CostAndTimeEstimation.java

```
package com.cts.conctes.client;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.service.ConstructionProjectEstimationService;
public class CostAndTimeEstimation {
public static void main(String[] args) throws ConstructionEstimationException
{
ConstructionProjectEstimationService cpeService = new
ConstructionProjectEstimationService();
boolean isTrue=cpeService.addConstructionProjectDetails("inputfeed.txt");
if(isTrue) {
        System.out.println("All are added successfully into the database");
}
}
}
```

CostAndTimeEstDAO.java

```
package com.cts.conctes.dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
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;
int index=0;
int size=constProjects.size();
connection=DBConnectionManager.getInstance().getConnection();
```

```
String query="insert into constructionproject values(?, ?, ?, ?, ?, ?, ?);";
for(ConstructionProject cp: constProjects) {

	try {
		PreparedStatement ps=connection.prepareStatement(query);
		ps.setString(1, cp.getProjectId());
		ps.setDate(2,
ApplicationUtil.utilToSqlDateConverter(cp.getPlannedDOStart()));
		ps.setString(3, cp.getTypeOfProject());
		ps.setString(4, cp.getStructure());
		ps.setDouble(5, cp.getAreaInSqFt());
		ps.setDouble(6, cp.getEstimatedCostInlac());
		ps.setDouble(7, cp.getEstimatedTimeInMonths());
		int row=ps.executeUpdate();
		if(row>0)
			index+=1;
		if(index==size) {
			recordsAdded=true;
		}
	} catch (SQLException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}




}
return recordsAdded;


}
public ArrayList <ConstructionProject> getConstructionProjectsData() throws
ConstructionEstimationException
{
ArrayList <ConstructionProject> consApplicants = new
ArrayList<ConstructionProject>();
connection=DBConnectionManager.getInstance().getConnection();
```

```
String query="select * from constructionproject";
try {
	PreparedStatement ps=connection.prepareStatement(query);
	ResultSet rs=ps.executeQuery();
	while(rs.next()) {
		String id=rs.getString(1);
		Date d=rs.getDate(2);
		String typeProject=rs.getString(3);
		String structure=rs.getString(4);
		double areaSqFt=rs.getDouble(5);
		double costInLac=rs.getDouble(6);
		double timeInMonths=rs.getDouble(7);
		consApplicants.add(new ConstructionProject(id, d, typeProject,
structure, areaSqFt, costInLac, timeInMonths));
	}
} catch (SQLException e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
}

return consApplicants;
}
}
```

DBConnectionManager.java

```
package com.cts.conctes.dao;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
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;
private DBConnectionManager() throws ConstructionEstimationException
{
//WRITE YOUR CODE HERE
//return con;
}
public static DBConnectionManager getInstance() throws
ConstructionEstimationException
{
	if(instance==null) {
		instance=new DBConnectionManager();
	}
	return instance;
}
public Connection getConnection()
{
	try {
		Class.forName("com.mysql.cj.jdbc.Driver");
	} catch (ClassNotFoundException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	Properties properties=new Properties();
	FileReader f=null;
	try {
		f = new FileReader("database.properties");
	} catch (FileNotFoundException e2) {
		// TODO Auto-generated catch block
		e2.printStackTrace();
	}
	try {
		properties.load(f);
	} catch (IOException e1) {
		// TODO Auto-generated catch block
```

```
                e1.printStackTrace();
        }
        String url=properties.getProperty("url");
        String user=properties.getProperty("username");
        String password=properties.getProperty("password");
        try {
                con=DriverManager.getConnection(url, user, password);
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return con;
}
}
```

ConstructionEstimationException.java

```
package com.cts.conctes.exception;
public class ConstructionEstimationException extends Exception{
String strMsg1;
Throwable strMsg2;
public ConstructionEstimationException() {
super();
}
}
```

ConstructionProject.java

```
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.java

