Object-Oriented Project Part 6: FINAL REPORT

Team: Nolawee Mengist
Shawn Polson
Theerarun Tubnonghee (Steve)
Jose Escobar

1. ***List the features that were implemented (table with ID and title).***

User Requirements		
ID	**Requirement**	**Implemented?**
UR-001	As a player, I can view the rules of Blackjack.	Yes ✓
UR-003	As a player, I can be dealt cards	Yes ✓
UR-004	As a player, I can place bets using chips	Yes ✓
UR-005	As a player, I can know the total amount of “money” I own.	Yes ✓
UR-006	As a player, I play against a CPU dealer	Yes ✓
UR-007	As a player, I can “hit” during my turn.	Yes ✓
UR-008	As a player, I can “stand” during my turn.	Yes ✓
UR-011	As a player, I can win the bet if I get “21”	Yes ✓
UR-012	As a player, I lose the bet if I “bust” over 21.	Yes ✓
UR-013	As a player, I can play through as many rounds as I want.	Yes ✓
UR-014	As a player, I can still lose to the dealer even if I get 21	Yes ✓
UR-016	As a player, I can save the current state of the game and continue to play.	Yes ✓

UR-017	As a player, I can continue my saved state of the game	Yes ✓
UR-018	As a player, I can start a new game and play with some amount of chips.	Yes ✓

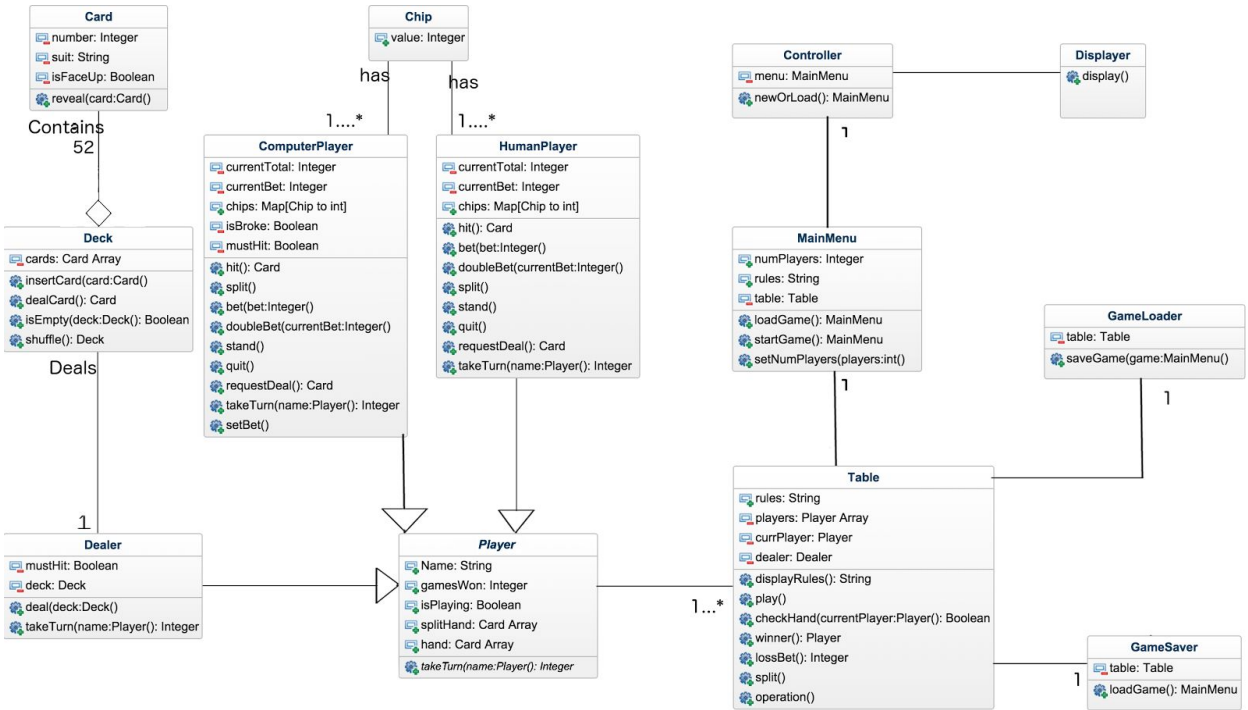
Functional Requirements		
ID	**Requirement**	**Implemented?**
FR-001	As a player, I lose if I run out of chips.	Yes ✓
FR-002	Card deck should be shuffled before play or when deck runs out	Yes ✓
FR-003	Dealer and computer players must run through their turns when needed	Yes ✓
FR-004	Computer players must track their own total money and not bet past 0	Yes ✓
FR-005	Dealer and computer players must not hit when their currentTotal>=16	Yes ✓
FR-006	When player saves the game and exits, all totals and current cards in deck must be saved for reloading	Yes ✓
FR-007	User can navigate through the game.	Yes ✓

