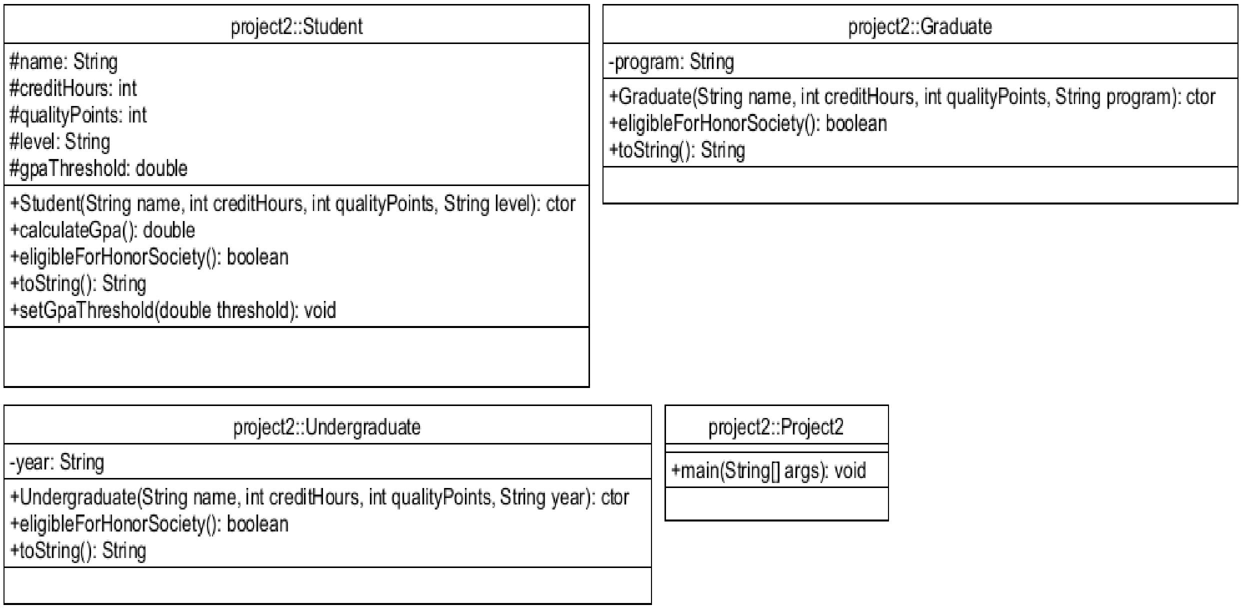
Victoria Lee
CMSC 215
6/10/2024

# Project 2 Documentation

## UML Class Diagrams and Package:



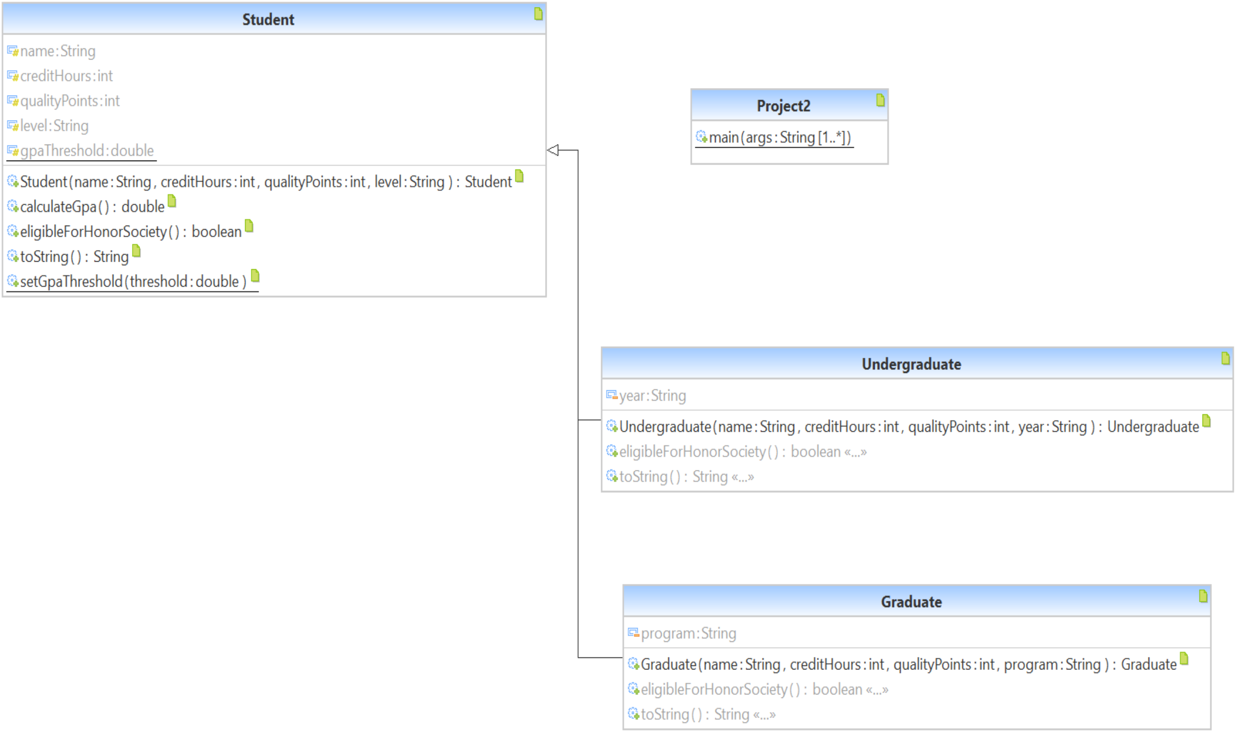
project2::Student
#name: String
#creditHours: int
#qualityPoints: int
#level: String
#gpaThreshold: double
+Student(String name, int creditHours, int qualityPoints, String level): ctor
+calculateGpa(): double
+eligibleForHonorSociety(): boolean
+toString(): String
+setGpaThreshold(double threshold): void

project2::Graduate
-program: String
+Graduate(String name, int creditHours, int qualityPoints, String program): ctor
+eligibleForHonorSociety(): boolean
+toString(): String

project2::Undergraduate
-year: String
+Undergraduate(String name, int creditHours, int qualityPoints, String year): ctor
+eligibleForHonorSociety(): boolean
+toString(): String

