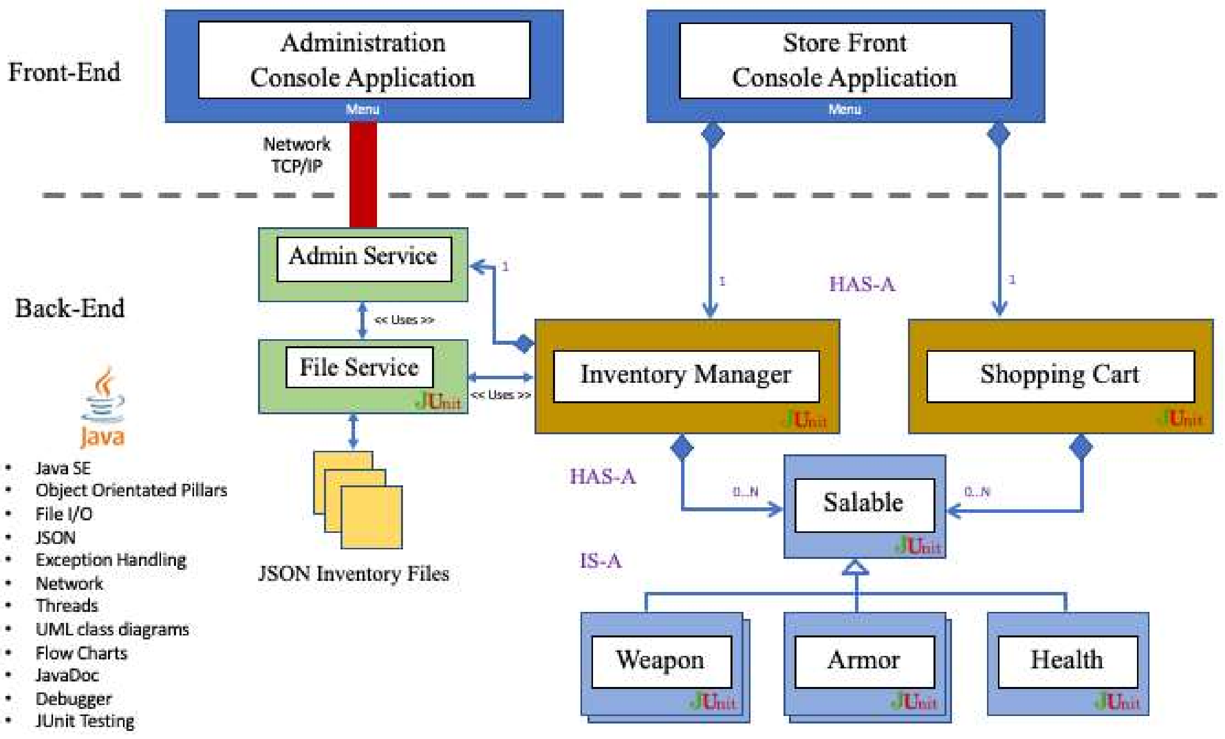
GRAND CANYON UNIVERSITY

CST-239 Milestone Project Guide

Introduction

In this course, you will work individually to design and build a Store Front for an Arena Fighting game using the Java programming language, and also leverage a Unit Testing framework to test your code. You will have some creative freedom to choose the products in your Store Front, but the following high-level requirements must be met for the assignments:

- The Java programming language version as specified in the Activity 1 Guide must be used in the course. The Eclipse Java development environment will be used in the course and will be installed and validated in the first Activity of this course.
- There will be 2 console-based applications and will not have a Graphical User Interface (GUI) The console applications will be: 1) Store Front for the Gamer User 2) Store Administration for the Admin User.
- The following illustrates a high-level diagram of the assignment:



