The Four Musketeers

Team 6

Team Members: Anaiah Quinn, Michael Ike, Erik LaNeave, Ian Gutierrez

Part A – Project

How did we use Object-Oriented Programming?

- How We Utilized Object-Oriented Programming In Our Program
  - We built classes that satisfied one specific requirement of the program. Combined, all of the classes produce the final program.
  - **Abstraction** - Abstraction allowed us to have base classes that other sub-classes implemented and extended from. In our code we made the Event and Venue classes abstract.
  - **Encapsulation** - By encapsulating data such as the private attributes, setters, and getters, within classes, we protected our data from being directly accessed and manipulated.
  - **Inheritance** - By utilizing inheritance, we achieved a modular and reusable codebase. In our code Event was inherited by Sport, Concert, Festival and Venue was inherited by Open Air, Arena, Auditorium, Stadium.
  - **Polymorphism -** We used polymorphism in order to reuse code and save time, specifically on our methods that parse CSV files.
  - Object-oriented programming resulted in low couplings and high cohesion.
- Our Strategy For Merging Code From All Teammates Into One
  - Conduct code reviews at the beginning.
  - Identify what was done differently and similarly.
  - Focus on code that is efficient, clean, and follows the object oriented approach
  - Focused on preserving the object structure and logic from one team while implementing UI functionality from another.
  - Merge whole parts instead of intertwining every single class.
  - The difficulty primarily stemmed from aligning the different coding styles, ensuring compatibility, and merging the distinct components maintain a cohesive and functional user interface.

Design Pattern Overview: How did we use Design Patterns?

- Factory Design Pattern: Event, Venue
- We used it to create the correct types of events and venues based of the information in the csv file.
- We implemented an Event factory class that takes String input and creates the correct type of event. We did the same for Venue factory.
- Singleton Design Pattern: Log, Ticket Miner
- We used it to make it easier to access and update the log file from any class. Same with information stored in the ticket miner class.
- We used the log object in the user interface classes, updating the users actions as the program ran. We used the ticket miner object in the user interface classes as well as the purchases classes to calculate and keep track of the various fees.

Data Structures Overview: How did we apply data structures?

- Our approach when selecting data structures came down to speed, given data structure, and ease of integration into pre-existing code. With this in mind, we decided to rely heavily on HashMaps/LinkedHashMaps for storing all event and customer information, this is for two reasons the given data had a unique key in the form of an ID and the constant access time is utilized extensively to complete all update operations.
- ArrayList were used in a couple of situations when frequent appending of information was needed, such as invoices that were created on a successful purchase. An ArrayList was also used to store all auto purchase information, because of the structure of the data and the order of the instructions needed to be kept.
- The final data structure used was the built-in Java StringBuilder, the reason for this is because it's a mutable succession of characters, unlike the immutable String class. This greatly increased the speed of log operations and the auto purchase as a whole.

Part B – Course Reflection

Major Takeaways of the Course

- **Ian Gutierrez**
  - Major takeaway - Low coupling and high cohesion code makes life a lot easier.
  - Course importance - Requires you to write code that not only works but is well designed.
  - I believe object-oriented programming is important because it results in code that is higher quality for everyone.

- **Anaiah Quinn**
  - Major takeaway - The 4 pillars of AOOP: Abstraction, Polymorphism, Encapsulation and Inheritance.
  - Course Importance- Teaches you how to code a project to completion.
  - I believe object-oriented programming is important because it teaches you to code more dynamic and professional .

- **Erik LaNeave**
  - Major takeaway - The importance of writing modular code that has both high cohesion and low coupling.
  - Course Importance - Eye opening to what makes object-oriented programming powerful when designing and writing programs.
  - I believe object-oriented programming is important because it makes modeling aspects of the world simple and can make the code easier to update and fix.

- **Michael Ike**
  - Major takeaway - Separate classes for individual tasks in order to it modular, reusable, and comprehensible.
  - Course Importance - Teaches you how useful the four pillars of OOP can be.
  - I believe object-oriented programming is important because it simplifies software development by making code maintenance much easier.

What did we learn in the course?

- **Design Patterns**
  - After learning about design patterns they gave simple and easy solutions to problems we were unsure how to address in a concisely. They also helped clean up the overall implementation of the code.
- **Usefulness of Diagrams**
  - The several different diagrams that were talked about during the course are useful for planning and finding missing requirements in the implementation.
- **The 4 Pillars**
  - Keeping The 4 Pillars in mind while programming made the overall implementation take full advantage of the power of Object Oriented Programming.
- **Low Coupling & High Cohesion**
  - Learning about these two principles greatly influenced and changed how we wrote our code in the programming assignments. This decreased the amount of time needed to refactor and fix bugs that appeared.
- **Testing**
  - The lecture on testing put a greater emphasis on the importance and process of testing which led to the creation of more reliable and robust code in the programming assignments.

Advice for Future Students

- Listen to feedback. Use it to learn and improve.
- Group work is not as bad as it seems in the beginning.
- Always look for ways to improve your code.

Part C – Demo