NEW FEATURES IMPLEMENTED:

OVERVIEW OF THE FEATURES IMPLEMENTED

FEATURE 1: Making code extensible for a Maximum of 6 players :

This is a minute change in the code. Since the code is already functional for a `maxPatronsPerParty` size of **5** players by default, all that was to be done in order to implement this feature was to modify this value from 5 to 6.

The file to be modified was `drive.java`. Here the class `drive` was modified.

Below is the snippet of code that was modified :

```
public class drive {

    public static void main(String[] args) {

        int numLanes = 3;
        int maxPatronsPerParty=6;

        Alley a = new Alley( numLanes );
        ControlDesk controlDesk = a.getControlDesk();

        ControlDeskView cdv = new ControlDeskView( controlDesk,
maxPatronsPerParty);
        controlDesk.subscribe( cdv );

    }
}
```

FEATURE 2: Searching the Database :

This feature has the functionality of enabling the users to make *ad-hoc* queries on the database.

The logic we have used to implement this is :

- Have 2 classes whereby we do not have any *Primitive obsession* code smell for our code. One that works with the UI for this feature and the other works with querying the .DAT file.
- In order to query on the database we have provide a *Dropdown* menu option that allows the user to select one among the many queries.
- Custom queries - searching for the highest/lowest score of a particular player can also be done using the input text-box provided.
- We have 6 functions for the drop down menu. The below code shows the function prototype snippets of the same :

```
    public static String giveHighestScoreSpecific(String playerName) ;

    public static String giveLowestScoreSpecific(String playerName) ;

    public static String giveLast5ScoreSpecific(String playerName);

    public static String giveHighestScoreAll() ;

    public static String giveLowestScoreAll() ;

    public static String giveTopPlayer() ;
```

- Error handling is also done.

FEATURE 3 Pause and Resume :

- This feature allows a party to pause their game for later.
- A UI button against each lane has been created that allows a user to pause an existing game in a lane
- The lane will remain frozen for that party until the party resumes their game and the game finishes.
- The text on the UI button toggles between *PAUSE* and *RESUME*
- Here are the snippets of the code reused from the existing codebase :

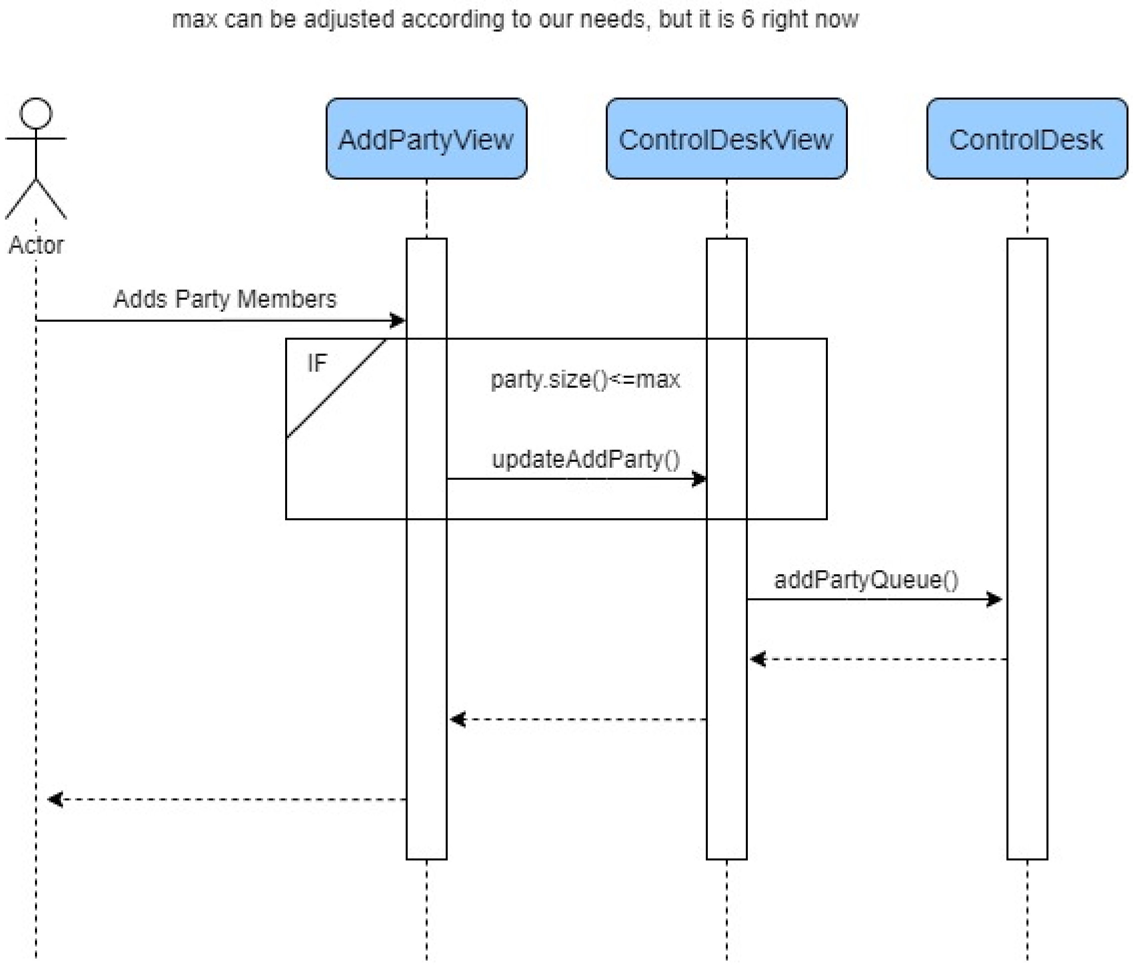
```
    /**
     * Pause the execution of this game
     */
    public void pauseGame() {
        gameIsHalted = true;
        laneSubscribe.publish(lanePublish());
    }

    /**
     * Resume the execution of this game
     */
    public void unPauseGame() {
        gameIsHalted = false;
        laneSubscribe.publish(lanePublish());
```

