**Strengths of original design:**
Codebase was well structured,naming classes and methods was also deone relevant to the action they performed.For view functionality the files are created separately and perform their actions.Thus it was easier to comprehend and understand what is the flow of code.Comments are nice with reasonably good naming conventions.Observer Pattern Design.

**Fidelity of original design**:
Print-out and send email functionality in the design document wasn't working

| Java Class File | CodeSmell Description |
|---|---|
| AddPartyView | ● Code Repetition of making Jbuttons<br>● Deprecation of methods show(),hide()<br>● actionPerformed had many nested if-else conditions which leads to High Complexity |
| ControlDesk | ● viewScores() is Deadcode means unused<br>● It has extensively used methods like addpatyQueue() and getpartyQueue |
| ControlDeskView | ● Copy paste of Making Jbuttons and assignPanel button is not used |
| Lane | ● Redundancy of methods like lanePublish() and publish were always used together<br>● Very Large Class of many methods and variables<br>● getScore(),receivePinsetterEvent(),run() have very high complexity due to more nested if-else conditions |
| LaneStatusview | ● actionPerformed() had repeated if conditions which leads to high conditional complexity |
| LaneEvent | ● Unused getFrame() and getCurScores() functions and data class is used for storage and getter methods only.No operations on data.It has a Long Parameter List. |
| NewPatron | ● Rewritten of Jbuttons or Jpanel |
| PinsetterView | ● Making Buttons Rewritten code |

**Refactoring:**

**1.Unused Variables:**Some variables which are declared initially are not used later in the code.Example selectedNick,selectMember in NewPatron.jave.They can also point to a lack of proper design when not used.unused variables are removed in every java file.

**2.Public Modifier in Interfaces**:All methods in an interface are already public and abstract so there was no need to make them public explicitly.Interfaces public modifier was used with many functions.

```
public interface LaneEventInterface extends java.rmi.Remote {
    public int getFrameNum( ) throws java.rmi.RemoteException;
    public HashMap getScore( ) throws java.rmi.RemoteException;
    public int[] getCurScores( ) throws java.rmi.RemoteException;
    public int getIndex() throws java.rmi.RemoteException;
    public int getFrame() throws java.rmi.RemoteException;
    public int getBall() throws java.rmi.RemoteException;
    public int[][] getCumulScore() throws java.rmi.RemoteException;
    public Party getParty() throws java.rmi.RemoteException;
    public Bowler getBowler() throws java.rmi.RemoteException;

}
```

There we removed the public modifiers from Interface files.

**3.Empty Catch Statements:**In many files Catch statements were empty.If catch code would not print anything it would be difficult to debug.So appropriate errors were printed.Example

```
public void receiveLaneEvent(LaneEvent le) {
    if (lane.isPartyAssigned()) {
        int numBowlers = le.getParty().getMembers().size();
        while (!initDone) {
            //System.out.println("chillin' here.");
            try {
                Thread.sleep(1);
            } catch (Exception e) {
            }
        }
```

Every empty catch statements are refactored

```
public void receiveLaneEvent(LaneEvent le) {
    if (lane.isPartyAssigned()) {
        int numBowlers = le.getParty().getMembers().size();
        while (!initDone) {
            // System.out.println("chillin' here.");
            try {
                Thread.sleep(1);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

**4.Unnecessary Imports:**Unused and useless imports affects the code's readability.Example java.text.* which is never used in the class.Removed unused imports in every file.

**5.Deprecated warnings:**The deprecated warnings are removed wherever possible.Some examples are show(),hide(),new Integer etc., are modified as SetVisibility(true),SetVisibility(false),Integer.valueOf.

**6.Method typo mistakes:**In EndGamePrompt has a method named `distroy` which is a typo.Renamed to destroy in every files.method typo mistakes are fixed.

**7.Redundant casting to various fields:**Unnecessary casting expressions make the code harder  to understand.So removed redundant casting in the codebase.

***Ideas

**8.Code Repetition:**The main reason for creation for the duplicate code is copy and paste programming.It will increase Lines Of Code(LOC).In AddPartview the same code is repeated for making buttons.So this could be easily made into function can be called multiple times.

```
addPatron = new JButton("Add to Party");
JPanel addPatronPanel = new JPanel();
addPatronPanel.setLayout(new FlowLayout());
addPatron.addActionListener(this);
addPatronPanel.add(addPatron);

remPatron = new JButton("Remove Member");
JPanel remPatronPanel = new JPanel();
remPatronPanel.setLayout(new FlowLayout());
remPatron.addActionListener(this);
remPatronPanel.add(remPatron);

newPatron = new JButton("New Patron");
JPanel newPatronPanel = new JPanel();
newPatronPanel.setLayout(new FlowLayout());
newPatron.addActionListener(this);
```

We encountered the same repetition of code in MakingPanel,Making TextField.

**9.Law of Demeter(LOD):**The law of demeter or functions attempts to minimize coupling between classes in any program.The law of demeter is often described as “talk to friends,not to strangers”.

```
public void receiveLaneEvent(LaneEvent le) {
    if (lane.isPartyAssigned()) {
        int numBowlers = le.getParty().getMembers().size();
        while (!initDone) {
```