```
package com.cts.employeedetailsreport.client;
import
com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
import com.cts.employeedetailsreport.service.HospitalManagement;
import com.cts.employeedetailsreport.skeleton.SkeletonValidator;

public class EmployeeDetailsMain {

	public static void main(String[] args) throws
InvalidEmployeeNumberException {
		// CODE SKELETON - VALIDATION STARTS
		// DO NOT CHANGE THIS CODE

		new SkeletonValidator();
		HospitalManagement hm=new HospitalManagement();
		hm.addEmployeeList("C:\\Users\\shams\\eclipse-
workspace\\solution\\solution\\EmployeeDetailsReport\\inputfeed.txt");//j
o waha diya rahega usey use karna
		// CODE SKELETON - VALIDATION ENDS

// TYPE YOUR CODE HERE
	}

}

============================================================
package com.cts.employeedetailsreport.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.employeedetailsreport.exception.InvalidEmployeeNumberException;


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 Properties props = null;



	 private static Connection con = null;
	 private static DBConnectionManager instance;
	public  DBConnectionManager()  throws
InvalidEmployeeNumberException
	{
		FileInputStream fis=null;

		try {
			fis = new FileInputStream(PROPERTY_FILE);
```

```
                  props = new Properties();
                  props.load(fis);
            Class.forName(props.getProperty(DRIVER));
            DBConnectionManager.con =
            DriverManager.getConnection(props.getProperty(URL),
            props.getProperty(USER_NAME),
            props.getProperty(PASSWORD));
            } catch (ClassNotFoundException ex) {

                  throw new InvalidEmployeeNumberException();
                  } catch (SQLException e) {
                  throw new InvalidEmployeeNumberException();
                  }
            catch (FileNotFoundException e) {

                  throw new InvalidEmployeeNumberException();
                  } catch (IOException e) {
                        throw new InvalidEmployeeNumberException();
                  }
            finally {
                  if (fis != null) {
                  try {
                  fis.close();
                  } catch (IOException e) {
                        throw new InvalidEmployeeNumberException();

                  }
                  }
                  }

            //Class.forName(com.mysql.cj.jdbc.Driver);
            //FILL THE CODE HERE
      }
      public static DBConnectionManager getInstance() throws
InvalidEmployeeNumberException {
            //FILL THE CODE HERE

            return instance;
      }
      public Connection getConnection() {
            return con;
      }
}


==========


package com.cts.employeedetailsreport.dao;

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

import
com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
import com.cts.employeedetailsreport.model.EmployeeDetails;

public class DetailsDAO {
```

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

		boolean recordsAdded = false;

		// FILL THE CODE HERE
		try(Connection con =
DBConnectionManager.getInstance().getConnection()) {
			for(EmployeeDetails stdAdmObj:eList) {
				String sql = "INSERT INTO students
	VALUES(?,?,?,?,?);";
				PreparedStatement prepState =
con.prepareStatement(sql);
				prepState.setString(1,
stdAdmObj.getEmployeeNumber());
				prepState.setString(2,
stdAdmObj.getEmployeeName());
				prepState.setString(3,stdAdmObj.getLevel());
				prepState.setInt(4,
stdAdmObj.getExtraWorkingHours());
				prepState.setDouble(5,
stdAdmObj.getTotalSalary());
				prepState.execute();
			}
			recordsAdded= true;
		}
	   catch(Exception e) {
		System.out.println(e.getMessage());
		throw new InvalidEmployeeNumberException(e.getMessage(),
e.getCause());
		}

		return recordsAdded;
	}



}
```

==========

```
package com.cts.employeedetailsreport.exception;

@SuppressWarnings("serial")
public class InvalidEmployeeNumberException extends Exception
{

	String strMsg1;
	Throwable strMsg2;


	public InvalidEmployeeNumberException() {
		super();
	}
	public InvalidEmployeeNumberException(String strMsg1)
	{
```

```
            super(strMsg1);
      }

      public InvalidEmployeeNumberException(String strMsg1, Throwable
strMsg2) {
            super();
            this.strMsg1 = strMsg1;
            this.strMsg2 = strMsg2;
      }
}
```

==========

```
package com.cts.employeedetailsreport.model;

public class EmployeeDetails {
      private String employeeNumber;
      private String employeeName;
      private String level;
      private int extraWorkingHours;
      private double totalSalary;

//Constructors

      public EmployeeDetails(String string1, String string2, String
string3, int i, double sal) {
            this.employeeNumber = string1;
            this.employeeName = string2;
            this.level = string3;
            this.extraWorkingHours = i;
            this.totalSalary = sal;
      }

      public EmployeeDetails() {
      }

//getters and setters

      public String getEmployeeNumber() {
            return employeeNumber;
      }

      public void setEmployeeNumber(String employeeNumber) {
            this.employeeNumber = employeeNumber;
      }

      public String getEmployeeName() {
            return employeeName;
      }

      public void setEmployeeName(String employeeName) {
            this.employeeName = employeeName;
      }

      public String getLevel() {
            return level;
      }

      public void setLevel(String level) {
```

```
            this.level = level;
      }

      public int getExtraWorkingHours() {
            return extraWorkingHours;
      }

      public void setExtraWorkingHours(int extraWorkingHours) {
            this.extraWorkingHours = extraWorkingHours;
      }

      public double getTotalSalary() {
            return totalSalary;
      }

      public void setTotalSalary(double totalSalary) {
            this.totalSalary = totalSalary;
      }

