EmployeeDetailsReport/database.properties

1 #IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE

2 #ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY

3 #YOU CAN CHANGE THE VALUE OF THE PROPERTY

4 #LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD using this properties file only.

5 #Do not hard code the values

6

7 DB\_DRIVER\_CLASS=oracle.jdbc.driver.OracleDriver

8 DB\_URL=jdbc:oracle:thin:@127.0.0.1:1521:XE

9 DB\_USERNAME=${sys:db\_username}

10 DB\_PASSWORD=${sys:db\_password}

EmployeeDetailsReport/inputfeed.txt

1 PR47856,Steven,level1,24

2 TR87965,Sujatha,level1,12

3 PR78965,Seetha,level3,12

4 PR7895,Vaishnav,level2,6

5 PR78995,Yamuna,level2,25

6 PR78999,John,level4,13

7 TR88888,Wilsy,level4,16

8 PR12356,Nelson,level1,27

9 PR85203,Mithun,level3,11

EmployeeDetailsReport/src/com/cts/employeedetailsreport/client/EmployeeDetailsMain.java

1 package com.cts.employeedetailsreport.client;

2 import com.cts.employeedetailsreport.skeleton.SkeletonValidator;

3

4 public class EmployeeDetailsMain {

5

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

7 // CODE SKELETON - VALIDATION STARTS

8 // DO NOT CHANGE THIS CODE

9

10 new SkeletonValidator();

11

12 // CODE SKELETON - VALIDATION ENDS

13 // new HospitalManagement();

14

15 // TYPE YOUR CODE HERE

16

17 }

18

19 }

20

21

EmployeeDetailsReport/src/com/cts/employeedetailsreport/dao/DBConnectionManager.java

1 package com.cts.employeedetailsreport.dao;

2

3

4 import java.io.FileInputStream;

5 import java.sql.Connection;

6 import java.io.IOException;

7 import java.sql.DriverManager;

8 import java.util.Properties;

9

10

11 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;

12

13

14 public class DBConnectionManager {

15

16

17

18 private static Connection con = null;

19 private static DBConnectionManager instance;

20 public DBConnectionManager() throws InvalidEmployeeNumberException

21 {

22 FileInputStream fis=null;

23

24 try{

25 fis=new FileInputStream("database.properties");

26 Properties props=new Properties();

27 props.load(fis);

28 Class.forName(props.getProperty("DB\_DRIVER\_CLASS"));

29 con=DriverManager.getConnection(props.getProperty("DB\_URL"),props.getProperty("DB\_USERNAME"),props.getProperty("DB\_PASSWORD"));

30 } catch(Exception e){

31 e.printStackTrace();

32 }

33

34 //FILL THE CODE HERE

35 }

36 public static DBConnectionManager getInstance() throws InvalidEmployeeNumberException {

37 //FILL THE CODE HERE

38 instance=new DBConnectionManager();

39 return instance;

40 }

41 public Connection getConnection() {

42 return con;

43 }

44 }

45

46

47

48

49

50

51

52

53

54

55

EmployeeDetailsReport/src/com/cts/employeedetailsreport/dao/DetailsDAO.java

1 package com.cts.employeedetailsreport.dao;

2

3 import java.sql.Connection;

4 import java.sql.Statement;

5 import java.sql.SQLException;

6 import java.util.List;

7 import java.sql.\*;

8

9 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;

10 import com.cts.employeedetailsreport.model.EmployeeDetails;

11

12 public class DetailsDAO {

13

14

15 public boolean insertEmployeeList(List <EmployeeDetails> eList) throws InvalidEmployeeNumberException {

16

17 boolean recordsAdded = false;

18 DBConnectionManager obj=DBConnectionManager.getInstance();

19 Connection conne=obj.getConnection();

20 try{

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

22 String ins="insert into EmployeeDetails (employee\_Number,name,levelstatus,extra\_workinghours,total\_salary) values (?,?,?,?,?)";

23 PreparedStatement ps=conne.prepareStatement(ins);

24 ps.setString(1,eList.get(i).getEmployeeNumber());

25 ps.setString(2,eList.get(i).getEmployeeName());

26 ps.setString(3,eList.get(i).getLevel());

27 ps.setInt(4,eList.get(i).getExtraWorkingHours());

28 ps.setDouble(5,eList.get(i).getTotalSalary());

29 ps.execute();

30 }

31 recordsAdded=true;

32 }

33 catch(SQLException e)

34 {

35 e.printStackTrace();

36 try{

37 conne.rollback();

38 }

39 catch(Exception k)

40 {

41 k.printStackTrace();

42 }finally{

43 try{

44 conne.close();

45 }catch(Exception k){

46 k.printStackTrace();

47 }

48 }

49 }

50

51 // FILL THE CODE HERE

52

53

54 return recordsAdded;

55 }

56

57

58

59

60 }

61

EmployeeDetailsReport/src/com/cts/employeedetailsreport/exception/InvalidEmployeeNumberException.java

1 package com.cts.employeedetailsreport.exception;

