```java
/*************************************************************
* Dalton Nofs                                               *
* Login ID: nofs5491                                        *
* CS-102, Summer 2017                                       *
* Programming Assignment 3                                  *
* GpaCalc class:  calculator used to find database gpa        *
*************************************************************/
public class GpaCalc
{
    /*************************************************************
    * Method: calcGpa()                                           *
    * Purpose: Calculate gpa for given database                  *
    *                                                            *
    * Parameters:                                              *
    *         Database: targetDatabase: database to calc gpa         *
    * Returns:                                                   *
    *         double: the calculated gpa                          *
    *************************************************************/
    public double calcGpa(Database targetDatabase) throws IllegalArgumentException
    {
        int totalCredits = 0; // Total credits on not excluded classes
        double creditGpa = 0;  // Running total for credit * class grade

        // Check the status of the database
        if(targetDatabase.getArraySize() <= 0)
        {
            throw new IllegalArgumentException("N/A\nDatabase is empty!\n");
        }

        for(int index=0; index<targetDatabase.getArraySize(); index++)
        {
            int numCourses = targetDatabase.get(index).size();
            // Loop courses
            for(int index2=0;index2<numCourses;index2++)
            {
                // Check to see if exclude flag is set
                if(targetDatabase.getArrayPosition(index,index2).getExcludeFlag().
                        toUpperCase().equals("N")                                  )
                {
                    totalCredits += targetDatabase.getArrayPosition(index,index2).getCreditCount();
                    // Get grade and multiply by the credit count for the top of the gpa calc
                    try
                    {
                        // add to the total credit count
                        creditGpa += getClassGrade(targetDatabase.
                                getArrayPosition(index,index2)) *
                                targetDatabase.getArrayPosition(index,index2).
                                getCreditCount();
                    }
                    catch(IllegalArgumentException exc)
                    {
                        if(targetDatabase.getArrayPosition(index,index2).getCourseGrade().toUpperCase().equals("CR") ||
                           targetDatabase.getArrayPosition(index,index2).getCourseGrade().toUpperCase().equals("I"))
                        {
                            // do nothing
                            throw new IllegalArgumentException(
                                    "N/A\nThe grade for " +
                                       targetDatabase.getArrayPosition(index,index2).getCourseNumber() +
                                    " is \"" + targetDatabase.getArrayPosition(index,index2).getCourseGrade() +
                                    "\" which is not applicable for GPA calculations!\n");
                        }
                        else
                        {
                            throw new IllegalArgumentException(
                                    "N/A\nThe grade for " +
                                    targetDatabase.getArrayPosition(index,index2).getCourseNumber() +
                                    " is \"" + targetDatabase.getArrayPosition(index,index2).getCourseGrade() +
                                    "\" is invalid!\n");
                        }
                    }
                }
                else{/* do nothing */}
            }
        }
        // Calc final database gpa
        return (creditGpa/totalCredits);
    }

    /*************************************************************
    * Method: getClassGrade()                                    *
    * Purpose: figure out what the gpa value from string          *
    *                                                            *
    * Parameters:                                                   *
    *          Course: targetCourse: course to find the grade          *
    * Returns:                                                        *
    *          double: the calculated gpa                              *
    *************************************************************/
    double getClassGrade(Course targetCourse) throws IllegalArgumentException
    {
        double courseGrade = 0;
        // Switch for checking the uppercase version of the grade
        switch(targetCourse.getCourseGrade().toUpperCase())
        {
            case "A":     courseGrade = 4.0;
                          break;

            case "A-":     courseGrade = 3.7;
                             break;

            case "B+":     courseGrade = 3.3;
                             break;

            case "B":     courseGrade = 3.0;
                            break;

            case "B-":     courseGrade = 2.7;
                             break;

            case "C+":     courseGrade = 2.3;
                             break;

            case "C":     courseGrade = 2.0;
                             break;

            case "C-":     courseGrade = 1.7;
                             break;

            case "D+":     courseGrade = 1.3;
                        break;

            case "D":    courseGrade = 1.0;
                        break;

            case "F":    courseGrade = 0.0;
                        break;

            // Catch all non compiant grades
            default:    throw new IllegalArgumentException("Grade is not correct!");
        }
        // Return found grade if exception is not thrown first
        return courseGrade;
    }
}
```