Requirements

- The applications for the assignment need to meet the following functional requirements:
  - **Store Front Application:** Main application class used by the Game User who wants to shop for game weapons etc. The application will be console-based, providing a Welcome Message when the application is started, a menu of available of commands necessary for the Game User to view Salable Products, purchase a Salable Product, and return a Salable Product. The menu system will take user input from the keyboard and display appropriate feedback on the console.
  - **Inventory Manager:** A class used to manage the inventory of Salable Products available in the Store. The Store Front will interact with this class to allow a Game User to view Salable Products, purchase a Salable Product, and return a Salable Product. The Salable Products should be sortable by name and price. The Admin User will also be able to interact with this class, allowing them to get the current inventory of Salable Products and update the inventory of Salable Products.
  - **Shopping Cart:** A class used to manage the purchases of Salable Products by the Game User. The Store Front will interact with this class to allow a Game User to view the Shopping Cart contents, add a Salable Product to the Shopping Cart, remove a Salable Product from the Shopping Cart, and check out of the Store Front.
  - **Salable Products:** The base class used for Salable Products available in the product inventory. At minimum, a Salable Product should have a name, description, price, and quantity properties. There will be derived specializations of Salable Products that, at a minimum, will include: 2 different kinds of Weapons, 2 different kinds of Armor, and Health, which will all be available for purchase in the Store Front. Additional derived weapons, armor, or other items can be added as Salable Products as desired.
  - **File Service:** This class will encapsulate the access to the underlying JSON data files used to hold and maintain the state of the Salable Product inventory available in the Store.
  - **Administrative Application:** Main application class used by the Admin User who wishes to view and/or replenish the inventory of the Salable Products available in the Store. This application will be separate from the Store Front application and will also be console-based, with an appropriate menu system to support the features of the Admin User. This application will communicate to the Store Front application over the local network and by exchanging JSON-based communication messages.

- **Server Thread:** This class will implement a thread so the network server will run in the background of the Store Front Application and will process the commands sent over the network from the Administrative Application.
- **Admin Service:** This class will communicate with the Administrative Application and wait for command requests from the Admin User to send commands to the Store Front Application.

- The applications for the assignment need to include the following components and/or classes that meet the following technical requirements:
  - **Store Front Application:** Java class with a *main()* method, containing an instance of the Inventory Manager class and the Shopping Cart class.
  - **Inventory Manager:** Java class that contains an array or list of Salable Products. This class needs to be designed such that it can be unit tested.
  - **Shopping Cart:** Java class that contains an array or list of Salable Products. This class needs to be designed such that it can be unit tested.
  - **Salable Products:** Java classes that leverage encapsulation and inheritance in the design of base class(es) and derived specialization classes. Common products should be comparable and sortable. These classes need to be designed such that they can be unit tested.
  - **File Service:** Java class will be designed as an abstract or interface class with an implementation class that uses appropriate Java File I/O classes to read and write files that store data in the JSON format. The files will represent and maintain the state of the Salable Product inventory available in the Store. Proper exception handling must be supported with proper error messages being displayed to the Game User or Admin User using this service. This class needs to be designed such that it can be unit tested.
  - **Administrative Application:** Java class with a *main()* method that uses the appropriate Java Networking classes to support sending commands to the Store Front Application to view and/or replenish the inventory of the Store. Proper exception handling must be supported with proper error messages being displayed to the Admin User using this service. All data exchanged will be in JSON format. The following commands will be supported:
    - Command: U, will update the Store Front inventory with new Salable Products. The data payload will be a JSON string of Salable Products that is terminated by a newline character (‘\n’).
    - Command: R, will return all of the Salable Products from the Store Front inventory in JSON format that is terminated by a newline character (‘\n’).
    - A command and its data payload should be separated by the pipe (‘|’) separator character.

- **Admin Service:** Java class that uses the appropriate Java networking classes for communication between the Administrative Application and the Store Front Application. See the Administrative Application technical requirements for the data formats and command that will be supported.

GRAND CANYON UNIVERSITY

Project Milestones

The applications will be designed and built using an iterative approach and delivered using the following milestone assignments.

