

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Formula Sheet

Standard Deviation $\sigma = \sqrt{\sigma^2}$	Multiplicative Law of Probability
Variance $\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2$	$P(A \cap B) = P(A)P(B A) = P(B)P(A B)$
Permutation $nPr = \frac{n!}{(n-r)!}$	General Addition Rule
Multinomial Coefficients $N = \frac{n!}{n_1!n_2!...n_k!}$	$P(A \cup B) = P(A) + P(B) - P(A \cap B)$
Combination $\binom{n}{r} = \frac{n!}{r!(n-r)!}$	Bayes Theorem
Conditional Probability $P(A B) = \frac{P(A \cap B)}{P(B)}$	$P(A B) = \frac{P(A B)P(B)}{P(A)}$ w/ $P(A) > 0$ and $P(B) > 0$
Independent $P(A B) = P(A)$	Theorem of Total Probability if $0 < P(B) < 1$
$P(B A) = P(B)$	$P(B A) = \frac{P(A B)P(B)}{P(A B)P(B) + P(A B)P(B)}$
$P(A \cap B) = P(A)P(B)$	Discrete Random Variables Expectations and Variance
Probability Mass Function $P(X = x)$	$E[Y] = \sum_{y \in Y} yP(y)$   $V[Y] = E[(Y - \mu)^2]$   $\mu = E[Y]$
Binomial Distribution $P_x = \binom{n}{x} p^x q^{n-x}$	Standard Deviation of Y is $\sqrt{V[Y]}$
Geometric Distribution	Extra Formulas (Derived Geometric Distribution)
$P(X = x) = q^{x-1} p$	$P(X \leq n) = 1 - (1 - p)^n$
$E(Y) = \frac{1}{p}$ and $V(Y) = \frac{(1-p)}{p^2}$	$P(X < n) = 1 - (1 - p)^{n-1}$
Hypergeometric Distribution	$P(X \geq n) = (1 - p)^{n-1}$
$n_A = \binom{r}{y} \binom{N-r}{n-y}$	$P(X > n) = (1 - p)^n$
$\mu = E(Y) = \frac{nr}{N}$ and	Negative Binomial Probability Distribution
$\sigma^2 = V(Y) = n(\frac{r}{N})(\frac{N-r}{N})(\frac{N-n}{N-1})$	$p(y) = \binom{y-1}{r-1} p^r q^{y-r}$ and $\mu = \frac{r}{p}$ and $\sigma^2 = \frac{r(1-p)}{p^2}$
Poisson Distribution	Tchebysheff's Theorem $1 - \frac{1}{k^2}$
$p(y) = \frac{\lambda^y}{y!} e^{-\lambda}$ and $\mu = \lambda$ and $\sigma^2 = \lambda$	$k = \frac{\text{the "within" number}}{\text{the standard deviation}}$ and $k$ must be $> 1$

