# DASS Assignment 3
# Design Document

## Project Information

**Name of Project:** Bowling Centre Management System
**Date of Submission:** 9th April 2020
**Members:**

- Kalp Shah - 2018113003
- Shreeya Pahune - 2018113011
- Vivek Pamnani - 2018

**Number of hours contributed to the project:**

- Kalp Shah - 25 Hours
- Shreeya Pahune - 25 hours
- Vivek Pamnani - 25 hours

**Role played in the project:**

- Kalp Shah - Refactoring (10 files), Feature Implementation
- Shreeya Pahune - Refactoring (10 files), UML Diagram, Report
- Vivek Pamnani - Refactoring (10 files), UML Diagram, Report

## Short Overview

### Product

The Lucky Strikes Bowling Center (LSBC) is a Bowling Game implemented in JAVA, which simulates a real-world Bowling Alley with multiple Bowling lanes, each lane has multiple players taking turns to knock over the bowling pins.

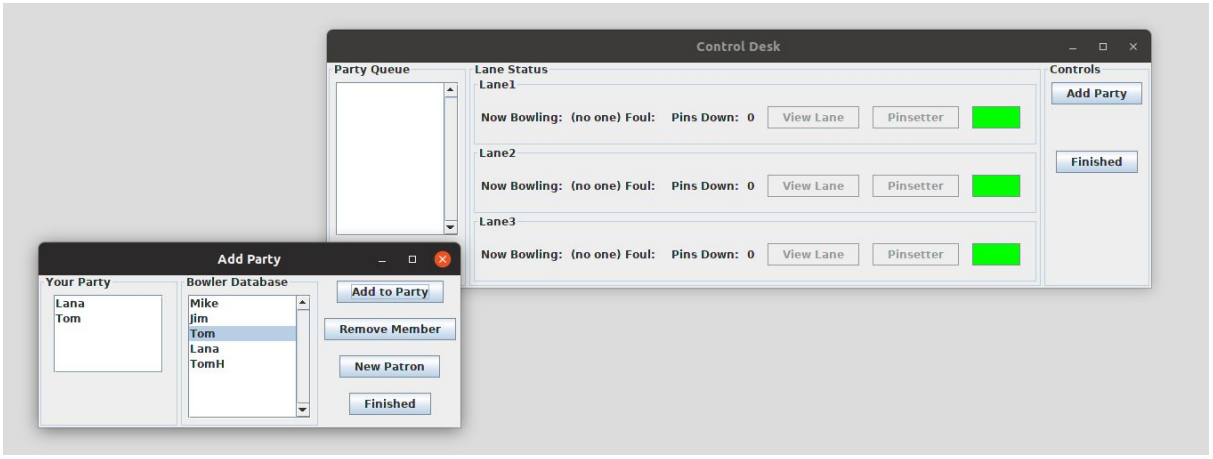
### Features

#### Control Desk

- Represents an Alley
- View games in progress in a particular Lane

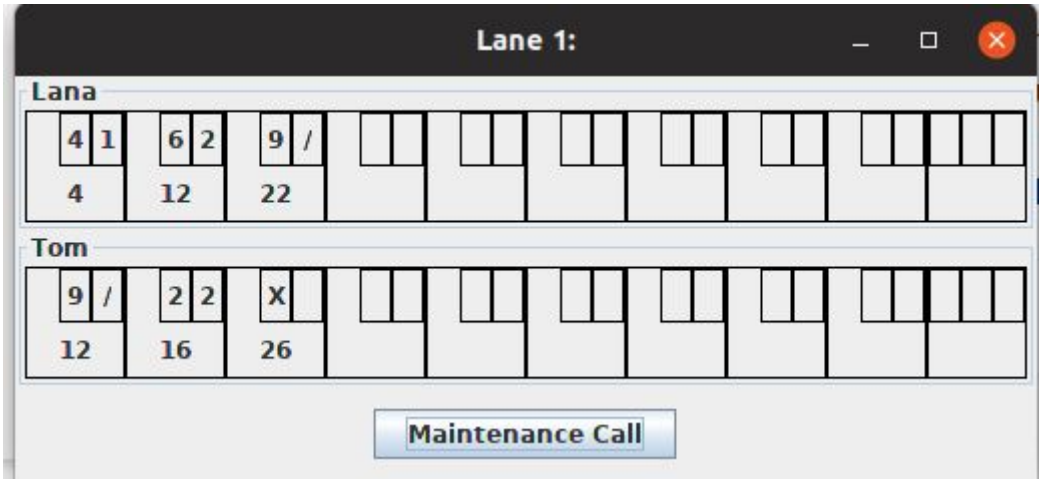

## Add Party

- Add Members to Party
- Remove members from a Party




## Scoring Station

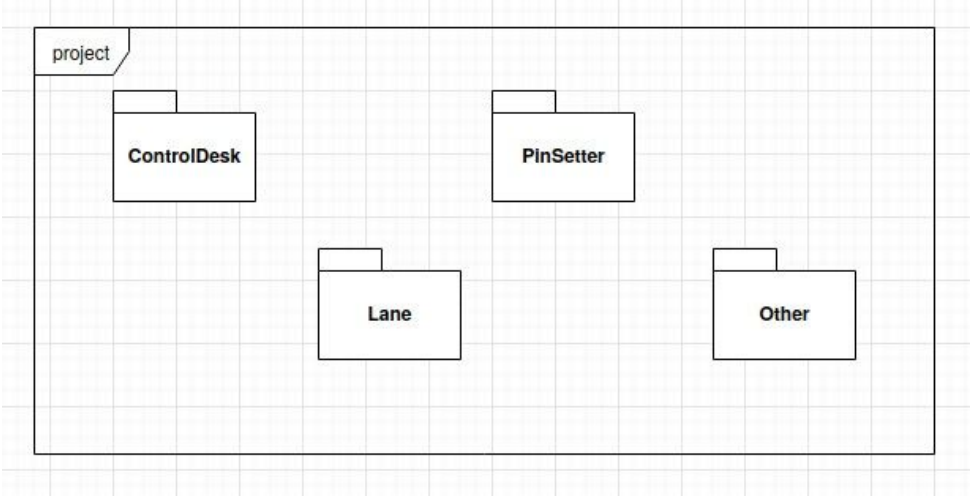
- A graphic representation of their scores
- Each lane is equipped with a stand-alone scoring station that lists the bowlers' names



Pinsetter Interface

- Communicates to the scoring station the pins that are left standing after a bowler has completed a throw
- Pins re-racked after 2 consecutive throws


# UML Class Diagrams




## Original code:

https://gitlab.com/Blizzard57/bowling-game/-/tree/dev-shreeya/Diagrams/UML_Class_Dgm/Original_Code