FEATURE 4 Saving an Existing Party's Game :

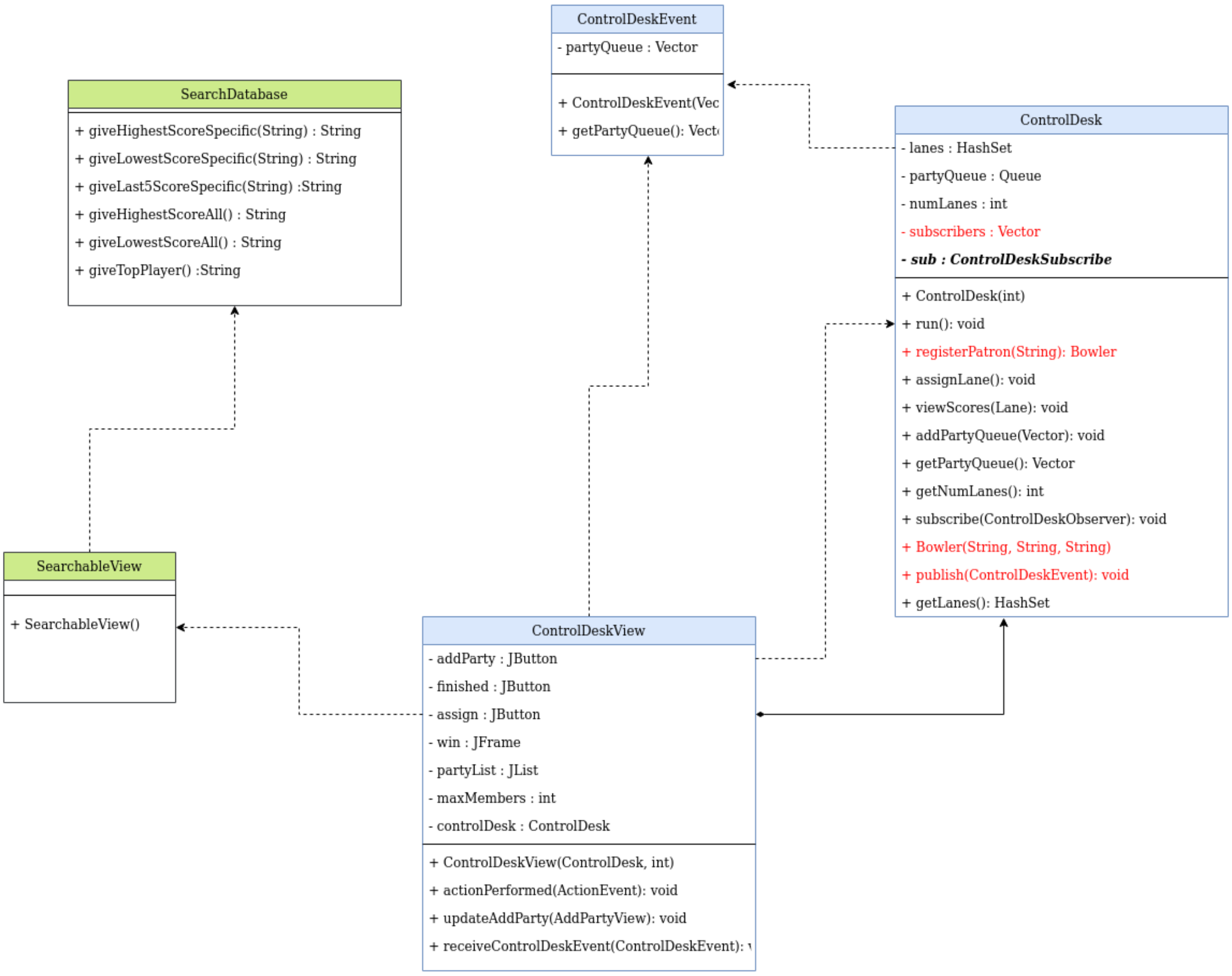
- Here we have created another UI button that allows the user to save the current status of an ongoing game in a particular lane.
- It is essentially storing a snapshot of the events of that lane into a file.