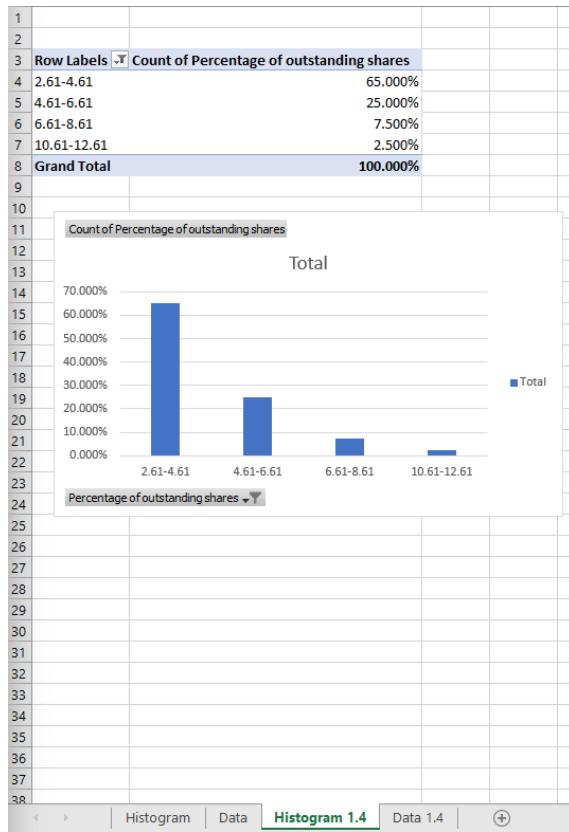
## Github

Git is a version control system that is considered the most common that people utilize. A user-friendly workflow is due to its flexibility in how its users can manage various changes, and there is no one way to interact with it or no standardized process. Git allows people to collaborate on projects and produce changes in a flow. Before teaming up, everyone should be on the same page and agree upon a workflow with several types available. It is recommended to use the Git workflow to accomplish the goal of productivity and consistency. The workflow helps to encourage and enhance the effectiveness of the team of developers and users. Before choosing any workflow, make sure it scales to the size of the team, that it is easy to undo mistakes and errors, and whether or not it creates unnecessary cognitive overheads to the team. To begin working with this workflow, users should understand how it functions. First, cloning the central repository will allow users to own a local copy of the project to edit files and commit changes, which will be stored locally. When ready to publish these changes to the official project, the user will have to push their local main branch to the central repository of the main project. The article "Comparing Git workflows: What you should know," by Atlassian, states that cloning will automatically create a shortcut, origin, pointing back to the parent repository. To make a change, the user, as mentioned, will have to commit it, editing, staging, and commit. The commit will be pushed to the central repository, and as the article states, once the local repository has these new changes committed, it will require a push to share with other developers on the project and to the central repository. This central repository is the official project with a history of commits. Sometimes, making changes to what anyone else may have done already will cause the flow to pause the rebasing process and let a developer of the team manually resolve any conflicts. If one commits their changes and pushes them to the origin, other teammates will solve the conflict, get their changes, and integrate with their local changes. Another developer can use git pull, which pulls the entire upstream commit history in their local repository to attempt integration with their local commits. Atlassian states that it is better to rebase so all commits go to the main branch tip, and if not, a merge commit will transfer each local commit to the updated main branch one at a time.

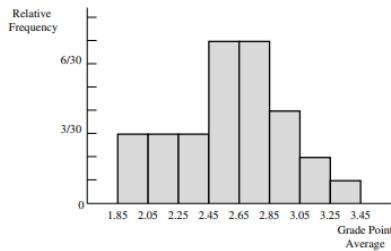
Git also allows collaborators to resolve other merge conflicts by locating where any problem is. To go over all the necessary terminology of git, start with git commit. A commit will take a staged snapshot and commit it to a project history. Combining this with git add, which is the basic workflow. Git merge is another way to integrate changes from divergence branches after the project history combines with the git branch and merge will put it all together again. Git pull is like git fetch but automated. It will download a branch from a remote repository and merge it into the current branch. Lastly is git push, the opposite of fetching, which allows one to move a local branch to another repository, allowing users to publish contributions. It will send a series of commits instead of a single changeset.

Dante Anzalone's Assignments and Program Screenshots  
 CSCI 3327: Probability and Applied Statistics

### Histogram



**1.5** Given here is the relative frequency histogram associated with grade point averages (GPAs) of a sample of 30 students:



- a Which of the GPA categories identified on the horizontal axis are associated with the largest proportion of students?
- b What proportion of students had GPAs in each of the categories that you identified?
- c What proportion of the students had GPAs less than 2.65?

- a → Categories 2.45-2.85
- b → 14 / 30 students
- c → 16 / 30 students

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Set Theory: Union, Intersection, and Complement

```
1 package statsOperations.Testers;
2 import java.util.ArrayList;
3 /**
4  * TestUIC Class - Creates three ArrayLists representing
5  * the two subsets, a and b, to test the UnionIntersection
6  * @author Dante Anzalone
7  * @version 2023-09 (4.29.0)
8  */
9 public class TestUIC
10 {
11     public static void main (String [] args)
12     {
13         UnionIntersectionComplement operating;
14         ArrayList<Integer> a, b, set; // Declaration
15         System.out.printf("_____\n");
16         set = new ArrayList <> (); // 1, 2, 3, 4,
17         set.add(0);
18         set.add(1);
19         set.add(2);
20         set.add(3);
21         set.add(4);
22         set.add(5);
23         set.add(6);
24         set.add(7);
25         set.add(8);
26         set.add(9);
27         set.add(10);
28         a = new ArrayList <> (); // 0, 0, 4, 4, 6,
29         a.add(0);
30         a.add(0);
31         a.add(4);
32         a.add(4);
33         a.add(6);
34         a.add(8);
35         a.add(10);
36         b = new ArrayList <> (); // 0, 1, 2, 4, 5,
37         b.add(0);
38         b.add(1);
39         b.add(2);
40         b.add(4);
41         b.add(5);
42         b.add(9);
43         b.add(10);
44         operating = new UnionIntersectionComplement();
45         System.out.println("Intersection of the two arrays: " + operating.intersection(a, b));
46         System.out.println("Union of the two arrays: " + operating.union(a, b));
47         System.out.println("The complement of arrayListOne: " + operating.complement(a));
48         System.out.println("The complement of arrayListTwo: " + operating.complement(b));
49     }
50 }
51 
```