project2::Project2
+main(String[] args): void

## UML Class Diagrams (package not shown) with descriptions and not:

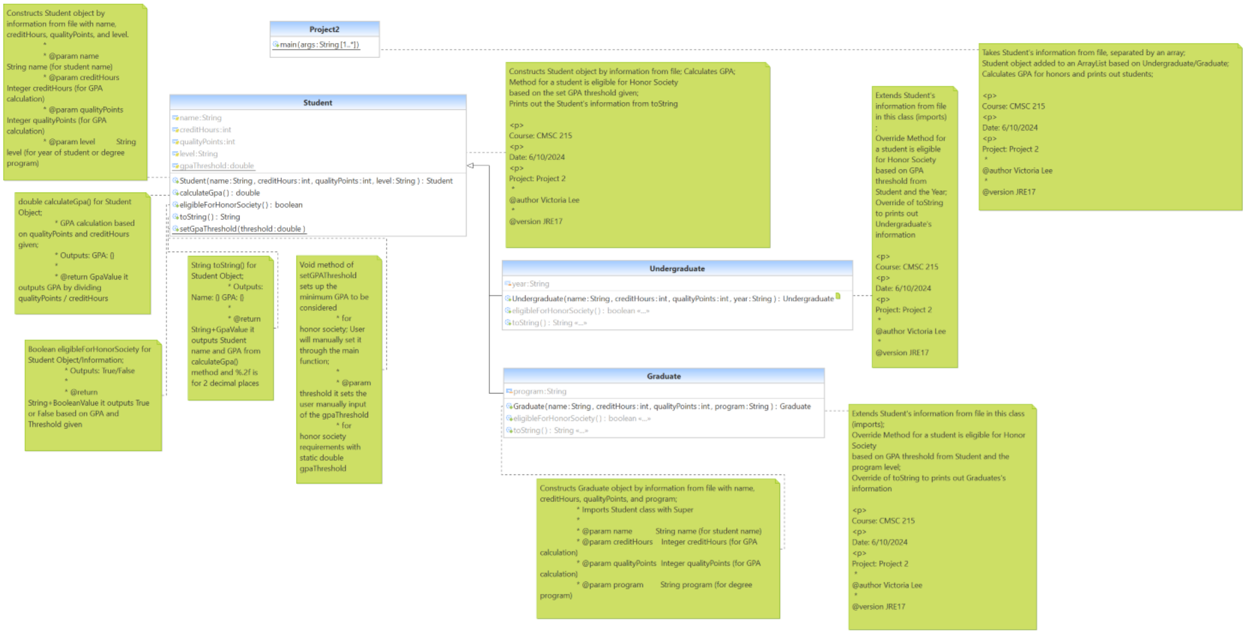


Student
name:String
creditHours:int
qualityPoints:int
level:String
gpaThreshold:double
Student(name:String, creditHours:int, qualityPoints:int, level:String) : Student
calculateGpa() : double
eligibleForHonorSociety() : boolean
toString() : String
setGpaThreshold(threshold:double)

Project2
main(args:String[1..*])

Undergraduate
year:String
Undergraduate(name:String, creditHours:int, qualityPoints:int, year:String) : Undergraduate
eligibleForHonorSociety() : boolean «...»
toString() : String «...»

Graduate
program:String
Graduate(name:String, creditHours:int, qualityPoints:int, program:String) : Graduate
eligibleForHonorSociety() : boolean «...»
toString() : String «...»


Project2
main(args: String [1..*])
Student
name: String
creditHours: int
qualityPoints: int
level: String
gpaThreshold: double
Student(name: String, creditHours: int, qualityPoints: int, level: String): Student
calculateGpa(): double
eligibleForHonorSociety(): boolean
toString(): String
setGpaThreshold(threshold: double)
Undergraduate
year: String
Undergraduate(name: String, creditHours: int, qualityPoints: int, year: String): Undergraduate
eligibleForHonorSociety(): boolean «...»
toString(): String «...»
Graduate
program: String
Graduate(name: String, creditHours: int, qualityPoints: int, program: String): Graduate
eligibleForHonorSociety(): boolean «...»
toString(): String «...»

```
Constructs Student object by
information from file with name,
creditHours, qualityPoints, and level.
            *
            * @param name
String name (for student name)
            * @param creditHours
Integer creditHours (for GPA
calculation)
            * @param qualityPoints
Integer qualityPoints (for GPA
calculation)
            * @param level         String
level (for year of student or degree
program)
```

```
double calculateGpa() for Student
Object;
            * GPA calculation based
on qualityPoints and creditHours
given;
            * Outputs: GPA: {}
            *
            * @return GpaValue it
outputs GPA by dividing
qualityPoints / creditHours
```

```
Boolean eligibleForHonorSociety for
Student Object/Information;
            * Outputs: True/False
            *
            * @return
String+BooleanValue it outputs True
or False based on GPA and
Threshold given
```

```
String toString() for
Student Object;
            * Outputs:
Name: {} GPA: {}
            *
            * @return
String+GpaValue it
outputs Student
name and GPA from
calculateGpa()
method and %.2f is
for 2 decimal places
```

```
Void method of
setGPAThreshold
sets up the
minimum GPA to be
considered
            * for
honor society; User
will manually set it
through the main
function;
            *
            * @param
threshold it sets the
user manually input
of the gpaThreshold
            * for
honor society
requirements with
static double
gpaThreshold
```

```
Constructs Student object by information from file; Calculates GPA;
Method for a student is eligible for Honor Society
based on the set GPA threshold given;
Prints out the Student's information from toString

<p>
Course: CMSC 215
<p>
Date: 6/10/2024
<p>
Project: Project 2
 *
@author Victoria Lee
 *
@version JRE17
```

```
Extends Student's
information from file
in this class (imports)
;
Override Method for
a student is eligible
for Honor Society
based on GPA
threshold from
Student and the Year;
Override of toString
to prints out
Undergraduate's
information

<p>
Course: CMSC 215
<p>
Date: 6/10/2024
<p>
Project: Project 2
 *
@author Victoria Lee
 *
@version JRE17
```

```
Takes Student's information from file, separated by an array;
Student object added to an ArrayList based on Undergraduate/Graduate;
Calculates GPA for honors and prints out students;

<p>
Course: CMSC 215
<p>
Date: 6/10/2024
<p>
Project: Project 2
 *
@author Victoria Lee
 *
@version JRE17
```

```
Extends Student's information from file in this class
(imports);
Override Method for a student is eligible for Honor
Society
based on GPA threshold from Student and the
program level;
Override of toString to prints out Graduates's
information

<p>
Course: CMSC 215
<p>
Date: 6/10/2024
<p>
Project: Project 2
 *
@author Victoria Lee
 *
@version JRE17
```

```
Constructs Graduate object by information from file with name,
creditHours, qualityPoints, and program;
            * Imports Student class with Super
            *
            * @param name          String name (for student name)
            * @param creditHours    Integer creditHours (for GPA
calculation)
            * @param qualityPoints  Integer qualityPoints (for GPA
calculation)
            * @param program        String program (for degree
program)
```

## All Test Plans/Cases:

Test 1 and Test 2 (Given):

Input:
Manually set the GPA threshold for Honor Society eligibility in main:

```
Student.setGpaThreshold(3.64);
Student.setGpaThreshold(3.59);
```

File paths of both path1 and path2 are in project2 package; project 2 package is in source;
The "src\\project2\\students.txt" & "src\\project2\\studentsTrek.txt" must be in project2 package

Output:

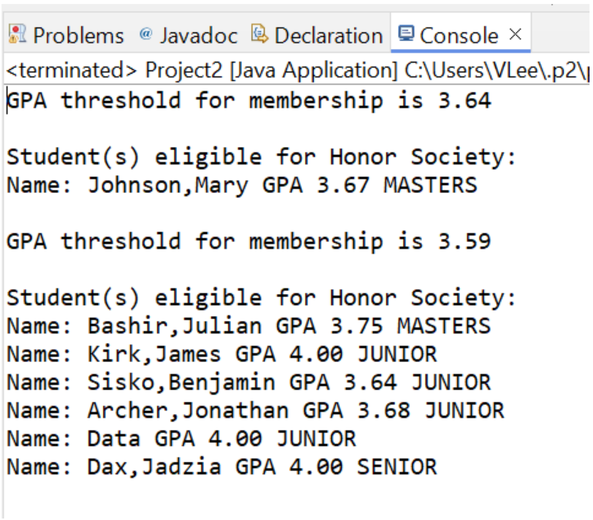
```
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 3.59

Student(s) eligible for Honor Society:
Name: Bashir,Julian GPA 3.75 MASTERS
Name: Kirk,James GPA 4.00 JUNIOR
Name: Sisko,Benjamin GPA 3.64 JUNIOR
Name: Archer,Jonathan GPA 3.68 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR
```

Demo test below:



<terminated> Project2 [Java Application] C:\Users\VLee\.p2\

```
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 3.59

Student(s) eligible for Honor Society:
Name: Bashir,Julian GPA 3.75 MASTERS
Name: Kirk,James GPA 4.00 JUNIOR
Name: Sisko,Benjamin GPA 3.64 JUNIOR
Name: Archer,Jonathan GPA 3.68 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR
```

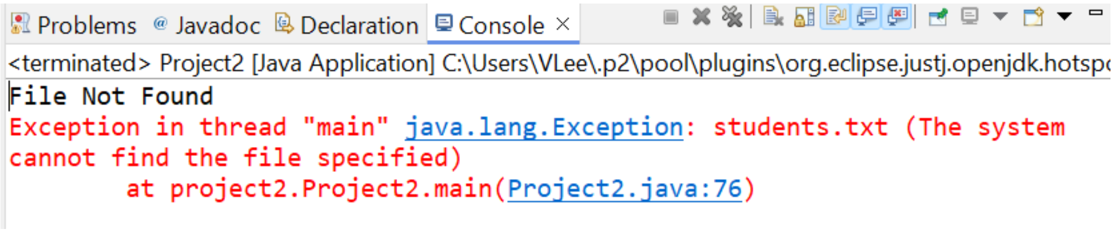
Test 3 (Created):

Note: I am testing for wrong file name or no files. I purposely changed a different file name ".txt" for one of them to throw and exception for "students.txt" file if the file is not found. It will not throw an exception for "studentsTrek.txt" as the file is found.