## Refactored code:

https://gitlab.com/Blizzard57/bowling-game/-/tree/dev-shreeya/Diagrams/UML_Class_Dgm/Refactored_Code

# UML Sequence Diagram

## Original Code:

https://gitlab.com/Blizzard57/bowling-game/-/blob/dev-shreeya/Diagrams/UML_Seq_Dgm/

# Refactored Code:

https://gitlab.com/Blizzard57/bowling-game/-/blob/dev-shreeya/Diagrams/UML_Seq_Dgm/

# Brief - Original Code

Reading through the codebase, we saw each section's role.
We identified roles of each class, the design patterns used and the fidelities to the given design doc.

Classes like Lane, ControlDeskView, etc. were long and complex. These classes could be internally easily broken down into multiple functions.
The cyclomatic complexity was pretty high across many files because of deep nested if-else statements.
Classes like Lane, ControlDeskView, etc. were long and complex. These classes could be internally easily broken down into multiple functions.
There were few catch blocks with empty implementation.
There were few places where implementation was repeated they could be enclosed in the separate methods.
The code was outdated as some deprecated methods were being used.

Coupling and cohesion was handled fairly
Variables were well defined, and were private
The code has good comments which help in understanding the code.
The code was well divided into Classes in a meaningful way.
The Classes have only important class-variables.

The Observer Pattern - subscription based setup where various event based statuses are broadcast to all objects subscribing for the same.
The Adapter Pattern - PrintableText Class module, during interactions with the file database.

