Mid-Semester Project: A How-To for Java
Due July 3

Description
This project acts as a synthesis of everything you will have learned up to the midpoint of the semester. Your goal is to write a "textbook" or how-to guide that demonstrates each of the core ideas this document asks for, both with text explanation and example code. Your textbook/how-to should be themed around something that acts as an anchor for all of your examples, similar to how the online modules return to books and authors as a reference for demonstrating new concepts.

The goal with this work is to better increase your own understanding by taking the time to explain these concepts in a way that makes sense to you. Be creative in the examples that you write and feel free to draw on interests and hobbies you have in your personal life. As you complete each section, ask yourself, "would a newcomer to programming better understand the topic after reading this?"

General Expectations:
- Your final submission should be neat, organized, and professional.
- Anywhere a graphic would help your explanation, feel free to use one.
- The most common deduction on this assignment is: insufficient answer. This is a graduate level course.
- Each chapter should contain at least a page in length (e.g. 300 words).
- Each chapter will require independent research and testing to determine how you wish to format and present your information.
- Avoid restating what you find online. Try to internalize what you are demonstrating and explain it in your own words. If using a personal metaphor or analogy helps you feel like the explanation is more your own, please use it.
- You may schedule office hours with Dr. Dym to review a chapter and receive a "sample grade" that shows what you would have earned for the work.
- Not only should you have text explanations, you will need to have code that is cleanly commented and runs.
- Clearly label and structure your code files.

Required Chapters

Chapter 1 : Getting Started with Java

Discuss "Java." Make sure to cover the following: explain why it was the language chosen for this course, compare and contrast it with Python, and explain the compilation process. Make sure to include a discussion of data type handling in Java.

Chapter 2: Object Oriented Design with Java

Explain classes and class creation in Java. Make sure to include a discussion of constructors, getters/setters, basic class design, and anything else someone would need to know to understand classes. Feel free to insert example code as a reference image or figure.

Chapter 3: Getting Deeper with Class Objects

Chapter 2 was about basic class setup. Now discuss classes in more detail. At least answer the following questions: how do you make class objects “do” things? What are dynamic and static variables/functions? How do you approach class design? Example code is useful here and you will want to do some research and reading on your own.

Useful starting references for research:

1. https://docs.oracle.com/javase/tutorial/
2. Discussion of Object-Oriented Design from Grady Booch, co-developer of Unified Modeling Language (UML): https://dl.acm.org/doi/abs/10.1145/989791.989795
3. Object-Oriented Design research, a historical perspective, from Rebecca Wirfs-Brock and Ralph Johnson: https://dl.acm.org/doi/abs/10.1145/83880.84526
4. Recent (2019) Evaluation of OO Design: https://ieeexplore.ieee.org/abstract/document/8805686, and direct access to paper: https://pure.tudelft.nl/ws/files/69508918/paper.pdf

Chapter 4: Testing and Exception Handling

It’s not enough to make a class, it needs to be tested as well. Explain test driven development, JUnit testing in Java, and compare driver testing to unit testing. Then, explain the logic behind exception handling. What do developers and users stand to benefit through well-designed exceptions, and why should we implement them? How does testing exception handling differ from testing the rest of the code?

Chapter 5: Inheritance and Composition

Describe the theory behind why program designers implement inheritance and composition. Furthermore, explain when one is appropriate to implement over the other, and what potential advantages there are to each design method. Consider how you might answer the following questions in this chapter: how do you build more complex classes with inheritance, how does this chaining help with building a class, what is dynamic dispatching and what does it have to do with inheritance, what are abstract and concrete classes?

Chapter 6: Interfaces

Explain what you know so far about interfaces, both based on what you have been introduced to through modules and what you might have uncovered in your own research. Make sure to explain the difference between an abstract class and an interface. Explain the purpose of an interface. When would you use an interface compared to an abstract class? Why are there restrictions placed on an interface?