Input:
Change in Main:
Path path1 = Paths.get("src\\project2\\studentsThrowException.txt");
Path path2 = Paths.get("src\\project2\\studentsTrek.txt");

Output:
File Not Found
Exception in thread "main" java.lang.Exception: students.txt (The system cannot find the file specified)
          at project2.Project2.main(Project2.java:76)

Demo test below:



<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp

```
File Not Found
Exception in thread "main" java.lang.Exception: students.txt (The system
cannot find the file specified)
        at project2.Project2.main(Project2.java:76)
```

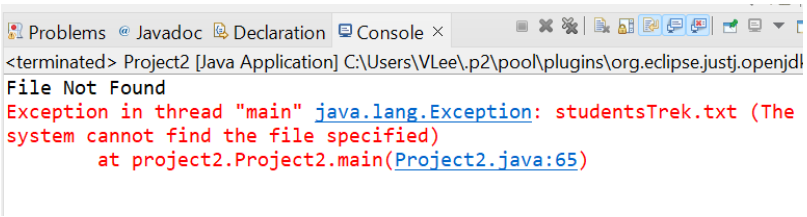
Test 4 (Created):

Note: I am testing for wrong file name or no files. I purposely changed a different file name ".txt" for one of them to throw and exception for "studentsTrek.txt" file if the file is not found. It will not throw an exception for "students.txt" as the file is found.

Input:
Change in Main:
Path path1 = Paths.get("src\\project2\\students.txt");
Path path2 = Paths.get("src\\project2\\studentsTrekThrowException.txt");

Output:
File Not Found
Exception in thread "main" java.lang.Exception: studentsTrek.txt (The system cannot find the file specified)
        at project2.Project2.main(Project2.java:65)

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk
File Not Found
Exception in thread "main" java.lang.Exception: studentsTrek.txt (The
system cannot find the file specified)
	at project2.Project2.main(Project2.java:65)
```

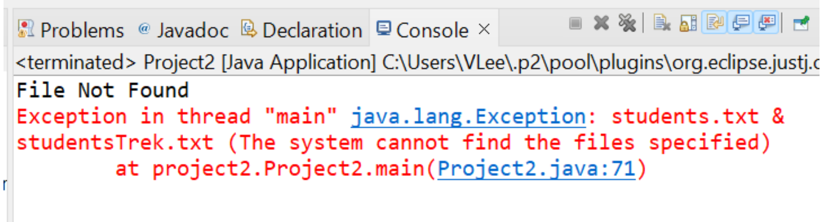
Test 5 (Created):

Note: I am testing for wrong file name or no files. I purposely changed a different file name ".txt" for both of them to throw and exception for "students.txt" and "studentsTrek.txt"  files if both files are not found.

Input:
Change in Main:
Path path1 = Paths.get("src\\project2\\studentsThrowException.txt");
Path path2 = Paths.get("src\\project2\\studentsTrekThrowException.txt");

Output:
File Not Found
Exception in thread "main" java.lang.Exception: students.txt & studentsTrek.txt (The system cannot find the files specified)
        at project2.Project2.main(Project2.java:71)

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.c
File Not Found
Exception in thread "main" java.lang.Exception: students.txt &
studentsTrek.txt (The system cannot find the files specified)
	at project2.Project2.main(Project2.java:71)
```

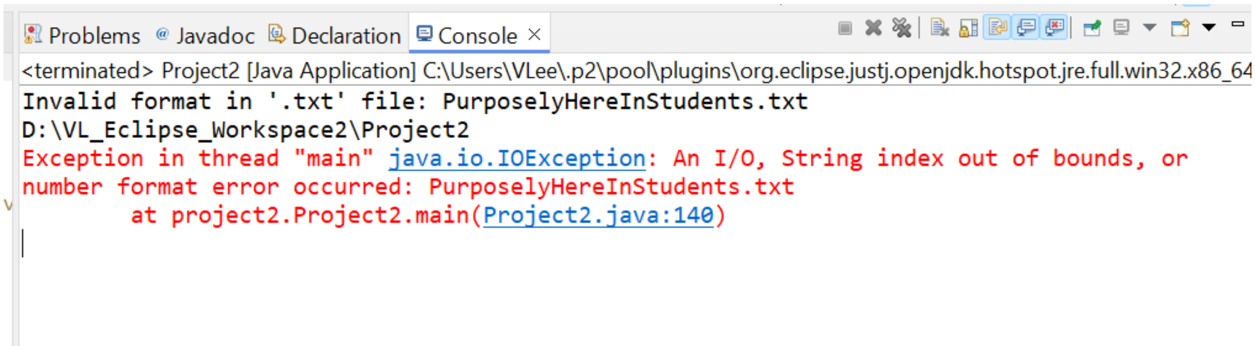
Test 6 (Created):

Note: I am testing for invalid data in the file. I purposely entered an invalid data into the “.txt” to throw and exception for “students.txt” when there are not four data in the file: name, grade1, grade2, level. It should tell what the data is invalid in the file and what exception is it. I made sure to include the I/O, String index out of bounds, and number format errors. And If both files have data invalid, it should output the exception error same as this case.

Input:
Change in “students.txt”:
Brown,William 72 230 Junior
Johnson,Mary 21 77 Masters
Jones,Sally 32 95 Sophomore
PurposelyHereInStudents.txt

Output:
Invalid format in '.txt' file: PurposelyHereInStudents.txt
D:\VL_Eclipse_Workspace2\Project2
Exception in thread "main" java.io.IOException: An I/O, String index out of bounds, or number format error occurred: PurposelyHereInStudents.txt
	at project2.Project2.main(Project2.java:140)

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
Invalid format in '.txt' file: PurposelyHereInStudents.txt
D:\VL_Eclipse_Workspace2\Project2
Exception in thread "main" java.io.IOException: An I/O, String index out of bounds, or
number format error occurred: PurposelyHereInStudents.txt
	at project2.Project2.main(Project2.java:140)
```

Test 7 (Created):

Note: I am testing for invalid data in the file. I purposely entered an invalid data into the “.txt” to throw and exception for “studentsTrek.txt” when there are not four data in the file: name, grade1, grade2, level. It should tell what the data is invalid in the file and what exception is it. I made sure to include the I/O, String index out of bounds, and number format errors. The first file should output the correct answer sample. The second file should output the exception error due to the invalid data in the file.

Input:
Change in “studentsTrek.txt”:
Sarek 105 380 Doctorate