2

3 public class InvalidEmployeeNumberException extends Exception

4 {

5

6 String strMsg1;

7 Throwable strMsg2;

8

9

10 public InvalidEmployeeNumberException() {

11 super();

12 }

13 public InvalidEmployeeNumberException(String strMsg1)

14 {

15 super(strMsg1);

16 }

17

18 public InvalidEmployeeNumberException(String strMsg1, Throwable strMsg2) {

19 super();

20 this.strMsg1 = strMsg1;

21 this.strMsg2 = strMsg2;

22 }

23 }

EmployeeDetailsReport/src/com/cts/employeedetailsreport/model/EmployeeDetails.java

1 package com.cts.employeedetailsreport.model;

2

3 public class EmployeeDetails {

4 private String employeeNumber;

5 private String employeeName;

6 private String level;

7 private int extraWorkingHours;

8 private double totalSalary;

9

10

11

12

13 //Constructors

14

15 public EmployeeDetails(String string1, String string2, String string3, int i,double sal) {

16 this.employeeNumber=string1;

17 this.employeeName=string2;

18 this.level=string3;

19 this.extraWorkingHours=i;

20 this.totalSalary=sal;

21 }

22

23

24

25

26

27

28 public EmployeeDetails() {

29 }

30

31

32

33

34 //getters and setters

35

36 public String getEmployeeNumber() {

37 return employeeNumber;

38 }

39

40

41

42

43

44

45 public void setEmployeeNumber(String employeeNumber) {

46 this.employeeNumber = employeeNumber;

47 }

48

49

50

51

52

53

54 public String getEmployeeName() {

55 return employeeName;

56 }

57

58

59

60

61

62

63 public void setEmployeeName(String employeeName) {

64 this.employeeName = employeeName;

65 }

66

67

68

69

70

71

72 public String getLevel() {

73 return level;

74 }

75

76

77

78

79

80

81 public void setLevel(String level) {

82 this.level = level;

83 }

84

85

86

87

88

89

90 public int getExtraWorkingHours() {

91 return extraWorkingHours;

92 }

93

94

95

96

97

98

99 public void setExtraWorkingHours(int extraWorkingHours) {

100 this.extraWorkingHours = extraWorkingHours;

101 }

102

103

104

105

106

107

108 public double getTotalSalary() {

109 return totalSalary;

110 }

111

112

113

114

115

116

117 public void setTotalSalary(double totalSalary) {

118 this.totalSalary = totalSalary;

119 }

120

121

122

123

124

125

126 @Override

127 public String toString() {

128 return "EmployeeDetails [employeeNumber=" + employeeNumber + ", employeeName=" + employeeName + ", level="

129 + level + ", extraWorkingHours=" + extraWorkingHours + ", totalSalary=" + totalSalary + "]";

130 }

131

132

133

134

135 }

136

EmployeeDetailsReport/src/com/cts/employeedetailsreport/service/HospitalManagement.java

1 package com.cts.employeedetailsreport.service;

2

3 import java.util.ArrayList;

4 import java.util.List;

5 import com.cts.employeedetailsreport.util.ApplicationUtil;

6 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;

7 import com.cts.employeedetailsreport.model.EmployeeDetails;

8

9 public class HospitalManagement {

10 private List<String>employeeRecords;

11 private List<String>getEmployeeRecords(){

12 return employeeRecords;

13 }

14 public void setEmployeeRecords(List<String>employeeRecords){

15 this.employeeRecords=employeeRecords;

16 }

17

18

19 public static ArrayList <EmployeeDetails> buildEmployeeList(List <String> employeeRecords) {

20

21

22 final String COMMADELIMITER = ",";

23 ArrayList <EmployeeDetails> empList = new ArrayList<EmployeeDetails>();

24 int listSize=employeeRecords.size();

25 int i=0;

26 EmployeeDetails empdet;

27 while(listSize-->0){

28 String[] employeeDetailsString=employeeRecords.get(i++).split(COMMADELIMITER);

29 try{

30 if (ApplicationUtil.validate(employeeDetailsString[0])){

31 int extraHours = Integer.parseInt(employeeDetailsString[3]);

32 double sal = calculateTotalSalary(employeeDetailsString[2], extraHours);

33 empdet = new EmployeeDetails(employeeDetailsString[0], employeeDetailsString[1], employeeDetailsString[2], extraHours, sal);

34 empList.add(empdet);

35 }

36 }

37 catch(InvalidEmployeeNumberException in){

38 System.out.println(in);

39 }

40 }

41

42 //fill the code here

43

44 return empList;

45 }

46

47

48

49

50

51 public boolean addEmployeeList(String inputFeed) throws InvalidEmployeeNumberException

52 {

53 //fill the code here

54 try{

55 this.setEmployeeRecords(ApplicationUtil.readFile(inputFeed));

56 return true;

57 }

58 catch(Exception e){

59 e.printStackTrace();

60 }

61 return false;

62 }

63

64

65 public static double calculateTotalSalary(String level,int extraWorkingHours)

66 {

67 double sal=0.0;