# Summarization of each Major Class

| Class Name | Responsibility |
| --- | --- |
| AddpartyView | This class is responsible for:<br>It opens a popup window, allowing us to add or remove a patron for a party,<br>It allows us to create a patron which will be in the party and<br>It allows us to start the game by pressing the finished button. |
| Alley | This class is responsible for:<br>Simulate the alley.<br>Store related information about the Control Desk, Number of lanes<br>Provide methods to access them. |
| BowlerFile | It allows us to interact with the list of Bowlers by getting or updating information about the current bowlers.<br>Store information of the nickname chosen, full name and email<br>Adding or removing bowlers. |
| ControlDesk | It carries out the functionalities in the back<br>Offered by: ControlDeskView<br>and AddPartyView offer<br><br>Contains methods of managing the game like assigning the empty lane to the party, create a waiting queue etc |
| ControlDeskView | Creates a view of the control desk.<br>Provides an interface for us to add a Party or to finish the game.<br>View the waiting parties and the current lanes assigned |
| Drive | This is the main class file it contains the definition of the number of lanes and the number of maxPatrons per party and creates the objects |
| EndGamePrompt | This class implements a view when the game is completed. It provides the option to play another game or to leave the game |

| | |
|---|---|
| EndGameReport | Print the report or not through a popup window. |
| Lane | This is the most significant class managing the whole lane. It contains methods to manage the lane related functionalities. It also contains variables to stop and halt the current game so all the functions related to the Running of the game are closely implemented with this class. Also, it implements PinSetterObserver to make PinSetterView coupled with this class and provide synchronization between them |
| LaneEvent | This class is the class to simulate the Event and store all related information to Lane Events like Current Bowler, Current Scores, etc. This class provides getters and setters for the information Stored. This class is helpful for getting the current state of the Lane if required. |
| LaneStatusView | This class is used in showing the lane status view of each lane. Each of its objects is embedded on the ControlDeskView, It provides options such as current bowler, no of pins down and the option to resume if in case its paused, etc. It implements LaneObserver and PinSetterObserver to provide the required information on each event. |
| LaneView | This class implements the detailed View for each lane like a detailed score of each bowler and displays the current state of each game |
| NewPatronView | This class implements a view for adding NewPatron to the system. Its also implements the action listener for the response to the buttons and implements getter and setter for getting information about the Patron |

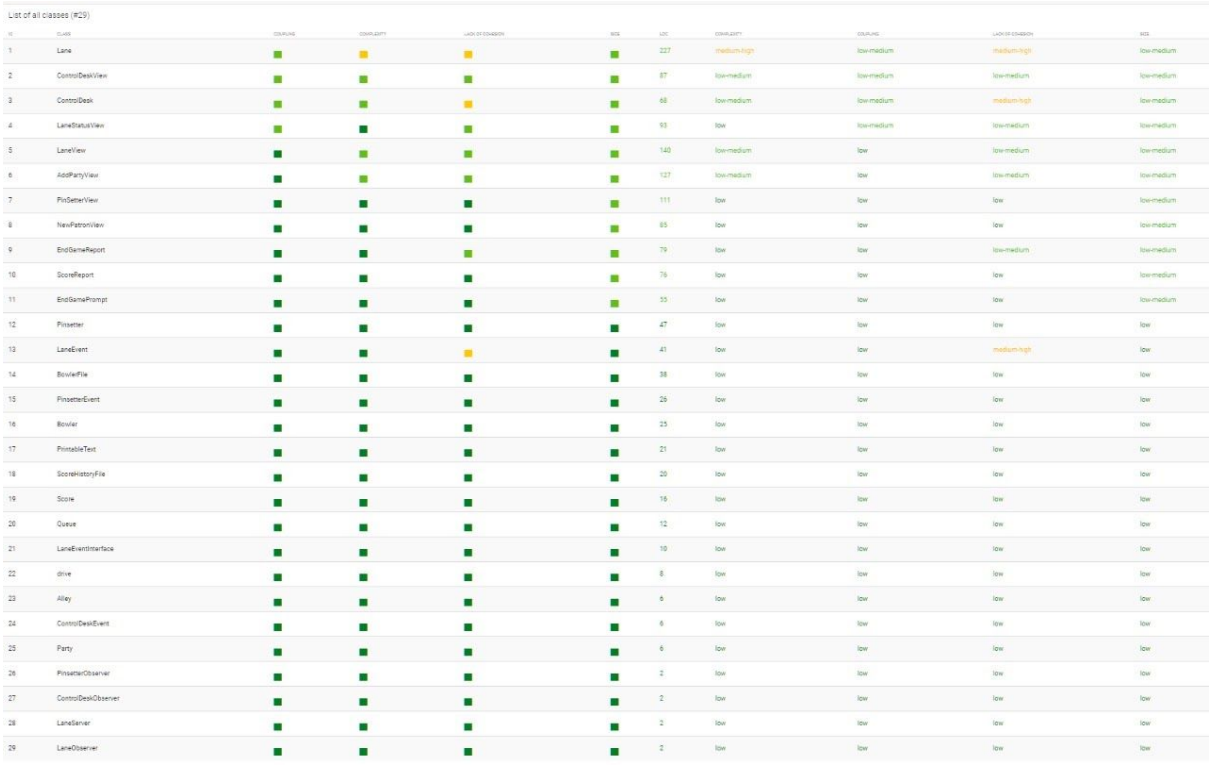
| | |
|---|---|
| Party | This class holds for details for Bowlers and provide getters and setters for them. |
| Pinsetter | This class is used to simulate PinSetter it holds the information's about the pins and methods to control them like resetting and send notification on event etc. |
| PinSetterEvent | This class contains methods for events related to pins like the number of pins down and check if foul is committed etc. |
| PinSetterView | This class implements the view for the Pinsetter and methods to change the view once the status of pins changes during the bowling event. |
| PrintableText | This class provides methods for converting the text to a printable format. |
| Queue | This class helps to provide a queue and related functions for managing the queues. |
| Score | This class is helpful in storing the scores of the players and related information like nickname, date and the score and contains related methods which have the relevance to score processing |
| ScoreHistoryFile | This class provides methods for storing and getting the score from the file for the given nickname. |
| ScoreReport | This class provides methods for generating the reports for the given person and formats them in a presentable format |