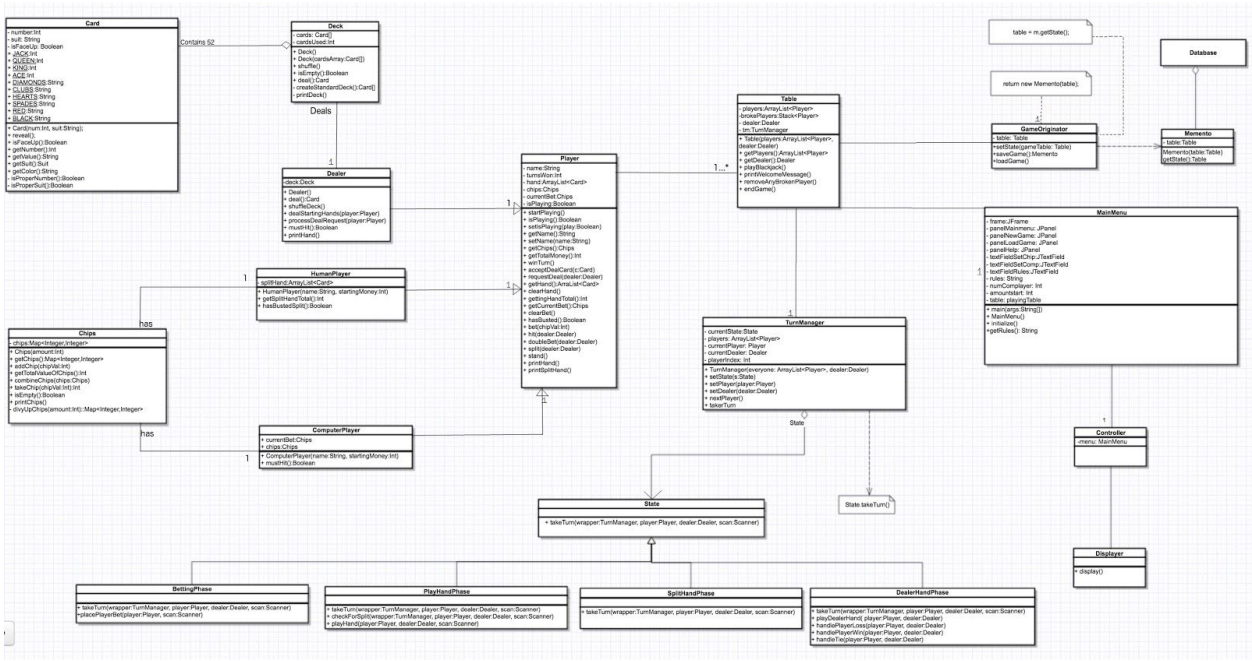
Non-Functional Requirements		
ID	**Requirement**	**Implemented?**
NFR-001	The game can be played on the command line before our GUI is implemented.	Yes ✓
NFR-002	The game should work properly each time it's opened	Yes ✓

