstructionProject>
buildConstructionProjectList(List
<String> consProjectRecords) {

            final String COMMADELIMITER =
",";
            ArrayList <ConstructionProject>
consProjectRecordList = new
ArrayList<ConstructionProject>();

            //WRITE YOUR CODE HERE

            return consProjectRecordList;
}

public boolean
addConstructionProjectDetails(String
inputFeed) throws
ConstructionEstimationException {

            //WRITE YOUR CODE HERE
 return false;
}

public static double[]
estimateTimeAndCostForConstruction(S
tring projectType,String
structure,double areaInSqFt)
{

        double
costEstimateInRs=0.0,timeEstimateInMo
nths=0.0;
        double costs[] =
{costEstimateInRs,timeEstimateInMonth
s};
        /*
         * The Cost Estimate and
         *
        Based on the type of the Project
& the Structure , according to the
required
        area of Construction, the cost &
time have to be calculated based on the
base
        data available in the table
provided in the use case document:
        For eg. If the Project Type is
“Commercial”  and the structure
        is “Shopping Complex” the cost
incurred for the construction of
        per sq. ft is Rs.2600 and the time
taken for the construction of
        the 1000 sq ft of the same
project  is 0.23 Months,
        calculation has to be performed
on the similar basis
        i.e Pro rata basis depending upon
the type and the area of construction.

         */
```

```
                                        //WRITE YOUR CODE HERE

                                        return costs;

                                    }

}
```

SkeletonValidator

```
package com.cts.conctes.skeleton;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.cts.conctes.model.ConstructionProject;



/**
 * @author 222805
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants
                                    thereby ensuring smooth auto
                                    evaluation
 *
 */
public class SkeletonValidator {

	public SkeletonValidator() {

		validateClassName("com.cts.conctes.model.ConstructionProject");

		validateClassName("com.cts.conctes.dao.CostAndTimeEstDAO");

		validateClassName("com.cts.conctes.dao.DBConnectionManager");

		validateClassName("com.cts.conctes.exception.ConstructionEstimationException");

		validateClassName("com.cts.conctes.service.ConstructionProjectEstimationService");

		validateClassName("com.cts.conctes.util.ApplicationUtil");


		validateMethodSignature("insertConstructionProject:boolean","com.cts.conctes.dao.CostAndTimeEstDAO");

		validateMethodSignature("getInstance:DBConnectionManager","com.cts.conctes.dao.DBConnectionManager");

		validateMethodSignature("getCo
nnection:Connection","com.cts.conctes.
dao.DBConnectionManager");

            validateMethodSignature("build
ConstructionProjectList:ArrayList,addCo
nstructionProjectDetails:boolean,estima
teTimeAndCostForConstruction:double[]
","com.cts.conctes.service.Construction
ProjectEstimationService");



}

private static final Logger LOG =
Logger.getLogger("SkeletonValidator");

protected final boolean
validateClassName(String className) {

            boolean iscorrect = false;
            try {

            Class.forName(className);
                        iscorrect = true;
                        LOG.info("Class Name " +
className + " is correct");

            } catch (ClassNotFoundException
e) {
                        LOG.log(Level.SEVERE,
"You have changed either the " + "class
name/package. Use the correct package
"
```

```
                                        + "and
class name as provided in the
skeleton");


            } catch (Exception e) {

                        LOG.log(Level.SEVERE,

                                        "There is
an error in validating the " + "Class
Name. Please manually verify that the "


            + "Class name is same as
skeleton before uploading");

            }

            return iscorrect;


}


protected final void
validateMethodSignature(String
methodWithExcptn, String className) {

            Class cls = null;

            try {


                        String[] actualmethods =
methodWithExcptn.split(",");

                        boolean errorFlag = false;

                        String[] methodSignature;

                        String methodName =
null;

                        String returnType = null;


                        for (String singleMethod :
actualmethods) {
```

```
                    boolean
foundMethod = false;

                    methodSignature
= singleMethod.split(":");


                    methodName =
methodSignature[0];

                    returnType =
methodSignature[1];

                    cls =
Class.forName(className);

                    Method[]
methods = cls.getMethods();

                    for (Method
findMethod : methods) {

                           if
(methodName.equals(findMethod.getN
ame())) {

       foundMethod = true;

                                  if
(!(findMethod.getReturnType().getSimpl
eName().equals(returnType))) {

       errorFlag = true;

       LOG.log(Level.SEVERE, " You
have changed the " + "return type in '" +
methodName

                    + "' method.
Please stick to the " + "skeleton
provided");

                                  }
else {
```

```
            LOG.info("Method signature of "
+ methodName + " is valid");
                                                    }


                                        }

                                    }

                                    if (!foundMethod)
{

                                        errorFlag =
true;


            LOG.log(Level.SEVERE, " Unable
to find the given public method " +
methodName


            + ". Do not change the " + "given
public method name. " + "Verify it with
the skeleton");

                                    }


                        }

                        if (!errorFlag) {

                                    LOG.info("Method
signature is valid");