68 //fill the code here

69 if(level.equals("level1")){

70 sal=75000+(1000\*extraWorkingHours);

71 }

72 else if(level.equals("level2")){

73 sal=50000+(1000\*extraWorkingHours);

74 }

75 else if(level.equals("level3")){

76 sal=35000+(1000\*extraWorkingHours);

77 }

78 else if(level.equals("level4")){

79 sal=25000+(1000\*extraWorkingHours);

80 }

81 return sal;

82 }

83

84

85 }

86

EmployeeDetailsReport/src/com/cts/employeedetailsreport/skeleton/SkeletonValidator.java

1 package com.cts.employeedetailsreport.skeleton;

2

3

4 import java.lang.reflect.Method;

5 import java.util.logging.Level;

6 import java.util.logging.Logger;

7

8

9

10 /\*\*

11 \* @author TJ

12 \*

13 \* This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth auto evaluation

14 \*

15 \*/

16 public class SkeletonValidator {

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

18 public SkeletonValidator() {

19

20 validateClassName("com.cts.employeedetailsreport.dao.DetailsDAO");

21 validateClassName("com.cts.employeedetailsreport.dao.DBConnectionManager");

22 validateClassName("com.cts.employeedetailsreport.model.EmployeeDetails");

23 validateClassName("com.cts.employeedetailsreport.service.HospitalManagement");

24 validateClassName("com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException");

25 validateClassName("com.cts.employeedetailsreport.util.ApplicationUtil");

26 //----

27 validateMethodSignature("buildEmployeeList:ArrayList,addEmployeeList:boolean","com.cts.employeedetailsreport.service.HospitalManagement");

28 validateMethodSignature("insertEmployeeList:boolean","com.cts.employeedetailsreport.dao.DetailsDAO");

29 validateMethodSignature("getInstance:DBConnectionManager,getConnection:Connection","com.cts.employeedetailsreport.dao.DBConnectionManager");

30

31

32

33 }

34

35

36

37 protected final boolean validateClassName(String className) {

38

39 boolean iscorrect = false;

40 try {

41 Class.forName(className);

42 iscorrect = true;

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

44

45 } catch (ClassNotFoundException e) {

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

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

48

49 } catch (Exception e) {

50 LOG.log(Level.SEVERE,

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

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

53 }

54 return iscorrect;

55

56 }

57

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

59 Class cls = null;

60 try {

61

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

63 boolean errorFlag = false;

64 String[] methodSignature;

65 String methodName = null;

66 String returnType = null;

67

68 for (String singleMethod : actualmethods) {

69 boolean foundMethod = false;

70 methodSignature = singleMethod.split(":");

71

72 methodName = methodSignature[0];

73 returnType = methodSignature[1];

74 cls = Class.forName(className);

75 Method[] methods = cls.getMethods();

76 for (Method findMethod : methods) {

77 if (methodName.equals(findMethod.getName())) {

78 foundMethod = true;

79 if (!(findMethod.getReturnType().getSimpleName().equals(returnType))) {

80 errorFlag = true;

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

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

83

84 } else {

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

86 }

87

88 }

89 }

90 if (!foundMethod) {

91 errorFlag = true;

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

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

94 }

95

96 }

97 if (!errorFlag) {

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

99 }

100

101 } catch (Exception e) {

102 LOG.log(Level.SEVERE,

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

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

105 }

106 }

107

108 }

109

110

EmployeeDetailsReport/src/com/cts/employeedetailsreport/util/ApplicationUtil.java

1 package com.cts.employeedetailsreport.util;

2

3

4 import java.util.ArrayList;

5 import java.util.List;

6 import java.io.FileInputStream;

7 import java.io.FileNotFoundException;

8 import java.io.InputStreamReader;

9 import java.io.BufferedReader;

10 import java.nio.charset.StandardCharsets;

11

12 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;

13

14 public class ApplicationUtil {

15 public static List<String> readFile(String filePath)

16 {

17 List<String> employeeList=new ArrayList<String>();

18

19 // FILL THE CODE HERE

20 try(BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(filePath),StandardCharsets.UTF\_8));){

21 String line;

22 while((line=br.readLine())!=null){

23 employeeList.add(line);

24 }

25 }

26 catch(Exception e){

27

28 }

29

30 return employeeList;

31

32 }

33 public static boolean validate(String employeeNumber) throws InvalidEmployeeNumberException

34 {

35 boolean val=false;

36 // FILL THE CODE HERE

37 int n = employeeNumber.length();

38 if(n!=7) throw new InvalidEmployeeNumberException("Invalid Employee Number");

39

40 char charArray[]=employeeNumber.toCharArray();

41 if(charArray[0]!='P' && charArray[1]!='R') throw new InvalidEmployeeNumberException("Invalid Employee Number");

42

43 for(int i=2;i<n;i++){

44 if(!Character.isDigit(charArray[i])){

45 throw new InvalidEmployeeNumberException("Invalid Employee Number");

46 }

47 }

48

49 val = true;

50

51 return val;

52

53

54

55 }

56

57

58 }

59