Milestone 1 – Initial Design and Implementation:

- ✓ Analyze and model out, using UML class diagrams, a Salable Product that would be available in a Store Front, Inventory Manager, and Shopping Cart that supports the functionality requirements.
  - ✓ At minimum, a Salable Product should have a Name, Description, Price, and Quantity.
  - ✓ At minimum, the Store Front should support initializing the state of the Store, purchasing a Salable Product, and canceling the purchase of a Salable Product.
- ✓ Draw a flow chart of the logic of a User interacting with the Store Front, along with the internal logic of the possible interactions with the Inventory Manager and the Shopping Cart.
- ✓ Implement the code for all UML class designs.
- ✓ Implement the code for Store Front Application to exercise all functions.
- ✓ Document all code using JavaDoc documentation standards and generate the JavaDoc files.
- ✓ Create a screencast video that includes a functional demonstration of the application and a code walk-through explaining how the design and code work. The screencast video should be 8–10 minutes long.
- ✓ Submit all code to an approved GCU GIT private repository.
- ✓ Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

Milestone 2 – Salable Products and Store Front Application:

- ✓ Analyze and model out, using UML class diagrams, Salable Product, Weapon, Armor, and Health classes that support the following properties:
  - ✓ At minimum, a Salable Product should have a Name, Description, Price, and Quantity.
- ✓ Analyze and model out, using UML class diagrams, a Store Front Application as a console-based application and the Inventory Manager that supports the following features:
  - ✓ Populate the Store Front with an initial inventory of Salable Products that includes 2 different kind of Weapons, 2 different kinds of Armor, and Health (the inventory can hard coded in the application).

    - Update the methods used to support the purchasing and canceling of a Salable Product as necessary.
- Draw a flow chart of the logic of a Game User interacting with the Store Front.
- Implement the code for the Store Front Application to exercise all code and support the following features:
    - ? Displays the name of the Store Front and a Welcome Message to the console.
    - Displays a list of actions for the User to interact with the Store Front.
    - Executes a desired Store Front action based on keyboard entry from the User.
    - Displays appropriate detailed feedback and error messages for the User to the console.
- Implement the code for all UML class designs.
- Document all code using JavaDoc documentation standards and generate the JavaDoc files.
- Create a screencast video that includes a functional demonstration of the application and a code walk-through explaining how the design and code work. The screencast video should be 8–10 minutes long.
- Submit all code to an approved GCU GIT private repository.
- Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

Milestone 3 – Inventory Manager and Shopping Cart:

- Analyze and model out, using UML class diagrams, an Inventory Manager that supports the following features:
    - Initialization of the Store inventory (should be invoked when the Store Front starts up).
    - Removing a Salable Product from Store inventory (should be invoked when a Salable Product is purchased).
    - Adding a Salable Product to Store inventory (should be invoked when a Salable Product purchase is cancelled).
    - Return the entire inventory.
    - Integration of this class with the Store Front application.
- Analyze and model out, using UML class diagrams, a Shopping Cart that supports the following features:
    - Initialization of the Shopping Cart (should be invoked when the Store Front starts up).
    - Adding a Salable Product to the Shopping Cart (should be invoked when a Salable Product is purchased).
    - Removing a Salable Product from the Shopping Cart (should be invoked when a Salable Product purchase is canceled).

    - Return the contents of the Shopping Cart.
    - Empty the contents of the Shopping Cart.
    - Integration of this class with the Store Front application.
- Update the Salable Product UML class diagrams to support the following new features:
    - Update all Weapon classes so that they implement the comparable interface. Comparison should be based on the name of the item and follow alphabetical ordering rules that ignore case.
- Update the Store Front UML class diagrams to support the following new features:
    - Integration of the Inventory Manager.
    - Integration of the Shopping Cart.
- Update the flow chart of the logic of a Game User interacting with the Store Front and with the internal Inventory Manager and Shopping Cart.
- Implement the code for all UML class designs.
- Document all code using JavaDoc documentation standards and generate the JavaDoc files.
- Create a screencast video that includes a functional demonstration of the application and a code walk-through explaining how the design and code work. The screencast video should be 8–10 minutes long.
- Submit all code to an approved GCU GIT private repository.
- Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

NOTE: This assignment introduces the ABET Student Outcome 2: An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.

Milestone 4 – File I/O Integration:

- Update the Inventory Manager UML class diagram to support the following new features:
    - Initialization of Store inventory from an external JSON file using the File Service.
    - Proper exception handling for all File I/O and JSON serialization.
- Update the flow chart of the logic of a Game User interacting with the Store Front and with the internal Inventory Manager and Shopping Cart.
- Implement the code for all UML class designs.
- Document all code using JavaDoc documentation standards and generate the JavaDoc files.

