Automatic evaluation[-]

Proposed grade: 95.0 / 100
Result Description
[-]SOURCE CODE ANALYZER REPORT

```
Error Msg: Avoid if (x != y) ..; else ..;
Variable Name:
Priority: Medium
Class Name: SkeletonValidator

Error Msg: Possible unsafe assignment to a non-final static field in a constructor.
Variable Name: connection
Priority: Medium
Class Name: DBConnectionManager
```

[-]Grading and Feedback

```
Collections and String - 10.0 / 10.0(Success)

Classes and Objects and JDBC API - 15.0 / 15.0(Success)

Operators,Control Statements and array - 10.0 / 10.0(Success)

Classes and Objects and Collections - 15.0 / 15.0(Success)

File IO - 10.0 / 10.0(Success)

Utility - 10.0 / 10.0(Success)

DB connection - 10.0 / 10.0(Success)

Java Date API, Operators and Control Statements - 10.0 / 10.0(Success)

Comments and best practices/standards - 5.0 / 10.0(Partial)
```

Test Case Failed

UNOAdmission/database.properties

```
#IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE
#ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY
#YOU CAN CHANGE THE VALUE OF THE PROPERTY
#LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD using this properties file only.
#Do not hard code the values

drivername=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/uno_admission
username=root
password=
```

UNOAdmission/inputFeed.txt

```
A001,S001,2020-01-15,EEE,2020-01-25,YES,YES,Approved
A002,S002,2020-02-04,MECH,2020-02-12,N0,YES,Approved
A003,S003,2020-04-23,CSE,2020-05-27,YES,NO,Approved
A004,S004,2020-07-16,IT,2020-07-24,NO,NO,Approved
A005,S005,2020-08-10,ECE,2020-08-11,YES,YES,Approved
A006,S006,2020-09-01,EEE,2020-09-10,YES,NO,Pending
A007,S007,2020-10-19,CIVIL,2020-10-28,N0,YES,Approved
```