## Code smells

Applying the initial metric on the code revealed that the code had a lot of cyclomatic issues. That was one of the major things to be fixed. That was the common theme as this was the major component to refactor. The others were bad linting practices and lack of  cohesion.

# Brief - Refactored Code

The refactored code was thought of to remain similar to original with major linting and complexity removed. Nothing major was changed as going back and testing would have to be done a lot.



List of all classes (#29)

| | CLASS | COUPLING | COMPLEXITY | LACK OF COHESION | SIZE | LOC | COMPLEXITY | COUPLING | LACK OF COHESION | SIZE |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Lane | | | | | 227 | medium-high | low-medium | medium-high | low-medium |
| 2 | ControlDeskView | | | | | 87 | low-medium | low-medium | low-medium | low-medium |
| 3 | ControlDesk | | | | | 68 | low-medium | low-medium | medium-high | low-medium |
| 4 | LaneStatusView | | | | | 93 | low | low-medium | low-medium | low-medium |
| 5 | LaneView | | | | | 140 | low-medium | low | low-medium | low-medium |
| 6 | AddPartyView | | | | | 127 | low-medium | low | low-medium | low-medium |
| 7 | PinSetterView | | | | | 111 | low | low | low | low-medium |
| 8 | NewPatronView | | | | | 85 | low | low | low | low-medium |
| 9 | EndGameReport | | | | | 79 | low | low | low-medium | low-medium |
| 10 | ScoreReport | | | | | 76 | low | low | low | low-medium |
| 11 | EndGamePrompt | | | | | 55 | low | low | low | low-medium |
| 12 | Pinsetter | | | | | 47 | low | low | low | low |
| 13 | LaneEvent | | | | | 41 | low | low | medium-high | low |
| 14 | BowlerFile | | | | | 38 | low | low | low | low |
| 15 | PinsetterEvent | | | | | 29 | low | low | low | low |
| 16 | Bowler | | | | | 25 | low | low | low | low |
| 17 | PrintableText | | | | | 21 | low | low | low | low |
| 18 | ScoreHistoryFile | | | | | 20 | low | low | low | low |
| 19 | Score | | | | | 16 | low | low | low | low |
| 20 | Queue | | | | | 12 | low | low | low | low |
| 21 | LaneEventInterface | | | | | 10 | low | low | low | low |
| 22 | drive | | | | | 8 | low | low | low | low |
| 23 | Alley | | | | | 6 | low | low | low | low |
| 24 | ControlDeskEvent | | | | | 6 | low | low | low | low |
| 25 | Party | | | | | 6 | low | low | low | low |
| 26 | PinsetterObserver | | | | | 2 | low | low | low | low |
| 27 | ControlDeskObserver | | | | | 2 | low | low | low | low |
| 28 | LaneServer | | | | | 2 | low | low | low | low |
| 29 | LaneObserver | | | | | 2 | low | low | low | low |

# Metric Analysis