# CarGoRod

Team 11

Team Members:

Ruben Carmona

Brian G. Rodiles Delgado

Jose Luis Espinoza

# Miner Airlines Project

# How did we use Object-Oriented Programming?

## •How did you utilize Object-Oriented Programming in your project?

While programming our project, we made sure that the relationships between classes and objects were concrete and understandable.

For this project we used, HashMap, and ArrayList with the purpose to store elements or objects, that will be utilized during the execution of the program and with this be able to look for specific information from an object.

In the case of the project's running time, it varies depending on the data structure used. The following table showcases some of those running time complexities:

| Operation | Avg. Time Complexity |
|---|---|
| Adding a person (customer or employee) or flight to a map. | O(1) |
| Searching for a person, flight on a map. | O(1) |

| Operation | Avg. Time Complexity |
|---|---|
| Adding a ticket for a person to an ArrayList. | O(1) or O(n) when resizing (infrequent) |
| Accessing a ticket by index in an ArrayList. | O(1) |

Abstraction was used for our Flight and Person classes and classes would inherit from them. Domestic and international inherited from flight. Employee and customer inherited from person. This allowed us to create objects from the child class that would be inherited from the parent class.

# How did we use Object-Oriented Programming?

## •Describe your process for implementing the project:

Overall, the journey of implementing this project was a good learning path that allowed us to have some realization on how we would work and put our skills to work for the industry. Meaning that starting a program from scratch and at the end merging the code with the team was good practice for us to get ready for the real world.

We made the code our own in various ways. For example, we approached the same solutions with a different problem-solving strategy, which led to different data structures and running times during the individual process. However, we also implemented extra functionalities for user interaction enhancement or file output management (see Figure 1 and 2).

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Arrival Tim | First Class | ID | Origin Cod | Surcharge | Destinatio | Destinatio | Duration | Origin Airp | Departing |
| 2 | 7:59 AM | 12 | 1 | ELP | 0 | Dallas/Ft. ' | DFW | 94 | Texas | ######## |
| 3 | 10:54 AM | 12 | 2 | ELP | 0 | Dallas/Ft. ' | DFW | 94 | Texas | ######## |
| 4 | 2:39 PM | 10 | 3 | ELP | 0 | Dallas/Ft. ' | DFW | 94 | Texas | ######## |
| 5 | 5:34 PM | 14 | 4 | ELP | 0 | Dallas/Ft. ' | DFW | 94 | Texas | ######## |
| 6 | 5:59 AM | 14 | 5 | DFW | 0 | El Paso Int | ELP | 94 | Texas | ######## |
| 7 | 8:54 AM | 14 | 6 | DFW | 0 | El Paso Int | ELP | 94 | Texas | ######## |
| 8 | 12:39 PM | 13 | 7 | DFW | 0 | El Paso Int | ELP | 94 | Texas | ######## |
| 9 | 3:34 PM | 10 | 8 | DFW | 0 | El Paso Int | ELP | 94 | Texas | ######## |
| 10 | 7:19 PM | 13 | 9 | DFW | 0 | El Paso Int | ELP | 94 | Texas | ######## |

Figure 1. View of FlightSchedulePA6.csv.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ID | Flight Num | Origin Airp | Origin Cod | Origin Airp | Origin Airp | Origin Airp | Origin Airp | Origin Airp | Destinatio |
| 2 | 1 | 2311 | El Paso Int | ELP | El Paso | Texas | United Sta | 3.5 | FALSE | Dallas/Ft. ' |
| 3 | 2 | 2322 | El Paso Int | ELP | El Paso | Texas | United Sta | 3.5 | FALSE | Dallas/Ft. ' |
| 4 | 3 | 2333 | El Paso Int | ELP | El Paso | Texas | United Sta | 3.5 | FALSE | Dallas/Ft. ' |
| 5 | 4 | 2344 | El Paso Int | ELP | El Paso | Texas | United Sta | 3.5 | FALSE | Dallas/Ft. ' |
| 6 | 5 | 2355 | Dallas/Ft. ' | DFW | Dallas | Texas | United Sta | 3 | TRUE | El Paso Int |
| 7 | 6 | 2366 | Dallas/Ft. ' | DFW | Dallas | Texas | United Sta | 3 | TRUE | El Paso Int |
| 8 | 7 | 2377 | Dallas/Ft. ' | DFW | Dallas | Texas | United Sta | 3 | TRUE | El Paso Int |
| 9 | 8 | 2388 | Dallas/Ft. ' | DFW | Dallas | Texas | United Sta | 3 | TRUE | El Paso Int |
| 10 | 9 | 2309 | Dallas/Ft. ' | DFW | Dallas | Texas | United Sta | 3 | TRUE | El Paso Int |

Figure 2. View of UpdatedFlightSchedule.csv.

## •Describe your strategy for merging code from all teammates into one:

It was somehow difficult to merge all three codes because even though we had to follow the same instructions and implementations were similar, relations between classes and methods being called during the run time were different from each other. Once we figured out how our codes would merge everything went smoothly from there.