UNOAdmission/src/com/cts/unoadm/dao/StudentAdmissionDAO.java

```
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;
import com.cts.unoadm.util.*;
import java.sql.*;

/*
CREATE DATABASE CTSUNO

CREATE TABLE UNO_ADMISSION(
    ADMISSION_ID  VARCHAR(50) PRIMARY KEY,
    STUDENT_CODE VARCHAR(50)NOT NULL,
    DATE_OF_COUNSELING DATE NOT NULL,
    DEPARTMENT_NAME VARCHAR(20),
    DATE_OF_ADMISSION DATE NOT NULL,
    PREFER_COLLEGE_HOSTEL VARCHAR(20),
    FIRST_GRADUATE VARCHAR(20),
    MANAGER_APPROVAL VARCHAR(10),
    ADMISSION_FEE DOUBLE,
    TUITION_FEE DOUBLE,
    HOSTEL_FEE DOUBLE,
    TOTAL_COLLEGE_FEE DOUBLE,
    FINAL_STATUS_OF_ADMISSION VARCHAR(20)
);
*/
public class StudentAdmissionDAO {

          public boolean addStudentAdmissionDetails(List<StudentAdmission> stdAdmissions) throws StudentAdmissionException {
                          boolean recordsAdded = false;
                          Connection con = DBConnectionManager.getInstance().getConnection();
                          PreparedStatement ps = null;
                          try{
                            //inserting values of list stdAdmissions into database
                            String query = "insert into UNO_ADMISSION(ADMISSION_ID,STUDENT_CODE,DATE_OF_COUNSELING,DEPARTMENT_NAME,DATE_OF_ADMISSION,PREFER_COLLEGE_HOSTEL,FIRST_GRADUATE,MANAGER_APPROVAL,ADMISSION_FEE,TUITION_FEE,HOSTEL_FEE,TOTAL_COLLEGE_FEE,FINAL_STATUS_OF_ADMISSION) values(?,?,?,?,?,?,?,?,?,?,?,?,?)";
                            for(StudentAdmission e:stdAdmissions) {
                                ps = con.prepareStatement(query);
                                ps.setString(1,e.getAdmissionId());
                                ps.setString(2,e.getStudentCode());
                                ps.setDate(3,ApplicationUtil.convertUtilToSqlDate(e.getDateOfCounseling()));
                                ps.setString(4,e.getDepartmentName());
                                ps.setDate(5,ApplicationUtil.convertUtilToSqlDate(e.getDateOfAdmission()));
                                ps.setString(6,e.getPreferCollegeHostel());
                                ps.setString(7,e.getFirstGraduate());
                                ps.setString(8,e.getManagerApproval());
                                ps.setDouble(9,e.getAdmissionFee());
                                ps.setDouble(10,e.getTuitionFee());
                                ps.setDouble(11,e.getHostelFee());
                                ps.setDouble(12,e.getTotalCollegeFee());
                                ps.setString(13,e.getFinalStatusOfAdmission());
                                int i = ps.executeUpdate();
```

```java
                    if(i>0)
                    {
                        recordsAdded = true;
                    }
                    else
                    {
                        recordsAdded = false;
                    }
                }


            }
            catch(SQLException e)
            {
                try{
                con.rollback();
                }catch(Exception e1){
                    e.printStackTrace();
                }
            }
            catch(Exception e) {
                e.printStackTrace();
                //throw new StudentAdmissionException("Database Value Insertion Failed", e.getCause());
            }
            finally{
                try{
                 ps.close();
                con.close();
                }catch(Exception e) {
                e.printStackTrace();
                //throw new StudentAdmissionException("Database Value Insertion Failed", e.getCause());
                }
            }
            //code here

            return recordsAdded;
    }

    public List<StudentAdmission> getAllStudentAdmissionDetails() throws StudentAdmissionException {

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

            //code here
            //Retrieval of all records from database
            String query = "select * from UNO_ADMISSION";
            try(Connection con = DBConnectionManager.getInstance().getConnection();
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(query);){
                while(rs.next())
                {
                    //storing retrieved records in object
                    StudentAdmission obj = new StudentAdmission();
                    obj.setAdmissionId(rs.getString(1));
                obj.setStudentCode(rs.getString(2));
                obj.setDateOfCounseling(new java.util.Date(rs.getDate(3).getTime()));
                obj.setDepartmentName(rs.getString(4));
                obj.setDateOfAdmission(new java.util.Date(rs.getDate(5).getTime()));
                obj.setPreferCollegeHostel(rs.getString(6));
                obj.setFirstGraduate(rs.getString(7));
                obj.setManagerApproval(rs.getString(8));
                obj.setAdmissionFee(rs.getDouble(9));
                obj.setTuitionFee(rs.getDouble(10));
```

```
                    obj.setHostelFee(rs.getDouble(11));
                    obj.setTotalCollegeFee(rs.getDouble(12));
                    obj.setFinalStatusOfAdmission(rs.getString(13));
                    //adding StudentAdmission object into arraylist
                    stdAdmissions.add(obj);
                    }

                }catch(Exception e)
                {
                    e.printStackTrace();
                    //throw new StudentAdmissionException("Database Value Retrieval Failed", e.getCause());
                }
                return stdAdmissions;

        }
}
```

UNOAdmission/src/com/cts/unoadm/exception/StudentAdmissionException.java

```
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);
        }
}

```

UNOAdmission/src/com/cts/unoadm/main/MainApp.java

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

import com.cts.unoadm.skeletonvalidator.SkeletonValidator;
import com.cts.unoadm.service.*;
import com.cts.unoadm.util.*;
public final class MainApp {
    private MainApp(){}
        public static void main(String[] args) {
                //Don't delete this code
                //Skeletonvalidaton starts
                new SkeletonValidator();
                //Skeletonvalidation ends

                //Write your code here..
                try{
                StudentAdmissionService service = new StudentAdmissionService();
                System.out.println(service.addStudentAdmissionDetails("inputFeed.txt"));
                System.out.println(service.searchStudentAdmission("A005"));
                }catch(Exception e){e.printStackTrace();}
                //List<StudentAdmission> studentAdmissionList = service.buildStudentAdmissionsList(ApplicationUtil.readFile("inputFeed.txt"));
                /*for(StudentAdmission e:studentAdmissionList)
                {
                    System.out.println(e);
```

```java
				}*/
		}

}

```

UNOAdmission/src/com/cts/unoadm/service/StudentAdmissionService.java

```java
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;
import com.cts.unoadm.util.*;
import com.cts.unoadm.dao.*;
public class StudentAdmissionService {