                        }


            } catch (Exception e) {

                        LOG.log(Level.SEVERE,

                                        " There is
an error in validating the " + "method
structure. Please manually verify that
the "
```

```
                                                                    + "Method signature is same as
                                                        the skeleton before uploading");
                                                                    }
                                                        }

}
```

ApplicationUtil

```
package com.cts.conctes.util;


import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.StringTokenizer;

import com.cts.conctes.exception.ConstructionEstimationException;




public class ApplicationUtil {
}

public static List<String> readFile(String
inputfeed) throws
ConstructionEstimationException {
        List<String> constructionProjects
= new ArrayList<String>();


        //WRITE YOUR CODE HERE


        return constructionProjects;
}
public static java.sql.Date
utilToSqlDateConverter(java.util.Date
utDate) {
        java.sql.Date sqlDate = null;


        //WRITE YOUR CODE HERE


        return sqlDate;
}


public static java.util.Date
stringToDateConverter(String
stringDate) {
        Date strDate = new Date();


        //WRITE YOUR CODE HERE


        return strDate;
}
public static boolean
checkIfCurrentFinYearProject(Date dos)
```

```
{
        boolean flag = false;
        int givenYear,givenMonth;
        givenYear =
(dos.getYear()+1900);
        givenMonth = dos.getMonth();
        Date curDate = new Date();
        int curYear,curMonth;
        curYear =
(curDate.getYear()+1900);
        curMonth = curDate.getMonth();
        if( curYear == givenYear)
        {
                if(((curMonth
>=0)&&(curMonth <= 2)) &&
((givenMonth >=0)&&(givenMonth <=
2)))
                {
                        flag = true;
                }
                else if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=3)&&(givenMonth <=
11)))
                {
                        flag = true;
                }
                else
                {
                        flag = false;
                }
        }
```

```
		else if(curYear > givenYear)
		{
			int dif = curYear - 
givenYear;
			if(dif == 1)
			{
				if(((curMonth 
>=0)&&(curMonth <= 2)) && 
((givenMonth >=3)&&(givenMonth <= 
11)))
				{
					flag = true;
				}
				else if(((curMonth 
>=3)&&(curMonth <= 11)) && 
((givenMonth >=3)&&(givenMonth <= 
11)))
				{
					flag = false;
				}
				else
				{
					flag = false;
				}
			}
			else
			{
				flag = false;
			}
		}
		else if(curYear < givenYear)
		{
```

```
                    int dif =  givenYear-
curYear;
                    if(dif == 1)
                    {
                              if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=0)&&(givenMonth <=
2)))
                              {
                                        flag = true;
                              }
                              else if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=3)&&(givenMonth <=
11)))
                              {
                                        flag = false;
                              }
                              else
                              {
                                        flag = false;
                              }
                    }
                    else
                    {
                              flag = false;
                    }
          }
          else
          {
                    flag = false;
          }
```

```
                                        return flag;

                                }

}
```

Insurance

CollectionAgency

```
import java.io.*;
import java.sql.*;
import java.util.*;

public class CollectionAgency
{
                                //write the required business logic
                                methods as expected in the question
                                description

                                public List<Payment>
                                generatePaymentAmount(String
                                filePath)
                                {
                                        // fill your code here
                                }
```

```
                                        public boolean validate(String policyId)
                                        throws InvalidPolicyIdException

                                        {

                                                  // fill your code here


                                        }


                                        public void updatePolicyDetails(List
                                        <Payment> paymentList)

                                        {

                                           // fill your code here

                                        }

}
```

DBHandler

```
import java.io.*;

import java.sql.*;

import java.util.*;


public class DBHandler {


//write the required business logic methods as expected in the question description

public Connection establishConnection()

{

   // fill your code here


}
}

InvalidPolicyIdException

//make the required changes to this class so that InvalidPolicyIdException is of type
                                                                    exception.
public class InvalidPolicyIdException
{
                                                                    //fill your code here

}
```

Main

```
import java.util.*;


public class Main
{

  public static void main(String[] args)
  {
                                                                      // fill your code here

  }
}


Payment
```

```
}

public class Payment
{
        private String policyId;
        private double monthlyPremium;
        private int noOfMonths;
        private double paymentAmount;
        public String getPolicyId() {
                return policyId;
        }
        public void setPolicyId(String policyId) {
                this.policyId = policyId;
        }
        public double getMonthlyPremium() {
                return monthlyPremium;
        }
        public void setMonthlyPremium(double
        monthlyPremium) {
                this.monthlyPremium =
        monthlyPremium;
        }
        public int getNoOfMonths() {
                return noOfMonths;
        }
        public void setNoOfMonths(int
        noOfMonths) {
                this.noOfMonths = noOfMonths;
        }
        public double getPaymentAmount() {
                return paymentAmount;
        }
        public void setPaymentAmount(double
        paymentAmount) {
                this.paymentAmount =
        paymentAmount;
        }
        //Write the required business logic as
        expected in the question description
        public void calculatePaymentAmount()
        {
                //fill your code here
        }

}
```