

# Simulating Blackjack using Java

## Abstract:


The real world problem I explored in this project was the card game Blackjack where I coded classes of a card, a deck, a hand, and finally the actual game. In this project, I broke down the code into different classes to make it easier to understand and simpler to read which helped better understand the problem. The core CS concepts used in this project were inheritance and polymorphism through all the classes that were connected to each other. My key findings were the probabilities of the player or dealer winning blackjack or the game ending in a tie if the game is completely randomized and run 1000 times.

## Results:

The first version of the code that I ran was simple and just looked at a single game of Blackjack. Once that was done and sufficiently working, I implemented and ran a Simulation class which simulated a total of 1000 games of Blackjack. The metrics I ended up using and reporting followed the game and rules of blackjack, so I used the number of wins by the player, by the dealer, and then the number of games that ended up tied.

```
svitakiran@Svitas-MacBook-Pro Project01 % cd /Users/svitakiran/Desktop/Project01 ; /usr/bin/env /Library/Java/JavaVirtualMachines/temurin-21.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/svitakiran/Library/Application\ Support/Code/User/workspaceStorage/5faf14d4f7520ad8f5ad3904141e2314/redhat.java/jdt_ws/Project01_410e365c/bin Simulation
-----
Total player wins: 413
Player win percentage: 41%
Total dealer wins: 494
Dealer win percentage: 49%
Total ties: 93
Tie percentage: 9%
```

Caption: This is the output with the metrics given earlier of a simulated 1000 games and their percentages.



mygames.txt

```
Dealer busts. Player wins!
Player Hand: [6, 9, 6] : 21 (Total: 21)
Dealer Hand: [2, 10, 11] : 23 (Total: 23)

Game result: 1

Dealer busts. Player wins!
Player Hand: [10, 10] : 20 (Total: 20)
Dealer Hand: [9, 4, 11] : 24 (Total: 24)

Game result: 1

Dealer wins!
Player Hand: [3, 8, 5] : 16 (Total: 16)
Dealer Hand: [9, 10] : 19 (Total: 19)

Game result: -1
```

Caption: This picture shows 3 games played and their outcomes, where 1 means the player wins and -1 means the dealer wins.

The outcome of my experiments shows me that both the player and dealer would win and lose pretty consistently, which shows that there isn't really a big bias with the shuffling and the dealing of the cards. Everything is pretty evenly distributed and I think that the game seems to be reliable.

### **Extensions:**

I didn't do any extensions for this project.

### **Reflection:**

1. The difference between this new test and the old one is that this new shuffle method is structured in a way that two random cards will be picked and then they will swap places. This is better and more efficient than the old shuffle method.
2. These take a constant  $O(1)$  time because they don't access the actual list of the array or their indexes. Instead, they just use the functions remove and add the First or Last element so they don't have to access and iterate through the list, meaning that it will take a constant amount of time for these functions to run.

### **Acknowledgements:**

I worked with Kamalani for parts of the project but we didn't see each other's code. I also used w3schools and looked up questions on google. I also got help from a TA during the lab period on Tuesday. I then went to TA hours on Sunday for help again.