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

			//Code here
			//storing each line into List of StudentAdmission objects
			for(String e:studentAdmissionRecords) {
			    String res[] = e.split(",");
			    String admissionId = res[0];
			    String studentCode = res[1];
			    String dateOfCounseling = res[2];
			    String departmentName = res[3];
			    String dateOfAdmission = res[4];
			    String preferCollegeHostel = res[5];
			    String firstGraduate = res[6];
			    String managerApproval = res[7];
			    StudentAdmission obj = new StudentAdmission();
			    obj.setAdmissionId(admissionId);
			    obj.setStudentCode(studentCode);
			    //converting String to java.uti.Date
			    obj.setDateOfCounseling(ApplicationUtil.convertStringToDate(dateOfCounseling));
			    obj.setDepartmentName(departmentName);
			    //converting String to java.uti.Date
			    obj.setDateOfAdmission(ApplicationUtil.convertStringToDate(dateOfAdmission));
			    obj.setPreferCollegeHostel(preferCollegeHostel);
			    obj.setFirstGraduate(firstGraduate);
			    obj.setManagerApproval(managerApproval);
			    double[] studentAdmissionCosts = calculateTotalCollegeFee(preferCollegeHostel,firstGraduate,departmentName);
			    obj.setAdmissionFee(studentAdmissionCosts[0]);
			    obj.setTuitionFee(studentAdmissionCosts[1]);
			    obj.setHostelFee(studentAdmissionCosts[2]);
			    obj.setTotalCollegeFee(studentAdmissionCosts[3]);
			    obj.setFinalStatusOfAdmission("AdmissionSuccessfull");


			    studentAdmissionList.add(obj);
			}

```

```java
                return studentAdmissionList;
        }


        public boolean addStudentAdmissionDetails(String inputFeed) throws StudentAdmissionException {

                //Code here
                List<StudentAdmission> studentAdmissionList = 
StudentAdmissionService.buildStudentAdmissionsList(ApplicationUtil.readFile(inputFeed));
                StudentAdmissionDAO stdDao = new StudentAdmissionDAO();
                return stdDao.addStudentAdmissionDetails(studentAdmissionList);
        }

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

                //Code here..
                if("YES".equals(preferCollegeHostel))
                {
                        studentAdmissionCosts[2]=75000;
                }
                else{
                        studentAdmissionCosts[2]=0;
                }
                if("EEE".equals(departmentName)) {

                        studentAdmissionCosts[0]=30000;
                        studentAdmissionCosts[1]=45000;
                    }
                else if("ECE".equals(departmentName)) {

                        studentAdmissionCosts[0]=30000;
                        studentAdmissionCosts[1]=50000;
                    }
                else if("CSE".equals(departmentName)) {
                    studentAdmissionCosts[0]=30000;
                        studentAdmissionCosts[1]=45000;

                    }
                else if("MECH".equals(departmentName)) {
                  studentAdmissionCosts[0]=30000;
                        studentAdmissionCosts[1]=55000;

                    }
                else if("CIVIL".equals(departmentName)) {
                        studentAdmissionCosts[0]=30000;
                        studentAdmissionCosts[1]=50000;

                    }
                else if("IT".equals(departmentName)) {
                        studentAdmissionCosts[0]=30000;
                        studentAdmissionCosts[1]=45000;

                    }
                 //for first graduate discount is there
                if("YES".equals(firstGraduate)) {
studentAdmissionCosts[3]=studentAdmissionCosts[0]+studentAdmissionCosts[1]+studentAdmissionCosts[2]-20000;
                }
                else{
```

```
studentAdmissionCosts[3]=studentAdmissionCosts[0]+studentAdmissionCosts[1]+studentAdmissionCosts[2];
                }
                return studentAdmissionCosts;
        }

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

                //Code here..
                StudentAdmissionDAO stdDao = new StudentAdmissionDAO();
                List<StudentAdmission> stdAdmissions = stdDao.getAllStudentAdmissionDetails();
                for(StudentAdmission e:stdAdmissions) {
                    if(e.getAdmissionId().equals(admissionId)) {
                        status = true;
                        System.out.println(e);
                        break;
                    }
                }
                return status;
        }
}

```

UNOAdmission/src/com/cts/unoadm/skeletonvalidator/SkeletonValidator.java