      @Override
      public String toString() {
            return "EmployeeDetails [employeeNumber=" + employeeNumber +
", employeeName=" + employeeName + ", level="
                        + level + ", extraWorkingHours=" +
extraWorkingHours + ", totalSalary=" + totalSalary + "]";
      }

}
=====================================================================

package com.cts.employeedetailsreport.service;

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

import
com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
import com.cts.employeedetailsreport.model.EmployeeDetails;
import com.cts.employeedetailsreport.util.ApplicationUtil;

public class HospitalManagement {


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


            final String COMMADELIMITER = ",";
            ArrayList <EmployeeDetails> empList = new ArrayList<>();

      //fill the code here
              for (String str : employeeRecords)
             {
                    String[] tokens = str.split(COMMADELIMITER);
                    EmployeeDetails emp=new EmployeeDetails();
                    emp.setEmployeeName(tokens[0].toString());
                    emp.setEmployeeName(tokens[1].toString());
                    emp.setLevel(tokens[2].toString());

emp.setExtraWorkingHours(Integer.parseInt(tokens[3].toString()));
```

```
emp.setTotalSalary(Double.parseDouble(tokens[4].toString()));
                  empList.add(emp);

            }
            return empList;
      }



      public boolean addEmployeeList(String inputFeed) throws
InvalidEmployeeNumberException
      {
      //fill the code here
            ApplicationUtil au=new ApplicationUtil();
            List<String> employeeList=new ArrayList<>();
            ArrayList <EmployeeDetails> empList = new ArrayList<>();
            employeeList=    au.readFile(inputFeed);
            empList=buildEmployeeList(employeeList);
            for(String res:employeeList)
            {
                  System.out.println(res);
            }
            return false;
      }



      public static double calculateTotalSalary(String level,int
extraWorkingHours)
    {
            double sal=0.0;
      //fill the code here
if("level1".equals(level))
{
      sal=sal+75000+(extraWorkingHours*1000);
}
else if("level2".equals(level))
{
      sal=sal+50000+(extraWorkingHours*1000);
}
else if("level3".equals(level))
{
      sal=sal+35000+(extraWorkingHours*1000);
}
if("level4".equals(level))
{
      sal=sal+25000+(extraWorkingHours*1000);
}
            return sal;
    }


}
=====================================================================


package com.cts.employeedetailsreport.skeleton;

import java.lang.reflect.Method;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author TJ
 *
 *         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 {
      private static final Logger LOG =
Logger.getLogger("SkeletonValidator");

      public SkeletonValidator() {


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

      validateClassName("com.cts.employeedetailsreport.dao.DBConnectionMa
nager");

      validateClassName("com.cts.employeedetailsreport.model.EmployeeDeta
ils");

      validateClassName("com.cts.employeedetailsreport.service.HospitalMa
nagement");

      validateClassName("com.cts.employeedetailsreport.exception.InvalidE
mployeeNumberException");

      validateClassName("com.cts.employeedetailsreport.util.ApplicationUt
il");
            // ----

      validateMethodSignature("buildEmployeeList:ArrayList,addEmployeeLis
t:boolean",

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

      validateMethodSignature("getInstance:DBConnectionManager,getConnect
ion:Connection",

      "com.cts.employeedetailsreport.dao.DBConnectionManager");

      }

      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.getName()))
{
                                    foundMethod = true;

                                    if
(!(findMethod.getReturnType().getSimpleName().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) {
```

```
                              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");
                              errorFlag = true;
                        }

                  }
                  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");
            }
      }

}
```

==========

```
package com.cts.employeedetailsreport.util;


import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import
com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
import com.cts.employeedetailsreport.service.HospitalManagement;

public class ApplicationUtil {
       public static List<String> readFile(String filePath) throws
InvalidEmployeeNumberException
          {
            List<String> employeeList=new ArrayList<>();

                  // FILL THE CODE HERE
            try(BufferedReader in = new BufferedReader(new
FileReader(filePath))) {
                        String str;
                        while ((str = in.readLine()) != null) {

                            String[] tokens = str.split(",");
                            String vemp = tokens[0].toString();
                           boolean check= validate(vemp);
                           if(check==true)
                           {
                          int sal=  (int)
HospitalManagement.calculateTotalSalary(tokens[2].toString(),Integer.pars
eInt(tokens[3]));
                           str=str+","+sal;
```

```
                        employeeList.add(str);
                         }
                    }
                }
                catch (IOException e) {
                    System.out.println("File Read Error");
                }

          return employeeList;

        }
        public static boolean validate(String employeeNumber) throws
InvalidEmployeeNumberException
          {
          boolean val=false;
                // FILL THE CODE HERE
        int len=     employeeNumber.length();
        String pr=employeeNumber.substring(0,2);
        if((pr.equals("PR"))&&(len==7)) {
          val=true;}
        else if(pr.equals("TR")) {
          val=false;}

          return val;



          }


}
```