Console X Git Staging

```
<terminated> TestUIC [Java Application] /Users/dante_anz/.p2/pool/plugins
Intersection of the two arrays: [0, 4, 10]
Union of the two arrays: [0, 1, 2, 4, 5, 6, 8, 9, 10]
The complement of arrayListOne: [1, 2, 3, 5, 7, 9]
The complement of arrayListTwo: [3, 6, 7, 8]
```

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Mean, Median, Mode, Standard Deviation, and Variance

The screenshot shows a Java development environment with two tabs open: `MeanMedianModeSD.java` and `TestMMMS.java`. The `TestMMMS.java` tab is active, displaying the following code:

```
2+ import java.util.ArrayList;
6*
7 * TestMMMS Class - Constructs an ArrayList
8 * which is to represent a set of elements
9 * operation is tested here.
10 *
11 * @author Dante Anzalone
12 * @version 2023-09 (4.29.0)
13 */
14 public class TestMMMS
15 {
16     public static void main (String [] args)
17     {
18         MeanMedianModeSD test; // 
19         ArrayList<Double> testNumbers; // 
20         double testerResults; // 
21
22         // Initializing the test Object and
23         test = new MeanMedianModeSD();
24         testNumbers = new ArrayList<Double>();
25
26         /*
27          * The list of elements inside the
28          * LIST OF ELEMENTS' VALUES MUST BE
29          * MODE WILL NOT CHECK IF THERE ARE
30          */
31         testNumbers.add(0.8);
32         testNumbers.add(0.8);
33         testNumbers.add(3.3);
34         testNumbers.add(3.9);
35         testNumbers.add(5.4);
36         testNumbers.add(5.5);
37         testNumbers.add(7.0);
38         testNumbers.add(15.0);
39
40         // Testing each method contained in
41         testerResults = test.getMean(testN
42         System.out.printf("Mean - %5.2f%
43         testerResults = test.getMedian(tes
44         System.out.printf("Median - %5.2f%
45
46         System.out.printf("Mode - %s%n"
47         testerResults = test.getStandardDe
48         System.out.printf("SD - %5.2f%
49         testerResults = test.getVariance(t
50         System.out.printf("Variance - %5.2%
```

The console output window below shows the results of the tests:

```
Mean - 5.21
Median - 4.65
Mode - 0.80
SD - 4.23
Variance - 17.93
```

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Permutations, Combinations, and Factorial

The screenshot shows the Eclipse IDE interface with two open files: `CombinationPermutationFactorial.java` and `TestCPF.java`. The `TestCPF.java` file contains the following code:

```
1 package statsOperations.Testers;
2 import statsOperations.CombinationPermutationFactorial;
3 /**
4  * TestCPF - Used to test the CombinationPermutationFactorial class and its o
5  * @author Dante Anzalone
6  * @version 2023-09 (4.29.0)
7  */
8 public class TestCPF
9 {
10    public static void main (String [] args)
11    {
12        CombinationPermutationFactorial solve;
13        solve = new CombinationPermutationFactorial ();
14
15        System.out.printf("Factorial : %s%n", solve.getFactorial(15));
16        System.out.printf("Permutation : %s%n", solve.getPermutation(30, 3));
17        System.out.printf("Combination : %s%n", solve.getCombination(5, 2));
18    }
20 }
```

The `Console` tab at the bottom shows the output of the application:

```
<terminated> TestCPF [Java Application] /Users/dante_anz/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.
Factorial : 1307674368000
Permutation : 24360
Combination : 10
```