```
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 {
    private static final Logger LOG = Logger.getLogger("SkeletonValidator");
        public SkeletonValidator() {
                validateClassName("com.cts.unoadm.util.DBConnectionManager");
                validateClassName("com.cts.unoadm.util.ApplicationUtil");
                validateClassName("com.cts.unoadm.service.StudentAdmissionService");
                validateClassName("com.cts.unoadm.dao.StudentAdmissionDAO");
                validateClassName("com.cts.unoadm.vo.StudentAdmission");
                validateClassName("com.cts.unoadm.exception.StudentAdmissionException");


                validateMethodSignature(
"addStudentAdmissionDetails:boolean,getAllStudentAdmissionDetails:List",
                                "com.cts.unoadm.dao.StudentAdmissionDAO");
                validateMethodSignature(
"buildStudentAdmissionsList:List,addStudentAdmissionDetails:boolean,calculateTotalCollegeFee:double[],searchStudentAdmission:boolean",
                                "com.cts.unoadm.service.StudentAdmissionService");
                validateMethodSignature(
"readFile:List,convertUtilToSqlDate:Date,convertStringToDate:Date,checkIfValidAdmission:boolean",
```

```java
                "com.cts.unoadm.util.ApplicationUtil");
        validateMethodSignature(
                "getConnection:Connection,getInstance:DBConnectionManager",
                "com.cts.unoadm.util.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) {
                    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");
        }
    }

}

```

UNOAdmission/src/com/cts/unoadm/util/ApplicationUtil.java

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


import java.util.*;
import java.io.*;
import java.text.*;

import com.cts.unoadm.exception.StudentAdmissionException;

public final class ApplicationUtil {

    /**
     * @param fileName
     * @return List<String>
     * @throws StudentAdmissionException
     */

     private ApplicationUtil(){}
    public static List<String> readFile(String fileName) throws StudentAdmissionException {
        List<String> studentAdmissionList = new ArrayList<String>();
          //Code here..
        FileReader fr = null;
        BufferedReader br = null;
        try{
            fr=new FileReader(fileName);
            br = new BufferedReader(fr);
            String line = null;
            while((line=br.readLine())!=null)
            {
                    String []res = line.split(",");
                    String managerApproval = res[7];
                    Date dtOfCounseling = convertStringToDate(res[2]);
                    Date dtOfAdmission = convertStringToDate(res[4]);
                    if(checkIfValidAdmission(dtOfCounseling,dtOfAdmission,managerApproval))
                    {
                        studentAdmissionList.add(line);
                    }

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

            return studentAdmissionList;
    }

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

            java.sql.Date sDate = new java.sql.Date(uDate.getTime());

            //Code here..


            return sDate;
    }

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

            //Code here..
            try{
                SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd",Locale.ENGLISH);
                return format.parse(inDate);
            }catch(Exception e){e.printStackTrace();
                return null;
            }
    }

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

            //Code here..
            if("Approved".equals(manager) && ((dtOfAdmission.getTime() -
dtOfCounseling.getTime())/(1000*60*60*24))%365<= 10)
            {
                admissionValidity = true;
            }

            return admissionValidity;
    }
}

```
UNOAdmission/src/com/cts/unoadm/util/DBConnectionManager.java

```java
/**
 * 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 final 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;
    private static DBConnectionManager instance = null;
        /**
         * @throws StudentAdmissionException
         */
        private DBConnectionManager() throws StudentAdmissionException {
                loadProperties();
                try {
                        Class.forName(props.getProperty(DRIVER));
                        connection = DriverManager.getConnection(props.getProperty(URL), props.getProperty(USER_NAME),
                                        props.getProperty(PASSWORD));
                } catch (ClassNotFoundException ex) {
                    ex.printStackTrace();
                        //throw new StudentAdmissionException("Could not find Driver class ", ex.getCause());
                } catch (SQLException e) {
                    e.printStackTrace();
                        //throw new StudentAdmissionException("Database Connection Creation Failed", e.getCause());
                }
                catch(Exception e)
                {
                    e.printStackTrace();
                    //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

        instance = new DBConnectionManager();

        return instance;
    }

    /**
     * @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) {
            e.printStackTrace();
            //throw new StudentAdmissionException("Database Property File Not Found", e.getCause());
        } catch (IOException e) {
            e.printStackTrace();
            //throw new StudentAdmissionException("Exception during property file I/O", e.getCause());
        } finally {
            if (inputStream != null) {
                try {
                    inputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                    //throw new StudentAdmissionException("Exception during property file I/O", e.getCause());
                }
            }
        }
    }
}


```

UNOAdmission/src/com/cts/unoadm/vo/StudentAdmission.java

```
/*
 * 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;
```

```java
	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 + "]";
	}

}
```