Code smells and antipatterns in design and their solutions

Type	Class name	Issue	Resolving
God class	Lane	Handling multiple functionalities like managing lane, calculating score, generating reports etc	Broke class into 2 classes. Shifted some methods into appropriate classes.
Feature Envy	Lane	Calling methods and attributes from several classes multiple times like pinsetter, score, party	Shifted some methods into appropriate classes.
Change Preventers	LaneView, PinsetterView	For changing a button in overall design each view has to be changed	Create a general view class which is inherited by all.
Change Preventers/ more than one responsibility	Pinsetter	Any change in Simulation would require to change pinsetter class	Broke pinsetter and separate functionality of simulation
Lava flow	Lane, Laneserver, Bowler, ScoreReport	Commented code with no descriptions Redundant	Cleaning and added appropriate comments
Spaghetti code	Laneserver Lane Pinsetter AddPartyView	Dead code and duplicate code. Unnecessary comments	Cleaning and changes to mvc structure
Modularization	All code in same structure	Decreases readability and flexibility.	Created MVC structure
Conditional complexity	NewPatronView LaneView Other view classes	Increases complexity	Change action listeners to lambda functions

Observer class - It is better to implement your own logic rather than using a Java observable class. As it is deprecated after JDK 8. Keeping this logic in mind we have not changed things to observable class.