New Features

Assignment Three

Design And Analysis Of Software Systems

Overview

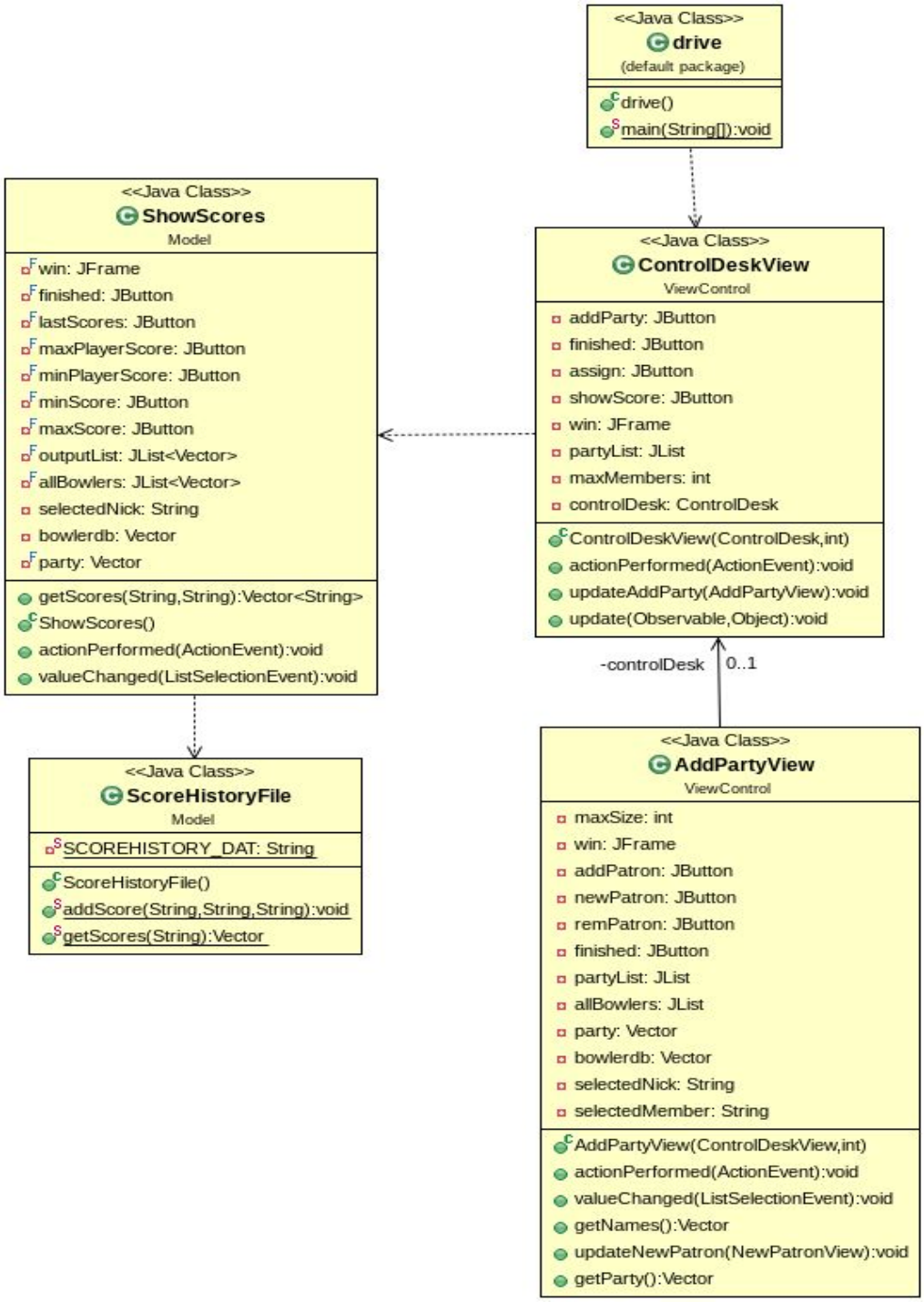
- The code has been extended and made working for multiplayer, where the maximum number of players is 6. There is an option to add and store players' names.

- A database layer to implement the persistence of the scores and players.
- A searchable view to make ad-hoc queries on the stored data has been provided.