```
Tuvok 110 410 Doctorate
Neelix 60 169 Doctorate
Martok 80 300 Doctorate
Bashir,Julian 80 300 Masters
SevenOfNine 95 207 Masters
Kirk,James 90 360 Junior
Sisko,Benjamin 110 400 Junior
Archer,Jonathan 95 350 Junior
Data 75 300 Junior
LaForge,Geordi 70 196 Junior
O'Brien,Miles 65 182 Junior
Kira,Nerys 75 210 Junior
Odo 50 140 Junior
Spock 45 126 Senior
Picard,Jean-Luc 120 336 Senior
Janeway,Kathryn 105 294 Senior
Riker,William 85 238 Senior
Worf 60 168 Senior
Dax,Jadzia 70 280 Senior
Troi,Deanna 40 112 Sophomore
Quark 30 84 Sophomore
Garak 30 84 Sophomore
Kim,Harry 25 70 Freshman
Crusher,Wesley 20 80 Freshman
PurposelyHereInStudents.txt
```

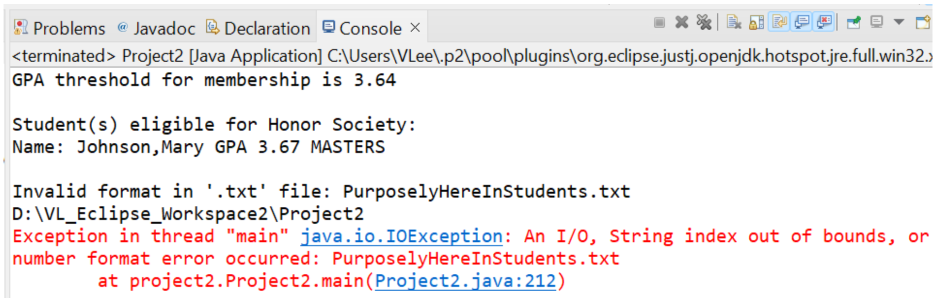
Output:

```
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

Invalid format in '.txt' file: PurposelyHereInStudents.txt
D:\VL_Eclipse_Workspace2\Project2
Exception in thread "main" java.io.IOException: An I/O, String index out of bounds, or number
format error occurred: PurposelyHereInStudents.txt
        at project2.Project2.main(Project2.java:212)
```

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

Invalid format in '.txt' file: PurposelyHereInStudents.txt
D:\VL_Eclipse_Workspace2\Project2
Exception in thread "main" java.io.IOException: An I/O, String index out of bounds, or
number format error occurred: PurposelyHereInStudents.txt
        at project2.Project2.main(Project2.java:212)
```

Test 8 (Created):

Note: I am testing different GPA ranges for eligible students for honor society. Using a mix of low and high numbers, I purposely changed the GPA threshold to 3.0 for file1/path1 and 3.8 for file2/path2. It should print out the students eligible for Honor Society.

Input:
Manually set the GPA threshold for Honor Society eligibility in main:
Student.setGpaThreshold(3.0);
Student.setGpaThreshold(3.8);
File paths of both path1 and path2 are in project2 package; project 2 package is in source;
The "src\\project2\\students.txt" & "src\\project2\\studentsTrek.txt" must be in project2 package

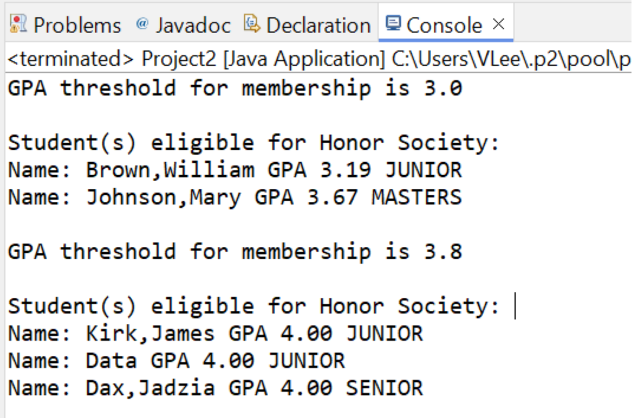
Output:
GPA threshold for membership is 3.0

Student(s) eligible for Honor Society:
Name: Brown,William GPA 3.19 JUNIOR
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 3.8

Student(s) eligible for Honor Society:
Name: Kirk,James GPA 4.00 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\p
GPA threshold for membership is 3.0

Student(s) eligible for Honor Society:
Name: Brown,William GPA 3.19 JUNIOR
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 3.8

Student(s) eligible for Honor Society:
Name: Kirk,James GPA 4.00 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR
```

Test 9 (Created):

Note: I am testing different GPA ranges for eligible students for honor society. Using a mix of low and high numbers, I purposely changed the GPA threshold to 2.0 for file1/path1 and 4.0 for file2/path2. It should print out the students eligible for Honor Society.

Input:
Manually set the GPA threshold for Honor Society eligibility in main:
Student.setGpaThreshold(2.0);
Student.setGpaThreshold(4.0);
File paths of both path1 and path2 are in project2 package; project 2 package is in source;
The "src\\project2\\students.txt" & "src\\project2\\studentsTrek.txt" must be in project2 package

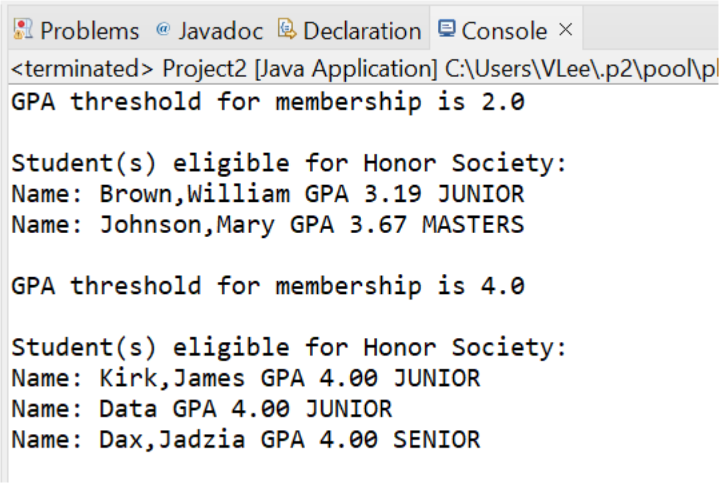
Output:
GPA threshold for membership is 2.0

Student(s) eligible for Honor Society:
Name: Brown,William GPA 3.19 JUNIOR
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 4.0

Student(s) eligible for Honor Society:
Name: Kirk,James GPA 4.00 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\pl
GPA threshold for membership is 2.0

Student(s) eligible for Honor Society:
Name: Brown,William GPA 3.19 JUNIOR
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 4.0

Student(s) eligible for Honor Society:
Name: Kirk,James GPA 4.00 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR
```

Test 10 (Created):

Note: I am testing if there are no students eligible for honor society. Using a mix of low and high numbers, I purposely changed the GPA threshold to 4.0 for file1/path1 and 2.6 for file2/path2. It should print out the students eligible for Honor Society. The first file1/path1 should not have any students eligible and it should print out the flag notification. The file2/path2 should have students eligible and print out according to the demo.

Input:
Manually set the GPA threshold for Honor Society eligibility in main:
Student.setGpaThreshold(2.0);
Student.setGpaThreshold(4.0);
File paths of both path1 and path2 are in project2 package; project 2 package is in source;
The "src\\project2\\students.txt" & "src\\project2\\studentsTrek.txt" must be in project2 package

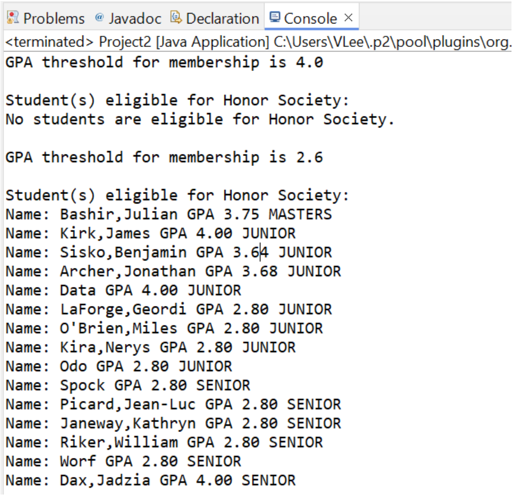
Output:
GPA threshold for membership is 4.0

Student(s) eligible for Honor Society:
No students are eligible for Honor Society.