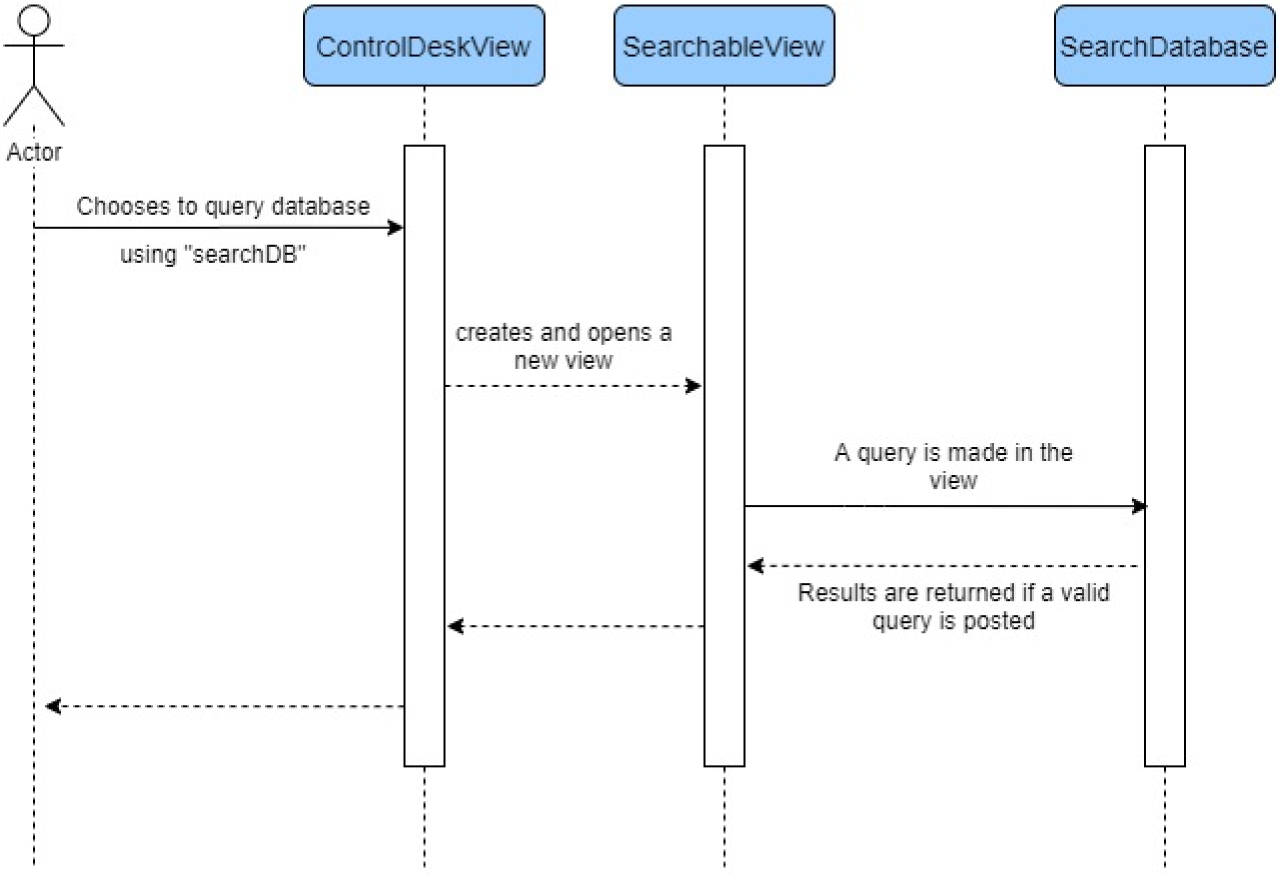
UML AND SEQ DIAGRAMS FOR NEW FEATURES:

FEATURE 1: Making code extensible for a Maximum of 6 players :

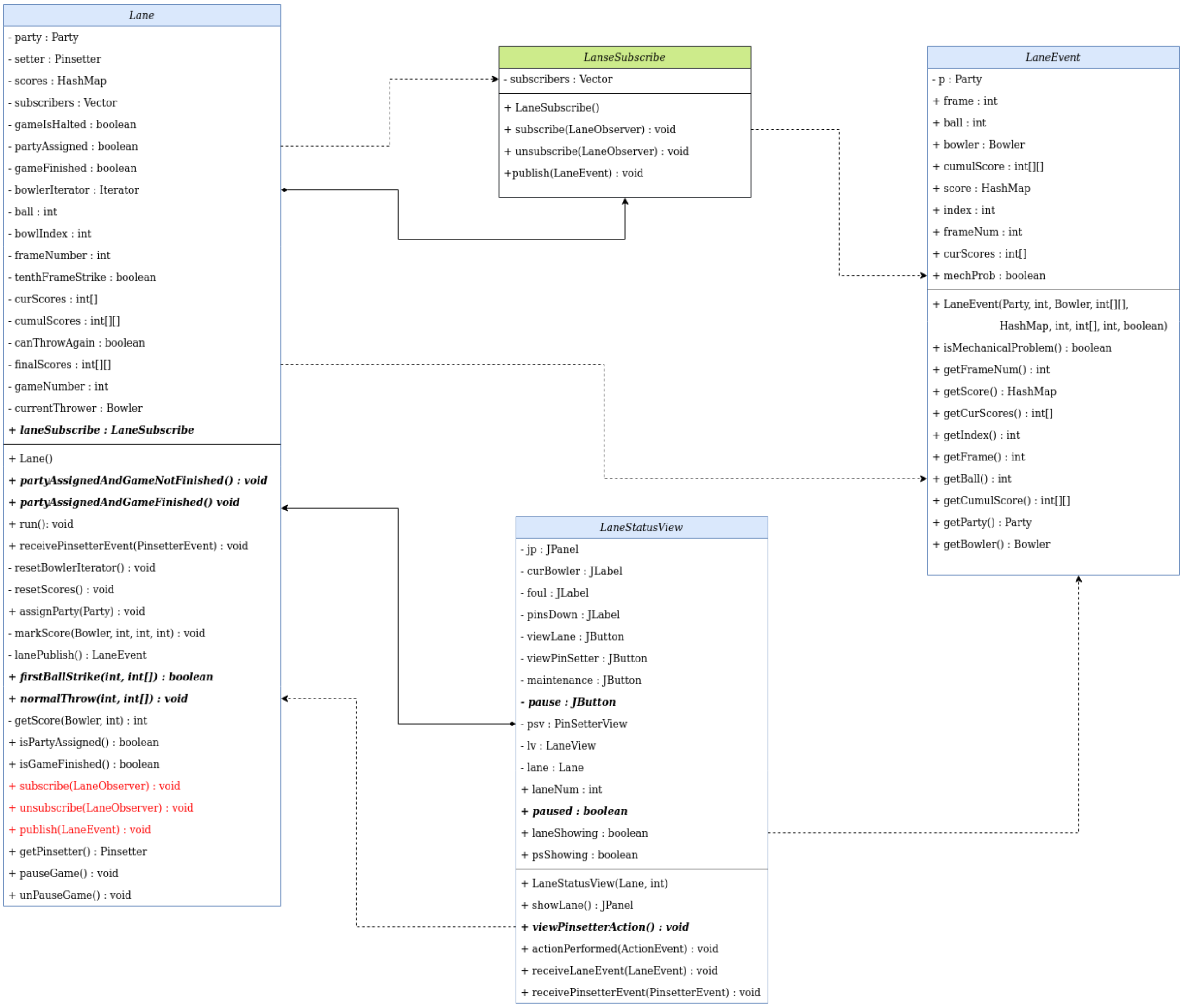