- Pause and resume are implemented like save, load. Add an option to load and search for the user's name in saved games' parties and ask which one to load has been implemented.

UML Class Diagram

Interfaces, inheritance, generalization, association, aggregation, composition, cardinality and role indicators are indicated using appropriate arrows.

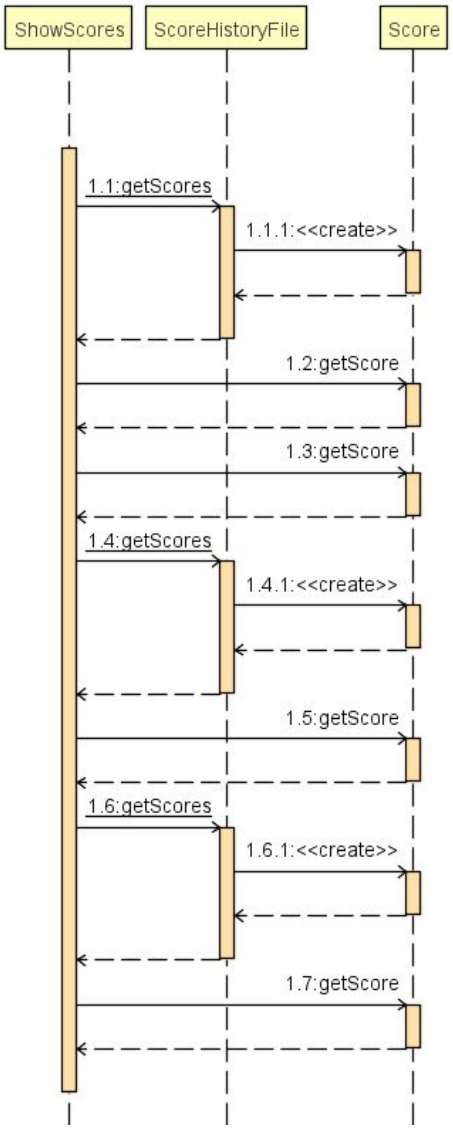


The new class added is the SetScores class as seen in the diagram above. This class stores the fields that are used to make the queries as a part of the feature implementation.

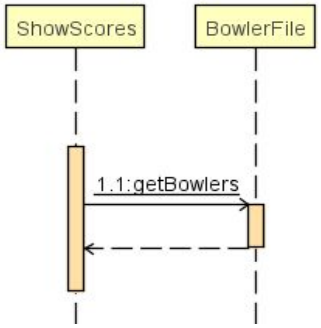
UML Sequence Diagram

Static and dynamic characteristics of the program are represented via the following sequence digrams.

Get Scores:



Show Scores:



Class Responsibility Table

Name	*Responsibility*
ShowScores	• Reads the SCOREHISTORY.DAT file using the getScores() method • Displays the scores records the actions and stores the changed values.

Functions

getScores()

`public Vector<String> getScores(String type,String nick)`

This function calls the `ScoreHistoryFile.getScores` function to retrieve all the scores of a particular player. Based upon the type parameter it gives the corresponding output like min of the player's scores, max of the player's scores or the last 5 scores of the player.
It returns a vector of strings which are then outputted to screen in actionPerformed function.

showScores()

`public showScores()`

This function constructs the window for displaying the scores as well as the various query buttons. It contains 3 panels: a score panel which shows the output of the queries, a bowlers panel which displays the list of bowlers in the software's database out of which the user can select so as to query that bowlers scores using the controls panel which contains the buttons

- Min Score: Lowest score overall
- Max Score: Highest score overall
- Player Min Score: Player's lowest score
- Player Max Score: Player's highest score
- Last scores by player: Player's score in the last 5 matches.
- Finished: closes the score window

actionPerformed()

```
public void actionPerformed(ActionEvent e)
```

This function gets called whenever we click any of the buttons in the controls panel.
It calls getScores() with the correct arguments corresponding to the button clicked and updates the values in the score panel.

valueChanged()

```
public void valueChanged(ListSelectionEvent e)
```

This function gets called whenever we select any of the bowlers and changes the value of the variable selectedNick which gives the nickname of the selected bowler which is needed in actionPerformed()