GPA threshold for membership is 2.6

Student(s) eligible for Honor Society:
Name: Bashir,Julian GPA 3.75 MASTERS
Name: Kirk,James GPA 4.00 JUNIOR
Name: Sisko,Benjamin GPA 3.64 JUNIOR
Name: Archer,Jonathan GPA 3.68 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: LaForge,Geordi GPA 2.80 JUNIOR
Name: O'Brien,Miles GPA 2.80 JUNIOR
Name: Kira,Nerys GPA 2.80 JUNIOR
Name: Odo GPA 2.80 JUNIOR
Name: Spock GPA 2.80 SENIOR
Name: Picard,Jean-Luc GPA 2.80 SENIOR
Name: Janeway,Kathryn GPA 2.80 SENIOR
Name: Riker,William GPA 2.80 SENIOR
Name: Worf GPA 2.80 SENIOR
Name: Dax,Jadzia GPA 4.00 SENIOR

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.
GPA threshold for membership is 4.0

Student(s) eligible for Honor Society:
No students are eligible for Honor Society.

GPA threshold for membership is 2.6

Student(s) eligible for Honor Society:
Name: Bashir,Julian GPA 3.75 MASTERS
Name: Kirk,James GPA 4.00 JUNIOR
Name: Sisko,Benjamin GPA 3.64 JUNIOR
Name: Archer,Jonathan GPA 3.68 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: LaForge,Geordi GPA 2.80 JUNIOR
Name: O'Brien,Miles GPA 2.80 JUNIOR
Name: Kira,Nerys GPA 2.80 JUNIOR
Name: Odo GPA 2.80 JUNIOR
Name: Spock GPA 2.80 SENIOR
Name: Picard,Jean-Luc GPA 2.80 SENIOR
Name: Janeway,Kathryn GPA 2.80 SENIOR
Name: Riker,William GPA 2.80 SENIOR
Name: Worf GPA 2.80 SENIOR
Name: Dax,Jadzia GPA 4.00 SENIOR
```

Test 11 (Created):

Note: I am testing an exception. Using a mix of low and high numbers, I purposely changed the GPA threshold to -1 for file1/path1 and 3.59 for file2/path2. It should print out the students eligible for Honor Society. The first file1/path1 should print out the throw IllegalArgumentException from the Student setGpaThreshold as the GPA threshold is a negative number or not at least 0. The file2/path2 should have students eligible and print out according to the demo.

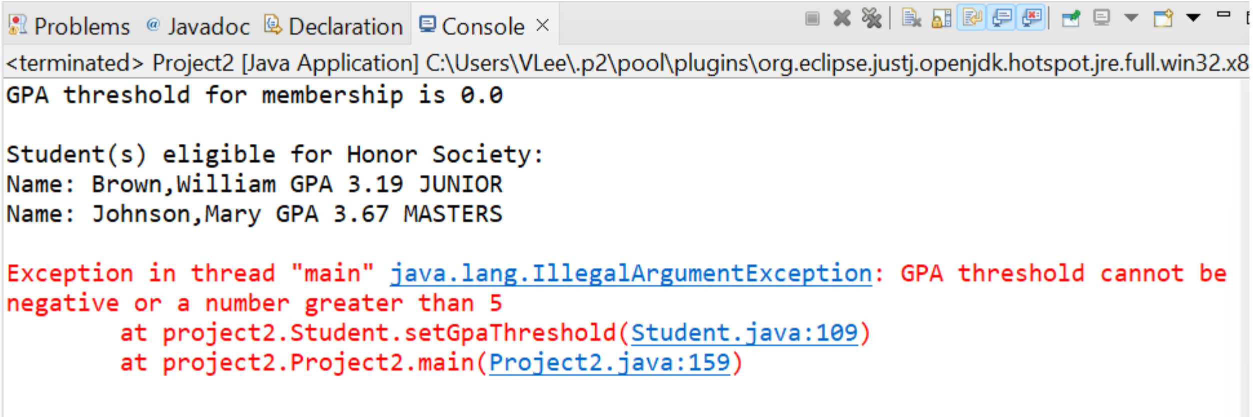
Input:
Manually set the GPA threshold for Honor Society eligibility in main:
Student.setGpaThreshold(-2.0);
Student.setGpaThreshold(3.59);
File paths of both path1 and path2 are in project2 package; project 2 package is in source;
The "src\\project2\\students.txt" & "src\\project2\\studentsTrek.txt" must be in project2 package

Output:
GPA threshold for membership is 0.0

Student(s) eligible for Honor Society:
Name: Brown,William GPA 3.19 JUNIOR
Name: Johnson,Mary GPA 3.67 MASTERS

Exception in thread "main" java.lang.IllegalArgumentException: GPA threshold cannot be negative or a number greater than 5
	at project2.Student.setGpaThreshold(Student.java:109)
	at project2.Project2.main(Project2.java:159)

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
GPA threshold for membership is 0.0

Student(s) eligible for Honor Society:
Name: Brown,William GPA 3.19 JUNIOR
Name: Johnson,Mary GPA 3.67 MASTERS

Exception in thread "main" java.lang.IllegalArgumentException: GPA threshold cannot be
negative or a number greater than 5
        at project2.Student.setGpaThreshold(Student.java:109)
        at project2.Project2.main(Project2.java:159)
```

Test 12 (Created):

Note: I am testing an exception. Using a mix of low and high numbers, I purposely changed the GPA threshold to 3.64 for file1/path1 and 6.0 for file2/path2. It should print out the students eligible for Honor Society. The first file1/path1 should have students eligible and print out according to the demo.The second file2/path2 should print out the throw IllegalArgumentException from the Student setGpaThreshold as the GPA threshold is greater than 5.0 which in reality there is no such thing as a 5.0 GPA. There is a possibility of a GPA greater than 4.0 which is why I did not put the max GPA cap at 4.0.

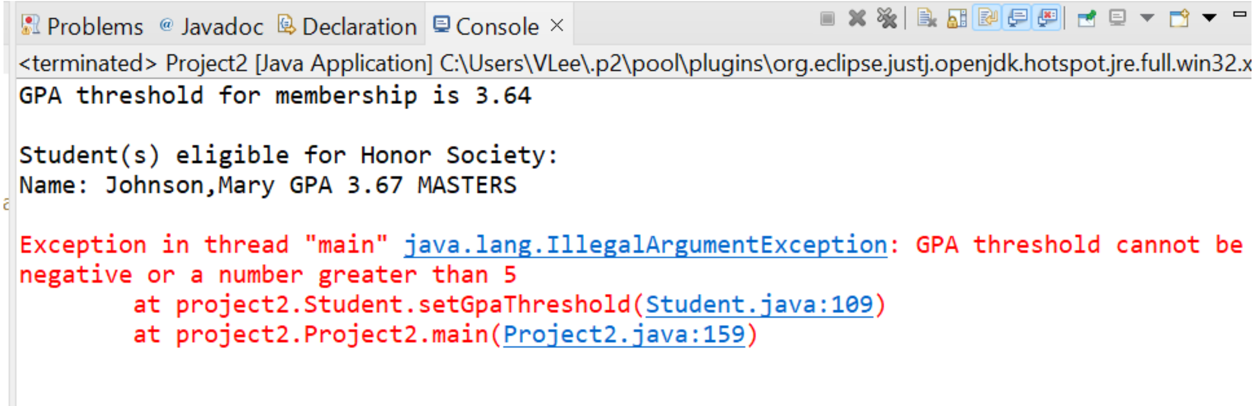
Input:
Manually set the GPA threshold for Honor Society eligibility in main:
Student.setGpaThreshold(3.64);
Student.setGpaThreshold(6.0);
File paths of both path1 and path2 are in project2 package; project 2 package is in source;
The "src\\project2\\students.txt" & "src\\project2\\studentsTrek.txt" must be in project2 package

Output:
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

Exception in thread "main" java.lang.IllegalArgumentException: GPA threshold cannot be negative or a number greater than 5
	at project2.Student.setGpaThreshold(Student.java:109)
	at project2.Project2.main(Project2.java:159)

