Code Smells Within Classes

1. Comments

1	All .java files	Several Javadoc errors in terms of syntax and formatting	All comments corrected and new comments explaining the whys of the code written.

2. Long Methods/ Parameters And Duplication

1	All .java files	Many files with redundant code	Long methods refactored. Parameters excluded wherever necessary. DRY followed.

3. Conditional Complexity

1	LaneStatusView.java	Redundant nested if statements if (lane.isPartyAssigned())	Nested if statements combined

4. Combinatorial Explosion

1	All .java files	Lots of code that does almost the same thing.. but with tiny variations in data or behavior.	Base classes to inherit the functionality from using parameters to carry out the modification

5. Large Classes

1	ControlDesk.java	Large class used by all subclasses to inherit from.	Decided to fix the ControLDesk Observer/Observable pattern as well, for consistency. Removed the ControlDeskEvent.java file, now just passing the ControlDesk itself to Observers.
2	Lane.java	Large class used by all subclasses to inherit from.	Restructured into smaller class with inheritance only where required.

6. Uncommunicative/Inconsistent Name

1	All .java files	Random non-descriptive single letter naming seen in many files.	Used standard terminology and stuck to it throughout the methods

7. Dead Code

1	LaneEvent.java	An empty statement (semicolon) not part of a loop.	Removed semicolon
2	LaneServer.java	An empty statement (semicolon) not part of a loop.	Removed semicolon
3	LaneStatusView.java	Commented code statements.	Removed multiple sections of

			commented code.
4	LaneStatusView.java LaneView.java	Deprecated function new Integer()	Changed to updated newer functionality Integer.valueOf()
5	LaneView.java NewPatronView.java	Deprecated function show() hide()	Changed to updated newer functionality setVisibility(true) setVisibility(false)
6	PrintableText.java		Changed the path in the BowlerFile.java and the ScoreHistoryFile.java classes to reflect the correct database file locations so that the project is runnable during the series of refactorings to be made.

8. Speculative Generality

1	EndGamePrompt.java NewPatronView.java	Same line declaration for multiple variables.	Use one line for each declaration, it enhances code readability.

Code Smells Between Classes

9. Alternative Classes With Different Interfaces

1	Lane.java	Two classes are similar on the inside, but different on the outside, they can be modified to share a common interface	Removed the implements LaneObserver class in the GUI elements, replaced with implements Observer. Changed the logic for adding an Observer

			to Lane's list of observers to be in line with the java implementation in the appropriate GUI classes.
2	ControlDesk.java	Two classes are similar on the inside, but different on the outside, they can be modified to share a common interface	Changed ControlDesk from extending Thread to extending Observable, and it now implements Runnable.

10. Data Class

1	LaneStatusView.java	Classe that passively stores only data.	Added a few getters within Lane so that LaneStatusView is able to generate the end-of-game routine originally found in Lane

11. Data Clumps

1	package ViewControl	Same data hanging around together, then verify whether it belongs together.	Moved all of the View/Control elements of the program to a new ViewControl package.
2	package Model	Same data hanging around together, then verify whether it belongs together.	Moved all of the Model elements of the program to a new Model package.

12. Refused Bequest

1	Lane.java	Inherit from a class but	In Lane, I got rid of the

		never use any of the inherited functionality	recievePinsetterEvent method and added the standard Observer.update method. I decided to keep the PinsetterEvent since it encapsulates data not inherit to Pinsetter.

13. Inappropriate Intimacy

1	BowlerFile.java ControlDesk.java	Classes that spend too much time together, or classes that interface in inappropriate ways.	Changed the class visibility of BowlerFile.java and ControlDesk.java so that the GUI components would have access to them being from a different package.
2			

14. Indecent Exposure

1	LaneEvent.java	Data items that were unnecessarily public. HashMap score; int frameNum;	Use explicit scoping instead of accidental usage of default package private level
2	LaneStatusView.java	Data items that were unnecessarily public. JLabel foul;	Replaced by a local variable.
3	LaneStatusView.java	Data items that were unnecessarily public. Boolean laneShowing;	Use explicit scoping instead of accidental usage of default package private level
4	LaneView.java	Data items that were unnecessarily public.	Use explicit scoping instead of accidental usage

		Container cpanel; JPanel[][] scores; JLabel[][] scoreLabel; JButton maintenance;	of default package private level

15. Feature Envy

1	Lane.java	Methods that make extensive use of another class.	The Lane.java file had some GUI code in it that needed migrating to the correct classes.

16. Lazy Class

1	PinSetterView.java	Classes should pull their weight	Changed the constructor of the PinSetterView.java class to accept a Pinsetter so that it has the responsibility of adding itself as an observer.
2	Pinsetter.java	If a class isn't doing enough to pay for itself, it should be collapsed or combined into another class.	Removed the PinsetterObserver class.
3	drive.java	If a class isn't doing enough to pay for itself, it should be collapsed or combined into	Removed the redundant alley class. Instead, drive simply creates the ControlDesk directly and an entire class is able to

		another class.	be removed. These changes are clear in drive.java.

17. Message Chains

1	Lane.java	Long sequences of method calls to get routine data.	Deleted the redundant lanePublish() method within Lane.java. Replaced the publish() method with the proper Observable notify operations within Lane.java.

18. Middle Man

1	LaneEvent.java	Class that is merely wrappers over other classes or existing functionality in the framework	Removed the LaneEvent imports in the GUI.

19. Divergent Change

1	LaneStatusView.java LaneView.java	Changes to a class that touch completely different parts of the class.	Fixed the issues with migrating the receiveLaneEvent code to the update method in LaneStatusView.java. Fixed the issues with migrating the receiveLaneEvent code to the update method in

			LaneView.java.

20. Shotgun Surgery

1	Lane.java	A score change in one class requires cascading changes in several related classes.	Added a publish() call in Lane once it is determine the game is finished. This allows the GUI to finalize the score display one last time.

21. Solution Sprawl

1	ScoreCalculator class	Simplifying and consolidating your design to ensure no more than one class is needed to make	Added the ScoreCalculator class and migrated all of the getScore functionality originally belonging to main into that class.