## •Describe your testing strategies:

Our testing strategies were no different from others, we tested our code a lot with different scenarios making sure that outputs were what we were expecting and that calculations and/or any algorithms that we implemented were working. We let friends and family members interact with the code and learn from there what our code was missing or where it was crashing.

# How did we use Design Patterns?

## •Design patterns we use:

For this project we used Factory and Singleton design patterns

## •Why we used them?

For factory design pattern we used it with the purpose to allow our code to be for flexible and maintainable.

For singleton design pattern we used it with the purpose to restrict the number of instances created on a class, this will allow us to limit resource consumption.

## •Describe how they were used:

We used factory for person and flight, this design will return a specific object. For example, for flight would return objects of either international or domestic, and the same for person, it would return objects of either customer or employee.

# UML Class Diagram

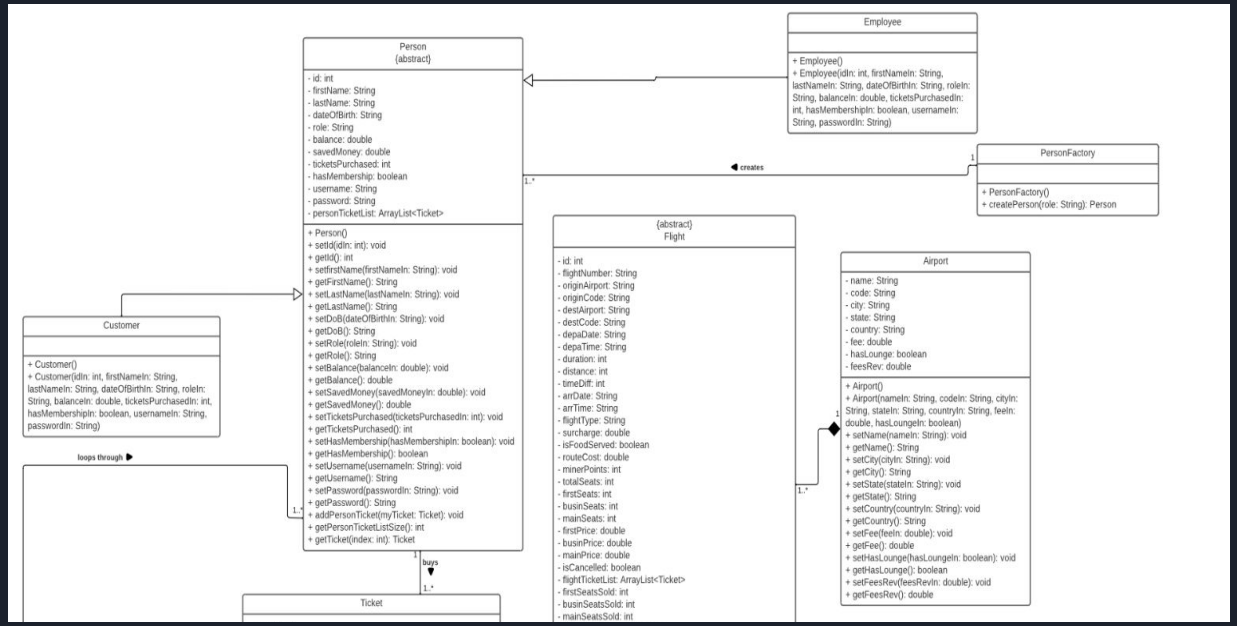


Figure 3. UML showcasing PersonFactory class relation to the rest of classes.

# UML Class Diagram

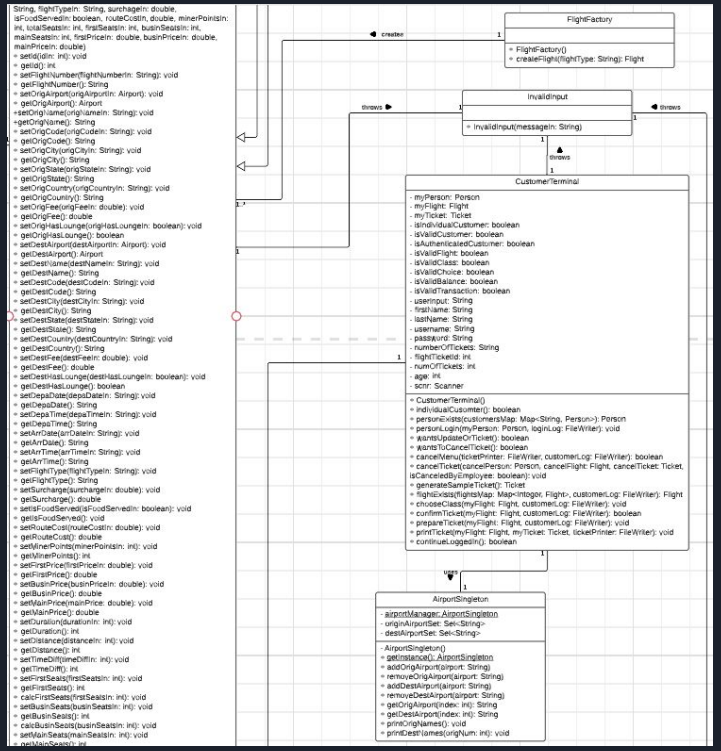


Figure 4. UML showcasing AirportSingleton, FlightFactory, and InvalidInput classes relation to the rest of the classes.