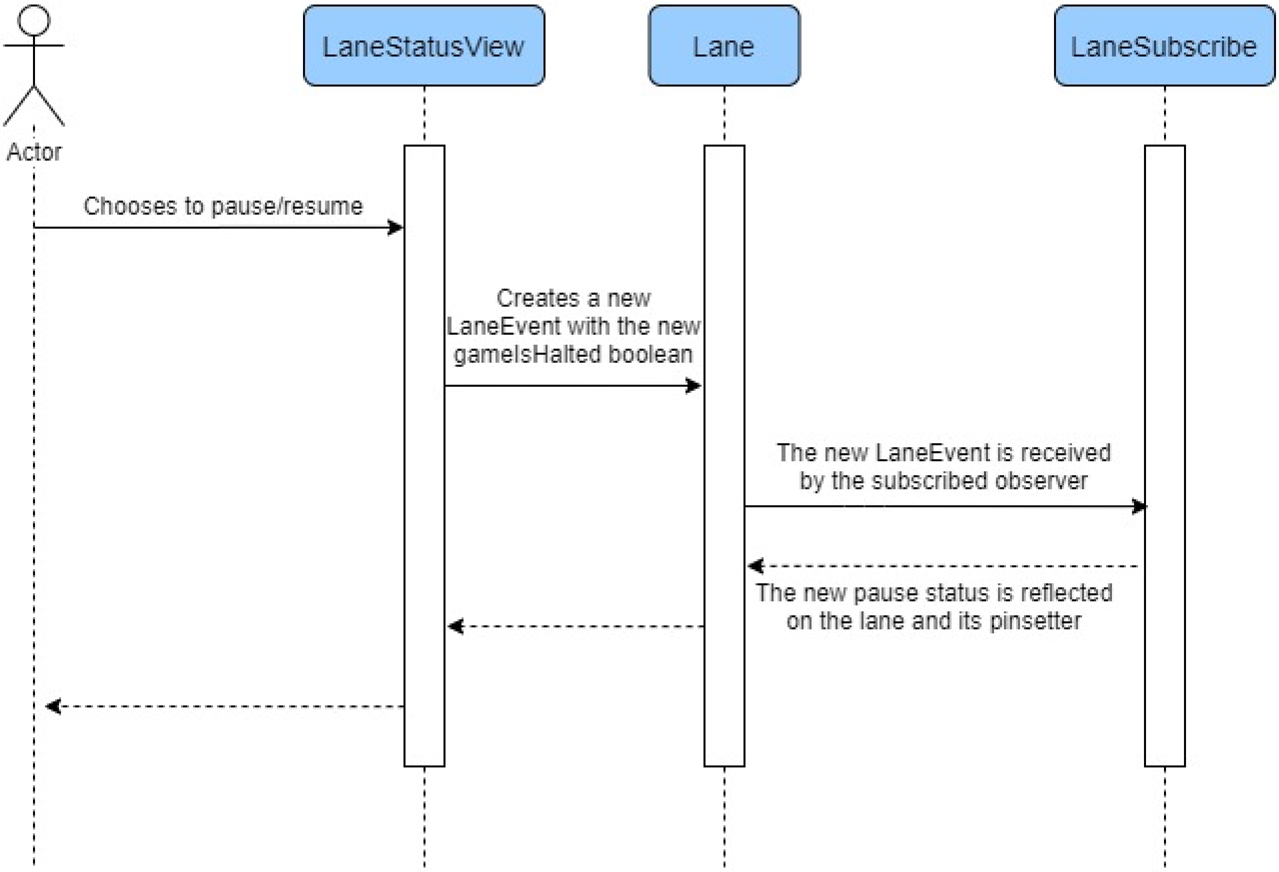
FEATURE 2: Searching the Database :