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

Exception in thread "main" java.lang.IllegalArgumentException: GPA threshold cannot be
negative or a number greater than 5
        at project2.Student.setGpaThreshold(Student.java:109)
        at project2.Project2.main(Project2.java:159)
```

Test 13 (Created):

Note: I am testing upper and lowercase when reading file. I purposely changed a data level "students.txt" from "Johnson,Mary 21 77 Masters" to "Johnson,Mary 21 77 masters" and other versions. I am testing the ignore upper and lower case. It should still print out the correct output when GPA threshold is 3.64. I have done this three times and so the demo image should be the same output for all three. (It should be the same image as test 1 and 2 as I did not change GPA.)

Input:
Manually set the GPA threshold for Honor Society eligibility in main:
Student.setGpaThreshold(3.64);
Student.setGpaThreshold(3.59);

Change in "students.txt":
Brown,William 72 230 Junior
Johnson,Mary 21 77 masters
Jones,Sally 32 95 Sophomore

Change in "students.txt":
Brown,William 72 230 Junior
Johnson,Mary 21 77 MASTERS
Jones,Sally 32 95 Sophomore

Change in "students.txt":
Brown,William 72 230 Junior
Johnson,Mary 21 77 mAsTeRs
Jones,Sally 32 95 Sophomore

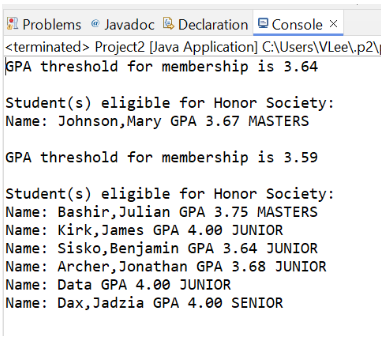
Output:
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 3.59

Student(s) eligible for Honor Society:
Name: Bashir,Julian GPA 3.75 MASTERS
Name: Kirk,James GPA 4.00 JUNIOR
Name: Sisko,Benjamin GPA 3.64 JUNIOR
Name: Archer,Jonathan GPA 3.68 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR

Demo test below:



```
<terminated> Project2 [Java Application] C:\Users\VLee\.p2\
GPA threshold for membership is 3.64