Chapter 7 : The Dangers of Inheritance

In module 11, you are introduced more thoroughly to inheritance and why it might not be the best-suited tool for every situation. Explain the dangers of inheritance when it gets out of control. Talk about how we can avoid those dangers. I suggest that the best way to explain this is through example.

Required Code

Create example code bundled in one package that uses the concepts outlined below. You may use code from this section to illustrate anything you discuss in your chapters for the textbook/how-to guide. Each class does not need to be related to each other, but combining them all as part of the same application helps create a cohesive project. Include files that demonstrate the following concepts:

1. Basic Java comments and basic Java syntax
2. Data types in Java
3. Type casting in Java
4. Basic Class Design and usage
5. Java methods
6. Java static methods/variables
7. Unit Testing and at least one case of exception handling
8. At least one example of Inheritance and one example of composition
9. Interfaces

What you need to submit:

1. Your code in a zip file, containing all the related class files, drivers, and test files needed to demonstrate the practical programming concepts outlined in “Required Code.”
2. A brief description labeled “Key Concepts” saved separately as a PDF that describes what parts of your code demonstrate each element required.
3. Create a code walkthrough video that explains how to operate each part of your code and why it works. The video should be 5-10 minutes in length. You can use tools like Screenshot or Quicktime if you are on Mac, or Open Broadcast Software (https://obsproject.com) on any computer. Just specify record, not broadcast when you launch your project in the application.

4. A PDF file of your textbook/how-to project. If you use screenshots of your code, include a caption with a reference to what file the code is located in. When you cite external sources, include a citation to that source in a **References** section at the end of the book in APA style (https://apastyle.apa.org/style-grammar-guidelines/citations/basic-principles). If you use clip art, icons, or graphics that are not your own, then please reference where you retrieved the imagery from in either a caption or in the references section.

Rubric
This project is worth 100 points, or 10% of your final grade.

50 points - The textbook/how-to guide:

- Full points: You have a clean and well-formatted document. Each chapter addresses the questions asked in the prompt in full. For all your research conducted, you have an appropriate APA-formatted citation. This implies that you have included at least a few references to external sources. You are not expected to know everything on your own. In fact, you need to demonstrate that you conducted research to articulate the information in this assignment.
    - Each chapter has at least 300 words devoted to it (that's at least 2100 words total dedicated to the body of the assignment).
- Partial points: You will begin to lose points if you do not appropriately answer the guiding prompts in this assignment sheet. If you are unsure of what a prompt is asking for, please contact Dr. Dym for clarification. You will receive partial credit if you do not include any references to external sources. You can lose points if you do not reference examples in your document. You are writing code files, so take advantage of them. Screenshot examples and paste them in as figures for reference. Include a diagram if it helps you illustrate your point. Draw and insert an explanatory cartoon if you want! In this section, the severity of points lost depends on how severe problems are across the entire document, not just in one chapter.
- What will not affect your grade: You will not lose points for
    - English language syntax and grammar mistakes–unless these are so severe that they completely alter the meaning of what you wrote.
    - Ugly visual aids, diagrams, and graphics. We can't all be artists.

50 points - The code and demo video:

- Full points: You have turned in a zip file containing code that demonstrates all the core concepts requested, including test files. It is cleanly formatted and contains comments throughout the code. You have the "Key Concepts" PDF included in your submission that points to which sections demonstrate what required programming concepts. You have also uploaded a 5-10 minute demo video that walks us through the code.
- Partial points: You will lose points for missing code that does not illustrate one of the requested programming concepts. The more components missing, the more points you lose. You will lose significant points for not turning in the video or "Key Concepts" PDF.

- What will not affect your grade:
    - Poor video quality
    - English language syntax and grammar errors in your comments and other written parts of the assignment, so long as the intended meaning is still clear.
    - Silly, strange, or boring examples for your programming files. They just need to work.

Good luck, and start working on this right away! You'll want to try to do 2 chapters a week.

Due July 3, 2023.