# Game Overflow

**Group Members**

Sami Cemek, Zach East, Sebastian Mark

**Title**
Game Overflow
Github: https://github.com/ascemek/GameOverflow

**Description**

- A text-based game system where you can choose between 2-3 games
    - All games are turn-based in some form
    - Pokemon Duel (Sami)
    - Punch-Out like game (Zach)
    - Hangman (Sebastian)
- Each game tracks the user's score
    - Keeps a leaderboard of high scores and displays them based on each game
- Each score/user is passed to a singleton database stores the data as hashmaps
    - This information can be passed to other classes like the class that constructs the leaderboard txt files
- The menu that is the observer (observer pattern) of the database informs the LeaderBoardDisplay class to update the txt files when the database is updated.

- Incorporating design patterns learned throughout the semester
- Integrating the proper design pattern to modularise code and integrate features

- Challenges: Integrating all three games to fit each pattern was challenging. For example, two out of the three games run on a similar game engine i.e., a fighting simulator. However, the hangman game has both a one and two-player feature but does not have an NPC fighting feature. Therefore, some patterns (template pattern) did not apply to the hangman game. Additionally, having all three systems running from the same menu and loading into one database was challenging. This could have easily caused tight coupling and dependency issues. Meanwhile, being conscious of not violating any S.O.L.I.D principles throughout the implementation.

**Resources**

- Previous Labs and Slides
- TAs
- https://refactoring.guru/

- Websites linked as resources in the slides
- https://www.geeksforgeeks.org/sorting-a-hashmap-according-to-values/

**Timeline**

We spent (approximately 4 days/week * 3 hours/day * 2 weeks = 24 hours per teammate) on the work. With additional hours of debugging and implementing additional features to each game for approximately 10 hours total.

**Sample Input-Output**

Check out the presentation recording to see the sample input, output, and how to run the code.

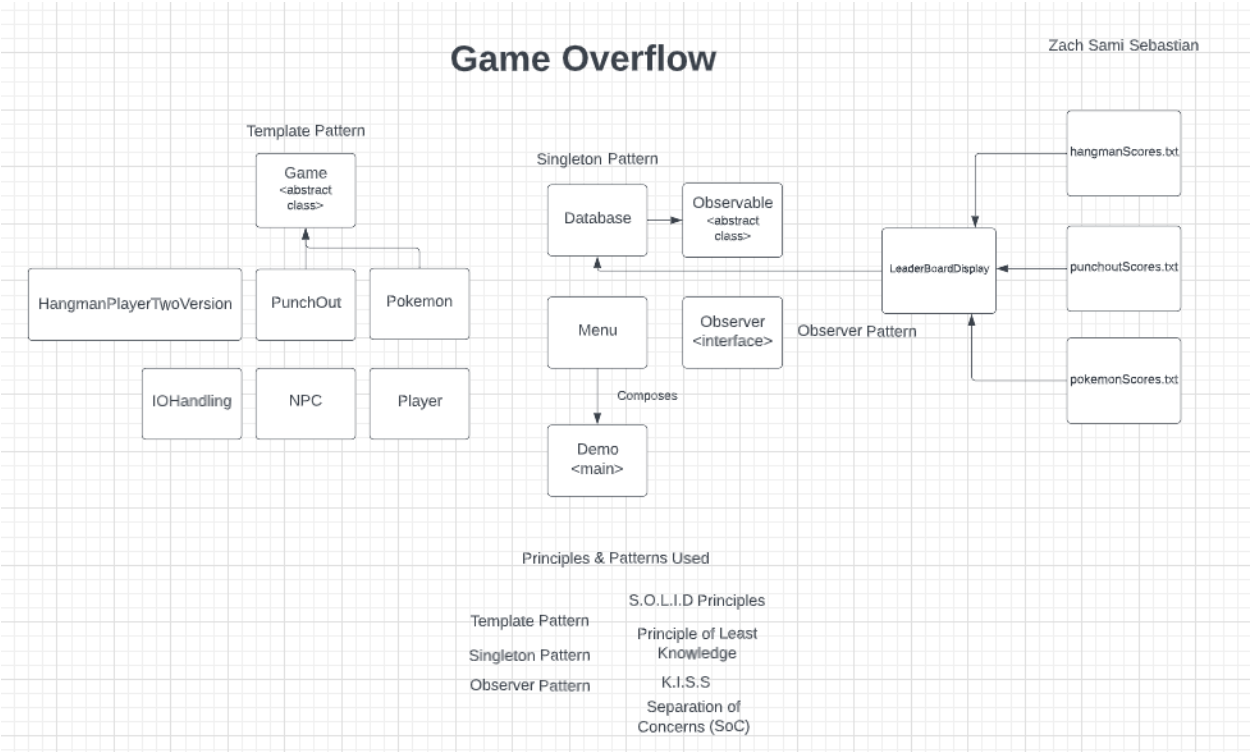
**Patterns**

Use S.O.L.I.D
Template Pattern = Gameplay
Singleton Pattern = We would want to have only one instance of the database where we keep track of the scores.
Observer Pattern = The player will be the observer, and the high score will be observable. Every time the high score changes the player will be updated on the terminal.

**Conclusion**

We were able to implement and practice several of the patterns and principles we learned during classes this semester. While we did run into a few issues, we tried to stick to S.O.L.I.D where we could and managed to create a working system with multiple games and a leaderboard. It is modular enough so that if someone wanted to implement a new game or switch out existing ones, there is a framework both to create new games (template method). They also would be able to add their game to the LeaderBoardDisplay and Database classes with very little modification. Finally, we applied what we learned to something fun and interesting that we are all passionate about, which is gaming and coding!

**UML**


Game Overflow
Zach Sami Sebastian
Template Pattern
Game
<abstract class>
HangmanPlayerTwoVersion
PunchOut
Pokemon
IOHandling
NPC
Player
Singleton Pattern
Database
Observable
<abstract class>
Menu
Observer
<interface>
Composes
Demo
<main>
Observer Pattern
LeaderBoardDisplay
hangmanScores.txt
punchoutScores.txt
pokemonScores.txt
Principles & Patterns Used
Template Pattern
Singleton Pattern
Observer Pattern
S.O.L.I.D Principles
Principle of Least Knowledge
K.I.S.S
Separation of Concerns (SoC)