Student(s) eligible for Honor Society:
Name: Johnson,Mary GPA 3.67 MASTERS

GPA threshold for membership is 3.59

Student(s) eligible for Honor Society:
Name: Bashir,Julian GPA 3.75 MASTERS
Name: Kirk,James GPA 4.00 JUNIOR
Name: Sisko,Benjamin GPA 3.64 JUNIOR
Name: Archer,Jonathan GPA 3.68 JUNIOR
Name: Data GPA 4.00 JUNIOR
Name: Dax,Jadzia GPA 4.00 SENIOR
```

## Lessons Learned (brief paragraphs):

I learned how to create a project, classes, subclasses with extend, try/catch, throw exceptions, override concept, methods, and functions to achieve my project goals. I learned to create objects for a solution design in Project 2 with some methods to perform on the attributes and classes which is a description of an object. An object is an instance of a class. A class defines all the attributes which an object can have and methods, which defines the functionality of the object. A subclass is a class that extends to another class (Extends), thereby inheriting its properties and behaviors (super). I learned more about encapsulation of important data such as information of an object and restricting access of the data and methods. I learned more about inheritance of classes and throwing error messages. I also learned polymorphism which provides a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name. For Project2 class utilizes Students, Undergraduate, and Graduate classes to read both files given in ".txt", create a student, calculate the data for GPA, and print out the eligible students for honor society based on the rubric and requirements. I utilize the import java Files, Paths, Array List, and Stream to read the file and gather the student's information for each object created and added to the Student Array List. I created a constructor and methods for all classes including calculating the GPA and setting the GPA threshold. I created methods and an override string for printing out the eligible students for honor society and the information based on Undergraduate v. Graduate. It should calculate the GPA and classify each student. Overall, I learned to apply it to project 2 with the lessons about try/catch/exceptions/throws, classes, subclasses, packages, importing libraries, constructors, toString(), returning, private vs public, static vs non-static, Array List, Objects, Arrays, and File/Path java import.

My design approach was to create the Student, Undergraduate, and Graduate classes first before creating the Project2 class. I started with a Bottom-Up Design when building the code, but then debugged the code through a Top-Down Design. I followed the instructions on what is asked for the Student, Undergraduate, and Graduate class. I utilized the lessons from the chapters that were in the past few weeks so that I can apply it to the Student, Undergraduate, and Graduate classes. Once it was finished, I went back into Project2 class to create the Student students Array List and pass it to the methods in the classes to print out the eligible students for honor society. It should add each student based on the subclass that classifies each student. I also made sure to check if a file does not exist, it must throw an exception as learned in the lessons. To debug the Project2, I ended up looking at the Sample outputs or lessons learned and then modified the classes. I also adjust the classes so that the eligibleForHonorSociety() and toString() would override each other for when a student is undergraduate or graduate. Then, I went back to the Project2 class and checked if the output was correct.

**(If necessary, my whole Javadoc to word is below)**

# Project 2 Javadoc Combined Document

Author: Victoria Lee

## Table of Contents

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# All Classes and Interfaces

Classes

Class

Description

Graduate

Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the program level; Override of toString to prints out Graduates's information

Project2

Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out students;

Student

Constructs Student object by information from file; Calculates GPA; Method for a student is eligible for Honor Society based on the set GPA threshold given; Prints out the Student's information from toString

Undergraduate

Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the Year; Override of toString to prints out Undergraduate's information

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# All Packages

Package Summary

Package

Description

project2

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Help:
- Navigation |
- Pages

SEARCH:

# JavaDoc Help

- Navigation:
  - Search
- Kinds of Pages:
  - Package
  - Class or Interface
  - Other Files
  - Use
  - Tree (Class Hierarchy)
  - All Packages
  - All Classes and Interfaces
  - Index

---

## Navigation

Starting from the Overview page, you can browse the documentation using the links in each page, and in the navigation bar at the top of each page. The Index and Search box allow you to navigate to specific declarations and summary pages, including: All Packages, All Classes and Interfaces

### Search

You can search for definitions of modules, packages, types, fields, methods, system properties and other terms defined in the API, using some or all of the name, optionally using "camelCase" abbreviations. For example:

- `j.l.obj` will match "java.lang.Object"
- `InpStr` will match "java.io.InputStream"
- `HM.cK` will match "java.util.HashMap.containsKey(Object)"

Refer to the Javadoc Search Specification for a full description of search features.

---

## Kinds of Pages

The following sections describe the different kinds of pages in this collection.

### Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. These pages may contain the following categories:

- Interfaces
- Classes
- Enum Classes
- Exceptions
- Errors
- Annotation Interfaces

### Class or Interface

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a declaration and description, member summary tables, and detailed member descriptions. Entries in each of these sections are omitted if they are empty or not applicable.

- Class Inheritance Diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class or Interface Declaration
- Class or Interface Description

- Nested Class Summary
- Enum Constant Summary
- Field Summary
- Property Summary
- Constructor Summary
- Method Summary
- Required Element Summary
- Optional Element Summary

- Enum Constant Details
- Field Details
- Property Details
- Constructor Details
- Method Details
- Element Details

Note: Annotation interfaces have required and optional elements, but not methods. Only enum classes have enum constants. The components of a record class are displayed as part of the declaration of the record class. Properties are a feature of JavaFX.

The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

## Other Files

Packages and modules may contain pages with additional information related to the declarations nearby.

## Use

Each documented package, class and interface has its own Use page. This page describes what packages, classes, methods, constructors and fields use any part of the given class or package. Given a class or interface A, its Use page includes subclasses of A, fields declared as A, methods that return A, and methods and constructors with parameters of type A. You can access this page by first going to the package, class or interface, then clicking on the USE link in the navigation bar.

## Tree (Class Hierarchy)

There is a Class Hierarchy page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. Classes are organized by inheritance structure starting with `java.lang.Object`. Interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on TREE displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking on TREE displays the hierarchy for only that package.

## All Packages

The All Packages page contains an alphabetic index of all packages contained in the documentation.

## All Classes and Interfaces

The All Classes and Interfaces page contains an alphabetic index of all classes and interfaces contained in the documentation, including annotation interfaces, enum classes, and record classes.

## Index

The Index contains an alphabetic index of all classes, interfaces, constructors, methods, and fields in the documentation, as well as summary pages such as All Packages, All Classes and Interfaces.

---

This help file applies to API documentation generated by the standard doclet.

JavaScript is disabled on your browser.

project2/package-summary.html

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Hierarchy For All Packages

Package Hierarchies:

- project2

## Class Hierarchy

- java.lang.Object
  - project2.Project2
  - project2.Student
    - project2.Graduate
    - project2.Undergraduate

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## C

**calculateGpa() - Method in class project2.Student**
double calculateGpa() for Student Object; GPA calculation based on qualityPoints and creditHours given; Outputs: GPA: {}

**creditHours - Variable in class project2.Student**
CreditHours of the student in Integer format read from file for GPA calculation

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## E

**eligibleForHonorSociety() - Method in class project2.Graduate**
Override with eligibleForHonorSociety; Outputs: Name: {} Age: {} Program: {};

**eligibleForHonorSociety() - Method in class project2.Student**
Boolean eligibleForHonorSociety for Student Object/Information; Outputs: True/False

**eligibleForHonorSociety() - Method in class project2.Undergraduate**
Override with eligibleForHonorSociety; Outputs: Name: {} GPA: {} Year: {};

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## G

**gpaThreshold - Static variable in class project2.Student**
Double gpaThreshold of file user manual input/set for the minimum GPA threshold eligibleForHonorSociety

**Graduate - Class in project2**
Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the program level; Override of toString to prints out Graduates's information

**Graduate(String, int, int, String) - Constructor for class project2.Graduate**
Constructs Graduate object by information from file with name, creditHours, qualityPoints, and program; Imports Student class with Super

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## L

**level - Variable in class project2.Student**

Level of the student in String format from file for year of student or degree program

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## M

**main(String[]) - Static method in class project2.Project2**

Project2 uses Student, Undergraduate, Graduate classes; Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out students; has try/catch, else for invalid inputs; It has a try/catch 2x because of 2 different files in ".txt" format

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use

- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## N

**name - Variable in class project2.Student**
Name of the student in string format from file for student name

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## P

**program - Variable in class project2.Graduate**
String program of file for for program of student such as B.S., M.S., PhD.

**project2 - package project2**

**Project2 - Class in project2**

Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out students;

**Project2() - Constructor for class project2.Project2**

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## Q

**qualityPoints - Variable in class project2.Student**

QualityPoints of the student in Integer format from file for GPA calculation

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package

- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## S

**setGpaThreshold(double) - Static method in class project2.Student**
Void method of setGPAThreshold sets up the minimum GPA to be considered for honor society; User will manually set it through the main function;

**Student - Class in project2**
Constructs Student object by information from file; Calculates GPA; Method for a student is eligible for Honor Society based on the set GPA threshold given; Prints out the Student's information from toString

**Student(String, int, int, String) - Constructor for class project2.Student**
Constructs Student object by information from file with name, creditHours, qualityPoints, and level.

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## T

**toString() - Method in class project2.Graduate**

Override with toString() from Student class; Outputs: Name: {} GPA: {} Program: {}

**toString() - Method in class project2.Student**

String toString() for Student Object; Outputs: Name: {} GPA: {}

**toString() - Method in class project2.Undergraduate**

Override with eligibleForHonorSociety from Student class; Outputs: Name: {} GPA: {} Year: {};

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## U

**Undergraduate - Class in project2**
Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the Year; Override of toString to prints out Undergraduate's information

**Undergraduate(String, int, int, String) - Constructor for class project2.Undergraduate**
Constructs Undergraduate object by information from file with name, creditHours, qualityPoints, and year; Imports Student class with Super

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Index

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

## Y

**year - Variable in class project2.Undergraduate**
String year of file for for year of student such as Freshman, Sophomore, Junior, or Senior

C E G L M N P Q S T U Y
All Classes and Interfaces|All Packages

JavaScript is disabled on your browser.

Skip navigation links

- Package

- Class
- Use
- Tree
- Index
- Help
- Summary:
- Nested |
- Field |
- Constr |
- Method
- Detail:
- Field |
- Constr |
- Method

SEARCH:

Package project2

# Class Graduate

java.lang.Object

project2.Student

project2.Graduate

---

public class Graduate extends Student

Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the program level; Override of toString to prints out Graduates's information

Course: CMSC 215

Date: 6/10/2024

Project: Project 2

**Version:**
JRE17

**Author:**
Victoria Lee

## Field Summary

Fields

Modifier and Type

Field

Description

`private String`

`program`

String program of file for for program of student such as B.S., M.S., PhD.

### Fields inherited from class project2.Student

`creditHours, gpaThreshold, level, name, qualityPoints`

## Constructor Summary

Constructors

Constructor

Description

`Graduate(String name, int creditHours, int qualityPoints, String program)`

Constructs Graduate object by information from file with name, creditHours, qualityPoints, and program; Imports Student class with Super

## Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method

Description

`boolean`

`eligibleForHonorSociety()`

Override with eligibleForHonorSociety; Outputs: Name: {} Age: {} Program: {};

`String`

`toString()`

Override with toString() from Student class; Outputs: Name: {} GPA: {} Program: {}

### Methods inherited from class project2.Student

`calculateGpa, setGpaThreshold`

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Field Details

### program

private String program

String program of file for for program of student such as B.S., M.S., PhD.

## Constructor Details

### Graduate

public Graduate(String name, int creditHours, int qualityPoints, String program)

Constructs Graduate object by information from file with name, creditHours, qualityPoints, and program; Imports Student class with Super

**Parameters:**

name - String name (for student name)

creditHours - Integer creditHours (for GPA calculation)

qualityPoints - Integer qualityPoints (for GPA calculation)

program - String program (for degree program)

## Method Details

### eligibleForHonorSociety

public boolean eligibleForHonorSociety()

Override with eligibleForHonorSociety; Outputs: Name: {} Age: {} Program: {};

**Overrides:**

eligibleForHonorSociety in class Student

**Returns:**

string+value it outputs graduate information

### toString

public String toString()

Override with toString() from Student class; Outputs: Name: {} GPA: {} Program: {}

**Overrides:**

toString in class Student

**Returns:**

string+value it outputs undergraduate information, GPA with calculateGpa() with %.2f is for 2 decimal places, and program with toUpperCase() for all CAPS.

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Package:
- Description |
- Related Packages |
- Classes and Interfaces

SEARCH:

# Package project2

package project2

- Classes

  Class

  Description

  Graduate

  Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the program level; Override of toString to prints out Graduates's information

  Project2

  Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out students;

  Student

Constructs Student object by information from file; Calculates GPA; Method for a student is eligible for Honor Society based on the set GPA threshold given; Prints out the Student's information from toString

Undergraduate

Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the Year; Override of toString to prints out Undergraduate's information

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Hierarchy For Package project2

## Class Hierarchy

- java.lang.Object
    - project2.Project2
    - project2.Student
        - project2.Graduate
        - project2.Undergraduate

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Uses of Package project2

- Classes in project2 used by project2

  Class

  Description

  Student

  Constructs Student object by information from file; Calculates GPA; Method for a student is eligible for Honor Society based on the set GPA threshold given; Prints out the Student's information from toString

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Summary:
- Nested |
- Field |
- Constr |
- Method
- Detail:
- Field |
- Constr |
- Method

SEARCH:

Package project2

# Class Project2

java.lang.Object

project2.Project2

---

public class Project2 extends Object

Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out students;

Course: CMSC 215

Date: 6/10/2024

Project: Project 2

**Version:**
JRE17

**Author:**
Victoria Lee

- ## Constructor Summary

  Constructors
  Constructor
  Description
  `Project2()`

- ## Method Summary

  All Methods
  Static Methods
  Concrete Methods
  Modifier and Type
  Method
  Description
  `static void`
  `main(String[] args)`
  Project2 uses Student, Undergraduate, Graduate classes; Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out

students; has try/catch, else for invalid inputs; It has a try/catch 2x because of 2 different files in ".txt" format

### Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait
```

