

PA1 OOP Review

[Submit Assignment](#)

Due Oct 7 by 11:59pm **Points** 100 **Submitting** a file upload

Programming Assignment 1: OOP Review

1 Purpose

The primary purpose of this assignment is to help you review (and demonstrate that you have acquired) the knowledge and skills required to program in Java. From a language and algorithms perspective, there is nothing new in this assignment. There is also nothing new from a software engineering perspective.

This assignment will also help you gain some experience with the integrated development environment that you will be using this semester in this course. Some of the tools you will be using this semester will be new to some of you, and we will discuss them as the semester progresses. Regardless of whether you have used them in the past, at this point, you should be able to figure out everything that you need for this assignment.

2 Overview

SagaciousMedia is a (fictitious) company that develops educational hardware, software, and content for both the formal and informal education markets. SagaciousMedia's products are designed to excite and educate students, and to inspire and assist teachers/instructors.

They are in the process of developing a suite of products for working with grading assignments of various kinds (e.g., exams, homework assignments, labs, programming assignments). They have hired you to construct several interfaces/classes that will, ultimately, become part of these products.


These classes/interfaces might be used by the first application they are creating (called **Gradient**) to calculate the numerical grade for a course that has 6 programming assignments (with the lowest dropped) that account for 40% of the course grade, 5 homework assignments that account for 10% of the course grade, one midterm exam that accounts for 20% of the course grade, and a final exam that accounts for 30% of the course grade. For example, assuming the the programming assignments and homework assignments are each graded on a 20-point scale and the exams are each graded on a 100-point scale, the following grades

```
20.0 18.0 5.0 15.0 20.0 20.0 20.0 5.0 0.0 10.0 15.0 80.0 75.0
```

should result in a PA grade of $20.0+18.0+15.0+20.0+20.0=93.0$, a homework grade of $20.0+5.0+0.0+10.0+15.0=50.0$ and a course grade of $0.4\cdot93.0+0.1\cdot50.0+0.2\cdot80.0+0.3\cdot75.0=37.2+5.0+16.0+22.5=80.7$.

3 Preparatory Tasks

Before you do anything else you should:

1. Install the latest version of Eclipse on your computer. (Using other IDEs should be OK for this assignment. But since later in the semester we will practice using Eclipse for debugging, static analysis, and working with Git, it would be better to start using it now.)
2. Read all policies related to homework assignments in the [syllabus](#) .

4 Documents

Sagacious Media uses a software process called Scrum. As a result, the documents they have created are organized in a particular way.

The terms that Sagacious Media uses when discussing Gradient and its functionality are described in the following document.

- [Product Domain Glossary](#) 










The requirements of the first product they are building, **Gradient**, have been organized into what are called *stories*. *Epics* are abstract stories that might take several releases (developed over the course of many months) to completely realize. Each release has its own set of *sprintable stories* that describe its features. Both the epics and the sprintable stories (which are what you are concerned with for this assignment) for Gradient have been collected in the following document.

- [Stories for Gradient](#) 

The existing team at Sagacious Media used the *sprintable* stories to create a set of tasks for this assignment. They completed some of the tasks and have left others for you to complete. The complete set of tasks is contained in the following document.

- [Tasks for Gradient](#) 

The tasks that are "checked" have already been completed by other team members, the evidence for which is contained in the following documents (which are listed in alphabetical order).

- [Engineering Design for Gradient](#) 
- [Implementation of the `Gradient` class](#) 
- [Specification of the `DropFilter` class](#) 
- [Specification of the `Grade` class](#) 
- [Specification of the `Missing` class](#) 
- [Specification of the `SizeException` class](#) 
- [Specification of the `TotalStrategy` class](#) 
- [Specification of the `WeightedTotalStrategy` class](#) 
- [System/Integration Tests for Gradient](#) 

You must complete the tasks that have not been "checked-off".

5 A Recommended Process

The tasks that were identified by the team at Sagacious Media are organized by story. Hence, though they are numbered so that they can be referred to in documents and conversations, the numbers should not, in any way, influence the order in which you complete them. I would suggest you sequence your activities as follows.

1. Read all of the documents carefully. (This is utterly important, which is often ignored, unfortunately.)
2. Review the source code for the `Gradient` class so that you understand how the classes you are implementing will be used.
3. Implement the `Grade` class.
4. Test and debug the `Grade` class.
5. Implement the `Missing` class.
6. Test and debug the `Missing` class.
7. Implement the `SizeException` class.
8. Implement the `GradingStrategy` interface.
9. Think about how the `WeightedTotalStrategy` and `TotalStrategy` classes should be related (if at all).
10. Implement the `WeightedTotalStrategy` or `TotalStrategy` class (whichever one you think should be implemented first).
11. Test and debug the class you just implemented.
12. Implement the `WeightedTotalStrategy` or `TotalStrategy` class (whichever one you think should be implemented second).

13. Test and debug the class you just implemented.
14. Implement the `Filter` interface.
15. Implement the `DropFilter` class.
16. Test and debug the `DropFilter` class.
17. Test and debug the complete system.

Note that, you will need to have a stubbed-out version of all of the classes for your code to compile. This will, hopefully, encourage you to create and use stubs (i.e., pieces of code that stand in for the code that would be needed to provide full functionality) as part of your development process.

6 Getting Help

If you need help with UML syntax, please refer to the following ebook chapters:

- [Object and Classes](http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMDZ0aXRhX2h0bWw) [\(http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMDZ0aXRhX2h0bWw\)](http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMDZ0aXRhX2h0bWw)
- [Packages and Namespaces](http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMDh0aXRhX2h0bWw) [\(http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMDh0aXRhX2h0bWw\)](http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMDh0aXRhX2h0bWw)
- [The Static Model](http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMTB0aXRhX2h0bWw) [\(http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMTB0aXRhX2h0bWw\)](http://proquest.safaribooksonline.com.proxy.lib.csus.edu/book/certification/9780128097830/chapter-16dot-behavior-activity-diagrams/chp016titl_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODAxMjgwOTc4MzAIMkZjaHAwMTB0aXRhX2h0bWw)

7 Submission

Please create a zip file including all your source code, and submit the zip file through Canvas. By default, your latest submission will be graded.

8 Grading

Partial credit will be awarded. Your grade will be based on the percentage of the system tests (worth $7 \times 10 = 70$ points) that your classes pass and on the quality of your code (30 points), as determined by our grader.

Since we have not discussed unit testing yet you will not be penalized **now** if your classes do not conform to the specifications (as long as your system passes the system tests). However, you will be penalized for such defects in the future. Hence, you should strive to write code that conforms to the specifications (and test it as best you can). Also, you should strive to write code that is clear, elegant, and well-documented.

Rule of Thumb: Start Early, Read, Think.