# UML Use Case Diagram

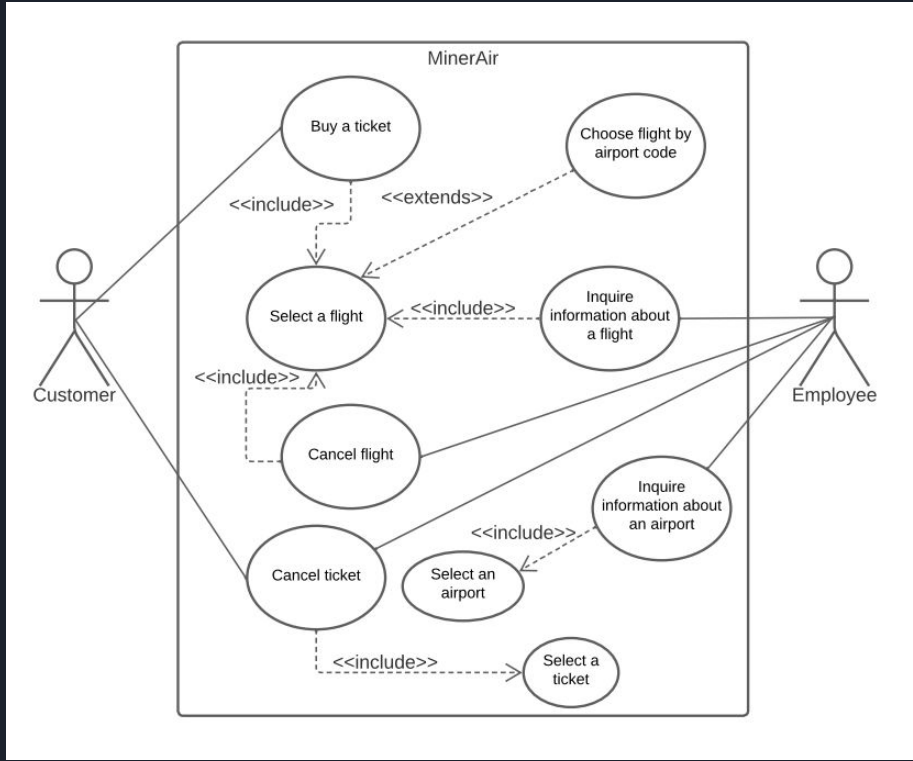


Figure 5. UML use case diagram with 5 use cases, 2 actors, 1 extend, and 5 includes.

# Course Reflection

# Major Takeaways of the Course

- **Ruben Carmona**
  - A major takeaway of this course is that it helped growth with my coding skills and it prepared me for the future to become a professional computer scientists.
- **Brian G. Rodiles Delgado**
  - It allowed me to not only code better but to also take into consideration all the work that is behind an object-oriented program such as state diagrams, use diagrams, and many more before-code requirements. It also allowed me to experience how to think proactively about software maintenance.
- **Jose Luis Espinoza**
  - This course helped me grow as a student and as a computer scientist. It really helped me a lot to be prepare for future projects inside and outside of the academic life.

# Why is this course and OO programming important?

- This course is important because it give students the knowledge of modular and structures approach when programming. Encapsulation, inheritance, abstraction, polymorphism are only a couple of examples that we learn during this course and had helped us when designing/coding a program.

- We agreed that some of the most important things we've learned in this course would be design patterns and UML diagrams, because these two allow developers have a good approach with the design and perspective of the program or problem that would need to be programmed.

- These two mentioned above would be a good help in the future because we would have the knowledge to create a diagram of the code and this will help us during the designing of the code.

# What did we learned?

- In this course we did covered a lot of helpful topics that will prepared us for what the real world is expecting from us.

- We learned that developers do not only code when working on a program or solution, we now know all the design and work that needs to be done or can be done before and during the development of a program. med.

- How to work with design patterns for our code to be able to improve quality, reusability, facilitate communication, and to follow best practices and design principles. As a result have a higher quality of code.

# How did we grow as a student?

- **Ruben Carmona**
  - This course helped grow as a student on how to get prepare for the real world and realized some expectations when working for the industry and it pushed me to practice my coding skills.
- **Brian G. Rodiles Delgado**
  - This class helped me to be more prepared for the industry with a sense of collaboration and consideration of computing practices.
- **Jose Luis Espinoza**
  - This course provided valuable insights of what to expect from now on when coding a program and it give more of an idea of all the work that needs to be done behind a program.

# What would you tell future students of the class

- **To start early!!!**
  - Mainly to practice OOP concepts and techniques and to make sure they have a good understanding of OOP fundamentals.
  - Organization and time management are also key for this course. Assign time outside of school for assignments and studying for quizzes and exams.
  - Always ask questions.
- **What would we tell them to look forward?**
  - Workflow.
  - Relevant information.
  - That they will improve their coding skills.

# THANKS