2. ***List the features that were not implemented from Part 2 (table with ID and title).***

User Requirements		
ID	**Requirement**	**Implemented?**
UR-002	As a player, I can pick how many computer players to play with.	No ✖
UR-009	As a player, I can “double” during my turn.	No ✖
UR-010	As a player, I can “split” during my turn if my first two cards are of the same denomination.	No ✖
UR-015	As a player, I can drop out and end the game with my winnings any time	No ✖

3. ***Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.***



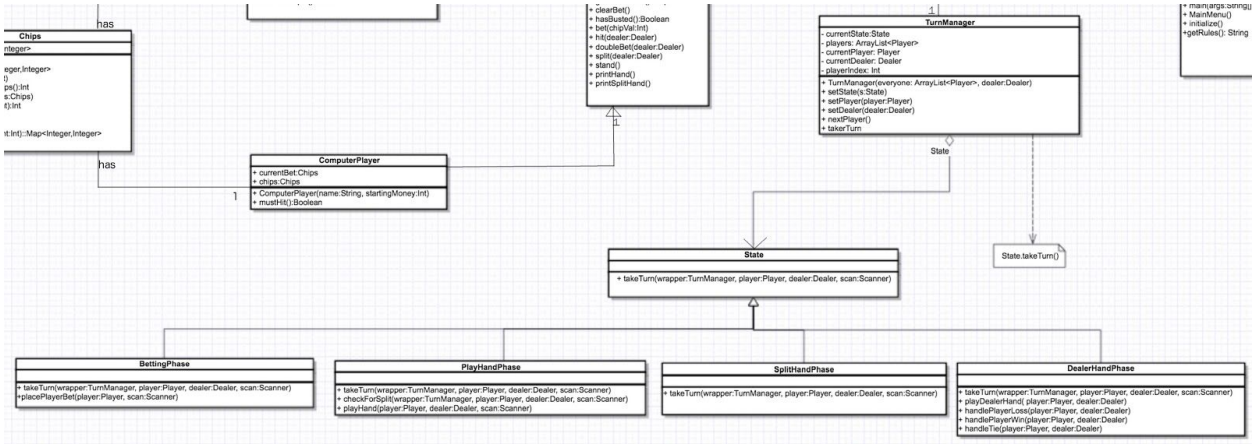


- **What changed:** Our chip class changed a lot. It used to do nothing more than wrap an int value, but we upgraded it to wrap a map (chip's value -> # of those chips). The new class more faithfully represents chips and also has various methods for doing things with chips like divvying them up based on a number value and combining piles of chips (to handle winning a bet, for example). We also added two design patterns: in the form of a new "TurnManager," the State design pattern replaced the takeTurn() methods in the Player classes; and in the form of GameOriginator & Memento, we also replaced our GameSaver/Loader classes in the new diagram. The Table class in our old diagram lost most of its methods from one diagram to the next because our original ideas about how the Table would work didn't pan out. The Card class grew significantly with the addition of constants to model things like suit and color, and Deck grew a little too to make it more useful. The Player class also grew a lot because we were able to put way more functionality in that parent class than we thought, resulting in tiny subclasses (HumanPlayer, ComputerPlayer, and Dealer). Saving/Loading features were removed from the MainMenu of our new class diagram because, as previously stated, we handle that with the Memento pattern now. Generally speaking, every class from our original class diagram is different now in our new class diagram because our first diagram was not as well thought-out as it could have been. In hindsight, we recognize that better Activity/Sequence diagrams could have made that more clear to us from the beginning, and had we thought through it more, our final implementations might have matched up more closely to our original design. But we learned a lot along the way!

