

---

# Project 2 - Single Student Grade Calculator

Kevin Nilson, <http://www.smccd.net/accounts/nilsonk/>

CS 382 Project 2, Java Programming I, Fall 06

## 1. Assignment

Due - Due November 3

Write a Java Program to calculate grades for a single student.

- Your Java Classes should use the packages `edu.smccd.cis382.fall2006.project2`, `edu.smccd.cis382.fall2006.project2.grabber`, and `edu.smccd.cis382.fall2006.project2.exception`.
- Your program must be compatible with `project2-build.xml`. Do not include your UnitTests in the base directory when running the unit test. I do not want to require configuring location of `junit.jar` when running the build script.

**I expect the following classes for this assignment**

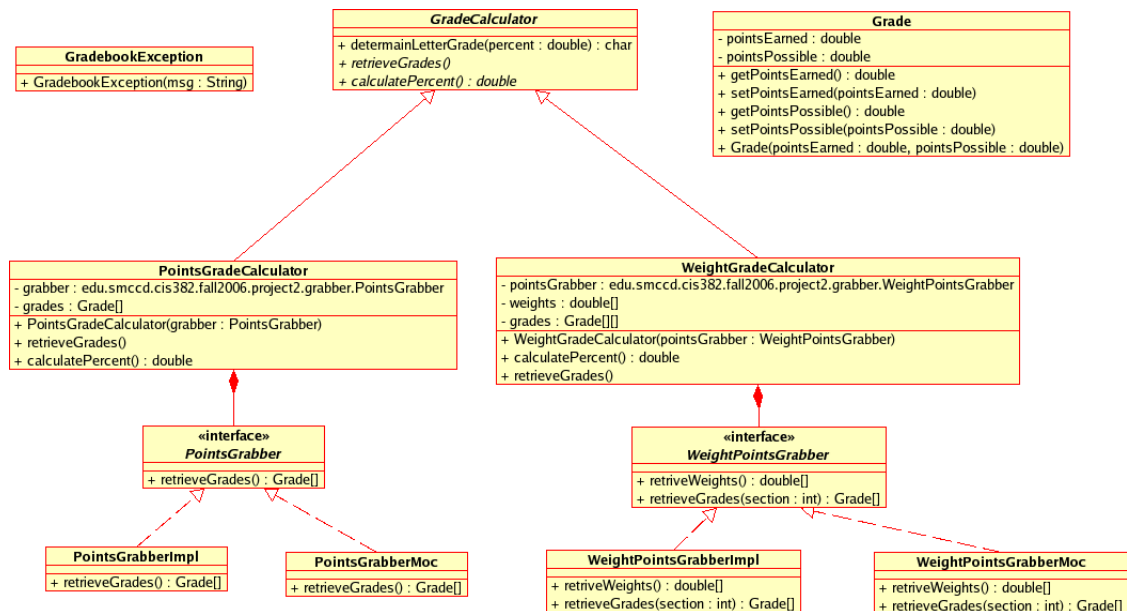
- `GradeCalculator`
- `PointsGradeCalculator`
- `WeightGradeCalculator`
- `Grade`
- `PointsGrabber`
- `PointsGrabberImpl`
- `PointsGrabberMoc`
- `WeightPointsGrabber`
- `WeightPointsGrabberImpl`
- `WeightPointsGrabberMoc`

- GradebookException
- GradeCalculatorTester
- GradeCalculatorTesterMocs
- GradeCalculatorUnitTest

Please review "How to Turn in Assignments" for information about how to submit and other requirements.

## 2. Description of Classes

Figure 1. UML



Class Diagram for Project 2

## 3. GradeCalculator

1. GradeCalculator is an abstract class with abstract methods retrieveGrades and calculatePercent.
2. Letter Grades are calculated as 90 >= A, 80 >= B, 70 >= C, 60 >= D, 60 < F, .
3. PointsGradeCalculator calculates grades by dividing the possible points by the points earned to calculate the percent

4. WeightGradeCalculator calculates grades like we do in CIS 382. Different sections are given weights. For example in CIS 382 the assignments for the final exam are given a weight of 40 and all other assignments are given a weight of 60. If there are no possible points for some of the sections you just ignore that section when calculating the grade. For example currently you have 100 percent in CIS 381 but you have 0 points possible for the final.
5. When the WeightPointsGrabber retrieves the grades for a section the section is one based not zero based. For example in CIS 382 section 1 would be the assignments and section 2 would be the final.
6. GradebookException is thrown when no possible points are available.

## 4. Grabbers

1. Must contain Grabber in the file name.
2. Used to retrieve grades. Several implementations of the interface may exist so that moc objects don't require users to type in values during testing.
3. You may wish to use Scanner to implement.
4. You should prompt the user with a meaningful message before requiring them to type in values.

## 5. Testing

1. GradeCalculatorTester is a standard tester that runs from a main method. This should require the user to type in values for grades on the command line.
2. GradeCalculatorTesterMocs is a standard tester that runs from a main method. The user should not be type in any values when running your unit test. You should use moc objects to perform testing. You should use PointsGrabberMoc and WeightPointsGrabberMoc to retrieve grades. These moc classes should not require any user input.
3. GradeCalculatorUnitTest is a unit test that uses JUnit to test your program. The user should not be type in any values when running your unit test. You should use moc objects to perform testing. You can create additional grabber classes for your testing. I would prefer if these classes were inner classes in the same file as the unit test.

## 6. GradeCalculatorTester

Here is a simple example of what your tester may look like:

```
GradeCalculator[] gradeCalculators=new GradeCalculator[2];

gradeCalculators[0]=new PointsGradeCalculator(new PointsGrabberImpl());
gradeCalculators[1]=new WeightGradeCalculator(new WeightPointsGrabberImpl());

for (int i = 0; i < gradeCalculators.length; i++) {

    gradeCalculators[i].retrieveGrades();

    double percent = gradeCalculators[i].calculatePercent();
    char grade = gradeCalculators[i].determineLetterGrade(percent);

    System.out.println(percent+" = "+grade);

}
```

## 7. Notes

- Due date may be extended.
- The public interface must remain exactly the same as the UML. You may add any additional private methods you wish.

## 8. Updates

- Grade[][] grades added to WeightGradeCalculator. Umbrello just ignored it when I generated the diagram.
- renamed GradeCalculator.determainLetterGrade to GradeCalculator.determineLetterGrade.
- Due date changed to November 1.
- Tester Code added.

## 9. Questions

Post questions to the forum or ask during lab and lecture.