## Constructor Details

### Project2

public Project2()

## Method Details

### main

public static void main(String[] args) throws Exception

Project2 uses Student, Undergraduate, Graduate classes; Takes Student's information from file, separated by an array; Student object added to an ArrayList based on Undergraduate/Graduate; Calculates GPA for honors and prints out students; has try/catch, else for invalid inputs; It has a try/catch 2x because of 2 different files in ".txt" format

**Parameters:**

`args` - the command line arguments for main method

**Throws:**

`Exception` - if the user input is incorrect with not 4 items on one line

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Summary:
- Nested |
- Field |
- Constr |
- Method
- Detail:

- Field |
- Constr |
- Method

SEARCH:

Package project2

# Class Student

java.lang.Object

project2.Student

**Direct Known Subclasses:**
`Graduate`, `Undergraduate`

---

public class Student extends Object

Constructs Student object by information from file; Calculates GPA; Method for a student is eligible for Honor Society based on the set GPA threshold given; Prints out the Student's information from toString

Course: CMSC 215

Date: 6/10/2024

Project: Project 2

**Version:**
JRE17

**Author:**
Victoria Lee

- ## Field Summary

  Fields
  Modifier and Type
  Field
  Description
  `protected int`
  `creditHours`
  CreditHours of the student in Integer format read from file for GPA calculation

`protected static double`

`gpaThreshold`

Double gpaThreshold of file user manual input/set for the minimum GPA threshold eligibleForHonorSociety

`protected String`

`level`

Level of the student in String format from file for year of student or degree program

`protected String`

`name`

Name of the student in string format from file for student name

`protected int`

`qualityPoints`

QualityPoints of the student in Integer format from file for GPA calculation

## Constructor Summary

Constructors

Constructor

Description

`Student(String name, int creditHours, int qualityPoints, String level)`

Constructs Student object by information from file with name, creditHours, qualityPoints, and level.

## Method Summary

All Methods

Static Methods

Instance Methods

Concrete Methods

Modifier and Type

Method

Description

`double`

`calculateGpa()`

double calculateGpa() for Student Object; GPA calculation based on qualityPoints and creditHours given; Outputs: GPA: {}

`boolean`

`eligibleForHonorSociety()`

Boolean eligibleForHonorSociety for Student Object/Information; Outputs: True/False

`static void`

`setGpaThreshold(double threshold)`

Void method of setGPAThreshold sets up the minimum GPA to be considered for honor society; User will manually set it through the main function;

`String`

`toString()`

String toString() for Student Object; Outputs: Name: {} GPA: {}

### Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait,
wait, wait
```

- ## Field Details

  - ### name

    protected String name

    Name of the student in string format from file for student name

  - ### creditHours

    protected int creditHours

    CreditHours of the student in Integer format read from file for GPA calculation

  - ### qualityPoints

    protected int qualityPoints

    QualityPoints of the student in Integer format from file for GPA calculation

  - ### level

    protected String level

    Level of the student in String format from file for year of student or degree program

  - ### gpaThreshold

    protected static double gpaThreshold

    Double gpaThreshold of file user manual input/set for the minimum GPA threshold eligibleForHonorSociety

- ## Constructor Details

  - ### Student

    public Student(String name, int creditHours, int qualityPoints, String level)

    Constructs Student object by information from file with name, creditHours, qualityPoints, and level.

    **Parameters:**

    `name` - String name (for student name)

`creditHours` - Integer creditHours (for GPA calculation)
`qualityPoints` - Integer qualityPoints (for GPA calculation)
`level` - String level (for year of student or degree program)

## Method Details

### calculateGpa

public double calculateGpa()
double calculateGpa() for Student Object; GPA calculation based on qualityPoints and creditHours given; Outputs: GPA: {}

**Returns:**

GpaValue it outputs GPA by dividing qualityPoints / creditHours

### eligibleForHonorSociety

public boolean eligibleForHonorSociety()
Boolean eligibleForHonorSociety for Student Object/Information; Outputs: True/False

**Returns:**

String+BooleanValue it outputs True or False based on GPA and Threshold given

### toString

public String toString()
String toString() for Student Object; Outputs: Name: {} GPA: {}

**Overrides:**

`toString` in class `Object`

**Returns:**

String+GpaValue it outputs Student name and GPA from calculateGpa() method and %.2f is for 2 decimal places

### setGpaThreshold

public static void setGpaThreshold(double threshold)
Void method of setGPAThreshold sets up the minimum GPA to be considered for honor society; User will manually set it through the main function;

**Parameters:**

`threshold` - it sets the user manually input of the gpaThreshold for honor society requirements with static double gpaThreshold

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help
- Summary:
- Nested |
- Field |
- Constr |
- Method
- Detail:
- Field |
- Constr |
- Method

SEARCH:

Package project2

# Class Undergraduate

java.lang.Object

project2.Student

project2.Undergraduate

public class Undergraduate extends Student

Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the Year; Override of toString to prints out Undergraduate's information

Course: CMSC 215

Date: 6/10/2024

Project: Project 2

**Version:**
JRE17

**Author:**
Victoria Lee

## Field Summary

Fields

Modifier and Type

Field

Description

`private String`

`year`

String year of file for for year of student such as Freshman, Sophomore, Junior, or Senior

### Fields inherited from class project2.Student

`creditHours, gpaThreshold, level, name, qualityPoints`

## Constructor Summary

Constructors

Constructor

Description

`Undergraduate(String name, int creditHours, int qualityPoints, String year)`

Constructs Undergraduate object by information from file with name, creditHours, qualityPoints, and year; Imports Student class with Super

## Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method

Description

`boolean`

`eligibleForHonorSociety()`

Override with eligibleForHonorSociety; Outputs: Name: {} GPA: {} Year: {};

`String`

`toString()`

Override with eligibleForHonorSociety from Student class; Outputs: Name: {} GPA: {} Year: {};

### Methods inherited from class project2.Student

`calculateGpa, setGpaThreshold`

### Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

- ## Field Details

  - ### year

    private String year

    String year of file for for year of student such as Freshman, Sophomore, Junior, or Senior

- ## Constructor Details

  - ### Undergraduate

    public Undergraduate(String name, int creditHours, int qualityPoints, String year)

    Constructs Undergraduate object by information from file with name, creditHours, qualityPoints, and year; Imports Student class with Super

    **Parameters:**

    `name` - String name (for student name)

    `creditHours` - Integer creditHours (for GPA calculation)

    `qualityPoints` - Integer qualityPoints (for GPA calculation)

    `year` - String level (for year of student)

- ## Method Details

  - ### eligibleForHonorSociety

    public boolean eligibleForHonorSociety()

    Override with eligibleForHonorSociety; Outputs: Name: {} GPA: {} Year: {};

    **Overrides:**

    `eligibleForHonorSociety` in class `Student`

    **Returns:**

    string+value it outputs undergraduate information

  - ### toString

    public String toString()

Override with eligibleForHonorSociety from Student class; Outputs: Name: {} GPA: {} Year: {};

**Overrides:**

`toString` in class `Student`

**Returns:**

string+value it outputs undergraduate information, GPA with calculateGpa() with %.2f is for 2 decimal places, and year with toUpperCase() for all CAPS.

# Uses of Class project2.Graduate

No usage of project2.Graduate

# Uses of Class project2.Project2

No usage of project2.Project2

JavaScript is disabled on your browser.

Skip navigation links

- Package
- Class
- Use
- Tree
- Index
- Help

SEARCH:

# Uses of Class project2.Student

- ## Uses of Student in project2

  Subclasses of Student in project2

  Modifier and Type

  Class

  Description

  `class`

  `Graduate`

  Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the program level; Override of toString to prints out Graduates's information

  `class`

  `Undergraduate`

  Extends Student's information from file in this class (imports); Override Method for a student is eligible for Honor Society based on GPA threshold from Student and the Year; Override of toString to prints out Undergraduate's information

JavaScript is disabled on your browser.

Skip navigation links

# Uses of Class project2.Undergraduate

No usage of project2.Undergraduate