- Create a screencast video that includes a functional demonstration of the application and a code walk-through explaining how the design and code work. The screencast video should be 8–10 minutes long.
- Submit all code to an approved GCU GIT private repository.
- Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

Milestone 5 – Generics and Collection Framework Integration:

- Update all UML class diagrams to support the following new features:
  - Convert all arrays to use generics as well as the use of classes and algorithms from the Collections Framework.
  - Being able to sort Salable Products in ascending or descending order based on name and price.
- Update the flow chart of the logic of a Game User interacting with the Store Front and with the internal Inventory Manager and Shopping Cart.
- Implement the code for all UML class designs.
- Document all code using JavaDoc documentation standards and generate the JavaDoc files.
- Create a screencast video that includes a functional demonstration of the application and a code walk-through explaining how the design and code work. The screencast video should be 8–10 minutes long.
- Submit all code to an approved GCU GIT private repository.
- Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

Milestone 6 – Administration Application:

- Analyze and model out, using UML class diagrams, an Administration Application as a console-based application and an Administration Service that supports the following features:
  - Ability to send Administration commands over the local network from an Admin User to the Store Front Application.
  - The following commands sent from the Administration Application by the Administration Service will be accepted and processed in the Store Front Application:
    - Command: U, will update the Store Front inventory with new Salable Products. The data payload will be a JSON string of Salable Products.

    The Store Front Applications inventory should be updated accordingly and be available to the Game User.
    - Command: R, will return all the Salable Products from the Store Front Inventory in JSON format to the Administrative Application. The Administrative Application should display the inventory to the console.
  - The Administrative Application will send commands over the network (i.e., network client) to the Store Front Application, which will receive the commands over the network (i.e., the network server). The Store Front Application should run the network server in the background, such that it does not impact the Game User while interacting with the Store Front Application, and that should leverage the File Server to read and write JSON inventory data files.
- Update the flow chart of the logic of a Game User interacting with the Store Front and with the internal Inventory Manager and Shopping Cart along with the new Administration Service that will receive and process commands. Also include the logic of an Admin User interacting with the Store Front Application.
- Implement the code for all UML class designs.
- Document all code using JavaDoc documentation standards and generate the JavaDoc files.
- Create a screencast video that includes a functional demonstration of the application and a code walk-through explaining how the design and code work. The screencast video should be 8–10 minutes long.
- Submit all code to an approved GCU GIT private repository.
- Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

Milestone 7 – Unit Testing and Final Project:

- Write JUnit unit tests for the following classes: all Salable Product classes (both base class and any derived classes) and Inventory Manager class.
- Refactor and clean up any code necessary for the final submission of the project.
- Update the final Flow Chart as necessary for the final submission of the project.
- Update the final UML diagram as necessary for the final submission of the project.
- Update the JavaDoc as necessary for the final submission of the project.
- Create a final screencast video that includes a functional demonstration of the completed application and demonstration of the JUnit tests being run. The screencast video should be 5-6 minutes long.
- Submit all code to an approved GCU GIT private repository.

- Zip up all diagrams, code, generated JavaDoc documentation, and a README text file with a link to a screencast video to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.

Milestone 7 – Final Project Presentation:

- Develop a PowerPoint presentation that outlines the following:
    - Functional Introduction to the application developed.
    - Goals and Design Decisions for the application developed.
    - Challenges encountered during the development of the application.
    - Pending bugs or issues remaining in the application.
    - Short (2-3-minute) screencast video demonstration of the working application.
    - Five things learned during the course that could be leveraged in future application projects.
    - Questions and Answers (in-class presentation only).
- For online, this will be a virtual presentation delivered via a screencast video.
- For ground or in-class, it is up to the instructor whether the presentation will be delivered virtually via a screencast video or delivered as a face-to-face presentation in the classroom.
- Zip up the PowerPoint presentation and also include a README text file with a link to a screencast video demonstration if the presentation was given virtually to a zip file named *cst-239-assignment.zip* and submit the zip file to the LMS.