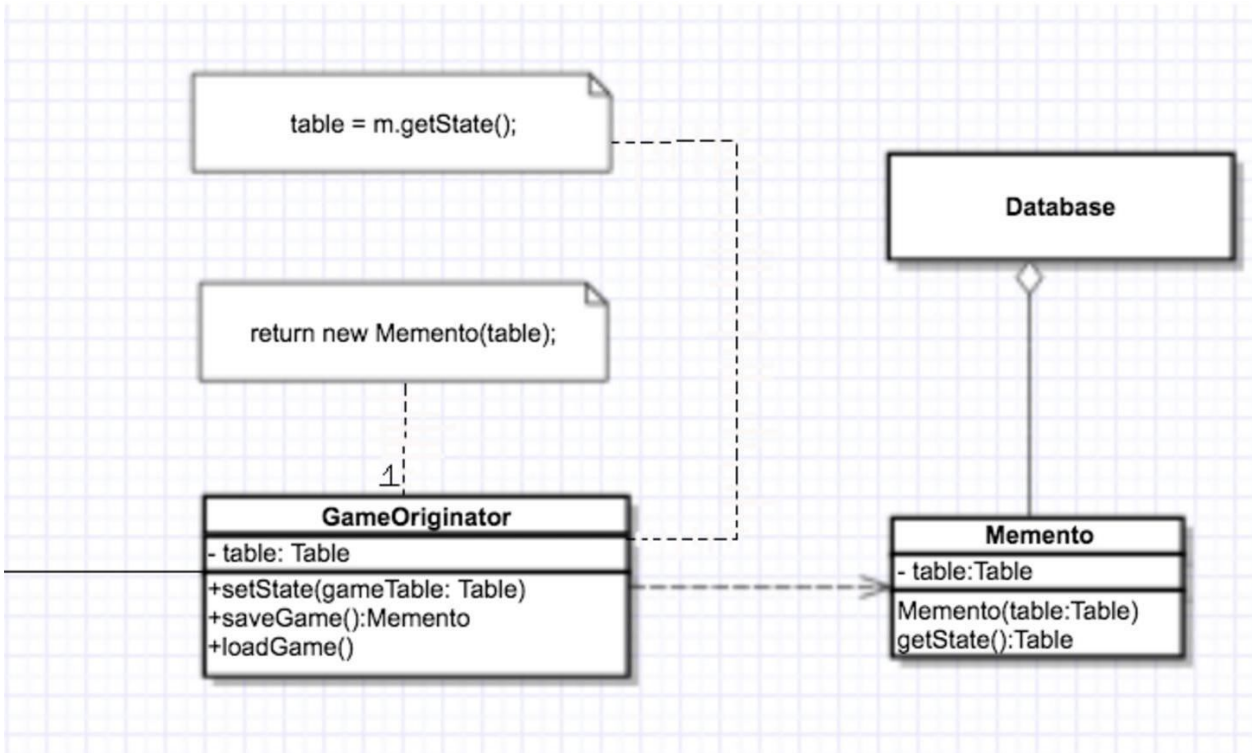
4. ***Did you make use of any design patterns in the implementation of your final prototype? If so, how? Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF).***

- Yes, we used two design patterns: State and Memento.

State:



Memento:



5. ***What have you learned about the process of analysis and design now that you have stepped through the process to create, design, and implement a system?***

We learned many things by going through this process. To start, the vast differences between our original class diagram and our final diagram taught us the value of making strong Activity/Sequence diagrams before starting an implementation. We also saw firsthand how much code can improve by using the object-oriented ideas we learned in class; by making the right kind of objects and giving them the right abilities, we were able to faithfully model real-world Blackjack objects (like cards, chips, dealers, etc.), which made our code uniquely easy to reason with. We were also able to write code that is “open for extension,” as our professor often put it, which made development unusually pain-free when major refactors were needed. On top of this, we learned how important it is to factor in technical aspects that are not in our class diagram. A perfect example was installing jar files needed to run Hibernate. The project as a whole was also a good exercise in treating software development as it is treated in industry; using IDEs to develop, following good git practices, working in feature branches that get merged into Master only when ready, etc. These were all invaluable learning experiences.