what went well - The overall design of the game was easily understandable, and the actual structure of the code base had minimal issues.

what didn't work regarding your original design and your team plan -
In our original design we missed a few implementations of certain card and move features. For example, in our original design we did not implement additional features that are necessary to functionality such as knowing if a center card was selected, there isn't an implementation to "make" the cards go to another player, or to know who is the player and who they should go to. These are things that may not have been implemented in our original design that would leave us with a problem later in the project.

what needed to be changed (features)-

Code added for features 1-2: There were lots of changes added for these features as they went across multiple classes, and thus I'm not going to put them here.

Code added for feature 3: Pretty simple, added a button class and dependencies to indicate new game type.

Code added for feature 5: During play, the players alternate selecting the card at the top of their pile

```
Pile fromPile;
        Pile toPile;
        if(player.getPlayerNum()%2 == 0) {
                fromPile = table.getPile(DROP_PILE);
        toPile = table.getPile(PICKUP_PILE);
        } else {
                fromPile = table.getPile(DROP_PILE2);
        toPile = table.getPile(PICKUP_PILE2);
        }
    Card c = fromPile.getCard(e.getId());
    if (c == null) {
        return new DoNothingMove();
    }
    return new WarCardMove(c, player, fromPile, toPile);
  }
```

Code added for feature 6: When a player wins the cards in the center they select the winning card and the cards are transferred to the bottom of their pile. A player that selects a card incorrectly is ignored.

```
  public Move battle_win(ConnectEvent e,Player p1, Player p2, Card player1card, Table table,
                Card player2card){
        Pile toPile;
```

```
        Pile fromPile =table.getPile(DROP_PILE) + table.getPile(DROP_PILE2);

    if(player1card.getRank() > player2card.getRank()) {
         System.out.println("Player 1 wins the battle!");
         toPile = table.getPile(PICKUP_PILE);
         return new WarCardMove(c, p1, fromPile, toPile);
    }
    else {
        System.out.println("Player 2 wins the battle!");
        toPile = table.getPile(PICKUP_PILE);
                return new WarCardMove(c, p2, fromPile, toPile);
    }


}
public Move collect_other_pile(Table table, Player Winner, CardEvent e) {
        Pile fromPile;
        Pile toPile;
        if(Winner.getPlayerNum()%2 == 0) {
                toPile = table.getPile(DROP_PILE);
        fromPile = table.getPile(PICKUP_PILE);
        } else {
                toPile = table.getPile(DROP_PILE2);
        fromPile = table.getPile(PICKUP_PILE2);
        }
    Card c = fromPile.getCard(e.getId());
    if (c == null) {
        return new DoNothingMove();
    }
    return new WarCardMove(c, Winner, fromPile, toPile);
}
```

The first method will take the winner's pile in the center and add it to their deck, and the second method will take the loser's pile and add it to the winer's deck aswell

Code added for feature 7: Needed to add new classes WarBattleRules, WarBattleDealMove, WarBattleCardMove, that all extended existing classes so no old code needed to be changed. These classes simply add features and override methods of their parent classes when created and called.

Code added for feature 8: No changes necessary as the score was simply updated in methods of the new classes for WarBattles.

Code added for feature 9: "When a player runs out of cards the other player immediately wins. Set the title of the game to 'Player X Wins'"
To Table.java: Player getOtherPlayer(Player player);
WHY- add get other player function

To GameController.java:

```
    /**
    * Handles the end of the game
    * @param endPlay
    * @param game2
    */
```

WHY- additional comments

To P52Rules.java:

```
                    // the other player has won
                    return new EndPlayMove(table.getOtherPlayer(player));
```

WHY - When player runs out of cards, the EndPlayMove function will be called to get the other player who has won

To EndPlayMove.java:

```
        private Player winner;

        public EndPlayMove(Player winner) {
                this.winner = winner;
        }

        public void apply(Table table) {
                table.setMatchOver(true);
                if (winner != null) {
                        table.setMatchOver(true);
                }
        }
        public void apply(ViewFacade view) {
                view.send(new SetGameTitleRemote(String.format("Player %s Wins!",
winner.getPlayerNum())));
```

WHY- Sets the winning player and displays the text “Player %s Wins!”

To TableBase.java:

```
        @Override
        public Player getOtherPlayer(Player player) {
                Optional<Integer> otherPlayerId = this.players.keySet().stream().filter(
                                id -> id != player.getPlayerNum()
                ).findFirst();

                return otherPlayerId.map(integer -> this.players.get(integer)).orElse(null);
        }
```

WHY- Takes in a player and returns the other player, and interface since its a 2 player game

Code added for feature 11: “When the game is finished (one player wins) show the deal button. This will be helpful.”

To EndPlayMove:

```
P52DealButton dealButton = new P52DealButton("DEAL", new Location(50, 50));
                view.register(dealButton);
```

WHY- when the game ends (EndPlayMove) the deal button will be shown