```
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>();
        for(String s: consProjectRecords) {
                String[] s1=s.split(COMMADELIMITER);
                String id=s1[0];
                Date d=ApplicationUtil.stringToDateConverter(s1[1]);
                String typeproject=s1[2];
                String structure=s1[3];
                double areaInSqFt=Double.parseDouble(s1[4]);
                double costs[]=estimateTimeAndCostForConstruction(typeproject,
structure, areaInSqFt);
                double estimatedCostInLac=costs[0];
                double estimatedTimeInMon=costs[1];
                ConstructionProject p=new ConstructionProject(id, d, typeproject,
structure, areaInSqFt,  estimatedCostInLac, estimatedTimeInMon);
                consProjectRecordList.add(p);
        }
        return consProjectRecordList;
}
public boolean addConstructionProjectDetails(String inputFeed) throws
        ConstructionEstimationException {
        ArrayList<ConstructionProject>
p=buildConstructionProjectList(ApplicationUtil.readFile(inputFeed));
        CostAndTimeEstDAO obj=new CostAndTimeEstDAO();
        if(obj.insertConstructionProject(p)) {
                ArrayList<ConstructionProject>
p1=obj.getConstructionProjectsData();
                for(ConstructionProject cp: p1) {
```

```
                        System.out.println(cp);
                }
                return true;
        }
        return false;
}
public static double[] estimateTimeAndCostForConstruction(String
projectType,String
structure,double areaInSqFt)
{
        double costEstimateInRs=0.0,timeEstimateInMonths=0.0;
        double costs[] = {costEstimateInRs,timeEstimateInMonths};
/*
* 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.
*/
        if(projectType.equals("Commercial")) {
                if(structure.equals("Shopping Complex")) {
                        costs[0]=2600*areaInSqFt;
                        costs[1]=0.23*areaInSqFt/1000;
                }
                else if(structure.equals("ResApartments")) {
                        costs[0]=2750*areaInSqFt;
                        costs[1]=0.24*areaInSqFt/1000;
                }
                else {
```

```
                costs[0]=2600*areaInSqFt;
                costs[1]=0.2*areaInSqFt/1000;
            }
        }
        else if(projectType.equals("Infrastructural")) {
            if(structure.equals("Bridge")) {
                costs[0]=10000*areaInSqFt;
                costs[1]=0.25*areaInSqFt/1000;
            }
            else if(structure.equals("FlyOver")) {
                costs[0]=14000*areaInSqFt;
                costs[1]=0.22*areaInSqFt/1000;
            }
            else {
                costs[0]=8000*areaInSqFt;
                costs[1]=(0.25/1000)*areaInSqFt;
            }
        }
        else {
            if(structure.equals("House")) {
                costs[0]=2250*areaInSqFt;
                costs[1]=0.26*areaInSqFt/1000;
            }
            else if(structure.equals("Apartments")) {
                costs[0]=2500*areaInSqFt;
                costs[1]=0.24*areaInSqFt/1000;
            }
            else {
                costs[0]=2750*areaInSqFt;
                costs[1]=0.23*areaInSqFt/1000;
            }
        }
        return costs;
    }
}
```

ApplicationUtil.java

```
package com.cts.conctes.util;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
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>();
	BufferedReader br=null;
	try {
		br = new BufferedReader(new FileReader(inputfeed));

	} catch (FileNotFoundException e1) {
		// TODO Auto-generated catch block
		e1.printStackTrace();
	}
	String s=null;
	try {
		s = br.readLine();
		while(s!=null) {
			String s1[]=s.split(",");

	if(checkIfCurrentFinYearProject(stringToDateConverter(s1[1])))
				constructionProjects.add(s);
```

```
                        s=br.readLine();
                }
        } catch (IOException e) {
                e.printStackTrace();
        }
        return constructionProjects;
}
public static java.sql.Date utilToSqlDateConverter(java.util.Date utDate) {
        java.sql.Date sqlDate;
        String s=new SimpleDateFormat("yyyy-MM-dd").format(utDate);
        sqlDate=java.sql.Date.valueOf(s);
        return sqlDate;
}
public static java.util.Date stringToDateConverter(String stringDate)  {
        Date strDate = null;
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
        try {
                strDate=sdf.parse(stringDate);
        } catch (ParseException e) {
                e.printStackTrace();
        }
        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;
}
}
```