ControlDeskEvent
- partyQueue : Vector
+ ControlDeskEvent(Vec
+ getPartyQueue(): Vect
SearchDatabase
+ giveHighestScoreSpecific(String) : String
+ giveLowestScoreSpecific(String) : String
+ giveLast5ScoreSpecific(String) :String
+ giveHighestScoreAll() : String
+ giveLowestScoreAll() : String
+ giveTopPlayer() :String
ControlDesk
- lanes : HashSet
- partyQueue : Queue
- numLanes : int
- subscribers : Vector
- sub : ControlDeskSubscribe
+ ControlDesk(int)
+ run(): void
+ registerPatron(String): Bowler
+ assignLane(): void
+ viewScores(Lane): void
+ addPartyQueue(Vector): void
+ getPartyQueue(): Vector
+ getNumLanes(): int
+ subscribe(ControlDeskObserver): void
+ Bowler(String, String, String)
+ publish(ControlDeskEvent): void
+ getLanes(): HashSet
SearchableView
+ SearchableView()
ControlDeskView
- addParty : JButton
- finished : JButton
- assign : JButton
- win : JFrame
- partyList : JList
- maxMembers : int
- controlDesk : ControlDesk
+ ControlDeskView(ControlDesk, int)
+ actionPerformed(ActionEvent): void
+ updateAddParty(AddPartyView): void
+ receiveControlDeskEvent(ControlDeskEvent): v


Actor
ControlDeskView
SearchableView
SearchDatabase
Chooses to query database
using "searchDB"
creates and opens a
new view
A query is made in the
view
Results are returned if a valid
query is posted

FEATURE 3: **Pause and Resume :**


Lane
- party : Party
- setter : Pinsetter
- scores : HashMap
- subscribers : Vector
- gameIsHalted : boolean
- partyAssigned : boolean
- gameFinished : boolean
- bowlerIterator : Iterator
- ball : int
- bowlIndex : int
- frameNumber : int
- tenthFrameStrike : boolean
- curScores : int[]
- cumulScores : int[][]
- canThrowAgain : boolean
- finalScores : int[][]
- gameNumber : int
- currentThrower : Bowler
+ laneSubscribe : LaneSubscribe
+ Lane()
+ partyAssignedAndGameNotFinished() : void
+ partyAssignedAndGameFinished() void
+ run(): void
+ receivePinsetterEvent(PinsetterEvent) : void
- resetBowlerIterator() : void
- resetScores() : void
+ assignParty(Party) : void
- markScore(Bowler, int, int, int) : void
- lanePublish() : LaneEvent
+ firstBallStrike(int, int[]) : boolean
+ normalThrow(int, int[]) : void
- getScore(Bowler, int) : int
+ isPartyAssigned() : boolean
+ isGameFinished() : boolean
+ subscribe(LaneObserver) : void
+ unsubscribe(LaneObserver) : void
+ publish(LaneEvent) : void
+ getPinsetter() : Pinsetter
+ pauseGame() : void
+ unPauseGame() : void
LaneSubscribe
- subscribers : Vector
+ LaneSubscribe()
+ subscribe(LaneObserver) : void
+ unsubscribe(LaneObserver) : void
+publish(LaneEvent) : void
LaneEvent
- p : Party
+ frame : int
+ ball : int
+ bowler : Bowler
+ cumulScore : int[][]
+ score : HashMap
+ index : int
+ frameNum : int
+ curScores : int[]
+ mechProb : boolean
+ LaneEvent(Party, int, Bowler, int[][],
HashMap, int, int[], int, boolean)
+ isMechanicalProblem() : boolean
+ getFrameNum() : int
+ getScore() : HashMap
+ getCurScores() : int[]
+ getIndex() : int
+ getFrame() : int
+ getBall() : int
+ getCumulScore() : int[][]
+ getParty() : Party
+ getBowler() : Bowler
LaneStatusView
- jp : JPanel
- curBowler : JLabel
- foul : JLabel
- pinsDown : JLabel
- viewLane : JButton
- viewPinSetter : JButton
- maintenance : JButton
- pause : JButton
- psv : PinSetterView
- lv : LaneView
- lane : Lane
+ laneNum : int
+ paused : boolean
+ laneShowing : boolean
+ psShowing : boolean
+ LaneStatusView(Lane, int)
+ showLane() : JPanel
+ viewPinsetterAction() : void
+ actionPerformed(ActionEvent) : void
+ receiveLaneEvent(LaneEvent) : void
+ receivePinsetterEvent(PinsetterEvent) : void


Actor
LaneStatusView
Lane
LaneSubscribe
Chooses to pause/resume
Creates a new LaneEvent with the new gameIsHalted boolean
The new LaneEvent is received by the subscribed observer
The new pause status is reflected on the lane and its pinsetter