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Determining Independence, and Conditional and Bayes

The screenshot shows an IDE interface with two tabs: "ConditionalBayesDependence.java" and "TestCBD.java". The code in "ConditionalBayesDependence.java" contains logic for calculating probabilities of intersections and unions of sets A, B, and C, and determining if they are mutually exclusive. It also includes Bayes Theorem calculations. The code is annotated with line numbers from 63 to 107. The "TestCBD.java" tab is visible in the background.

```
63     System.out.printf("Finding probability intersection of A and B : %.2f%n", (solve.probIntersection(a, b, totalObjects)));
64     System.out.printf("Finding probability intersection of A and C : %.2f%n", (solve.probIntersection(a, c, totalObjects)));
65
66     a.clear();
67     b.clear();
68     c.clear();
69     /*=====
70
71     // For Bayes-Theorem problem
72
73     a.add(2); // Die containing numbers 2, 4, and 5
74     a.add(4);
75     a.add(6);
76
77     b.add(5); // Die containing numbers 5 and 6
78     b.add(6);
79
80     totalObjects = 6;
81     System.out.printf("Bayes Theorem result: %.2f%%n", solve.bayesTheorem(a, b, totalObjects) * 100);
82     a.clear();
83     b.clear();
84     /*=====
85
86     // Testing if two sets are mutually or not mutually exclusive
87
88     a.add(2);
89     a.add(4);
90     a.add(8);
91
92     b.add(1);
93     b.add(3);
94     b.add(5);
95
96     c.add(2);
97     c.add(3);
98
99     totalObjects = 6;
100    System.out.printf("Is the sets A and B mutually exclusive? : %s%n", solve.isMutuallyExclusive(a, b, totalObjects));
101    System.out.printf("Is the sets A and C mutually exclusive? : %s%n", solve.isMutuallyExclusive(a, c, totalObjects));
102
103   System.out.printf("Finding probability union of A and B : %.2f%n", (solve.findProbUnion(a, b, totalObjects)));
104   System.out.printf("Finding probability union of A and C : %.2f%n", (solve.findProbUnion(a, c, totalObjects)));
105
106  /*=====
```

Console output:

```
<terminated> TestCBD [Java Application] /Users/dante_anz/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.8.v20230831-1047/jre/bin/java (Oct 24, 2023, 1
The probability of A given B is: 33.33%
Is the sets A and B independent? : false
Is the sets A and C independent? : true
Finding probability intersection of A and B : 0.00
Finding probability intersection of A and C : 0.17
Bayes Theorem result: 33.33%
Is the sets A and B mutually exclusive? : true
Is the sets A and C mutually exclusive? : false
Finding probability union of A and B : 1.00
Finding probability union of A and C : 0.67
```

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Distributions

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows two files: `Distributions.java` and `TestDistributions.java`.
- Code Editor:** Displays the content of `TestDistributions.java`. The code tests various probability distributions (Hypergeometric, Negative Binomial, Binomial, Geometric, Poisson, and Tchebycheff's Theorem) and prints their distribution, expectation, and variance.
- Console Output:** Shows the terminal output of the application. It includes:
  - Hypergeometric Distribution: Distribution 1.35%, Expectence 1.50, Variance 0.83
  - Negative Binomial Distribution: Distribution 3.07%, Expectence 15.00, Variance 60.00
  - Binomial Distribution: Distribution 20.13%
  - Geometric Distribution: Distribution 3.13%, Expectence 2.00, Variance 2.00
  - Poisson Distribution: Distribution 0.37%, Expectence 1.00, Variance 1.00
  - Tchebycheff's Theorem result: 75.00%
- Git Staging:** Shows the status of the Git repository.

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Birthday Checker Program

The screenshot shows the Eclipse IDE interface with the following details:

- Open Files:** Birthday.java, Person.java, Main.java (active tab).
- Code Editor Content:**

```
1 package birthday;
2 /**
3  * Main Class - Runs the class Birthday
4  * @author Dante
5  * @version 2023-09 (4.29.0)
6  */
7 public class Main
8 {
9     public static void main (String [] args)
10    {
11        Birthday days;
12        days = new Birthday();
13        days.run();
14    }
15 }
```
- Console Output:**

```
<terminated> Main [Java Application] /Users/dante_anz/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.8.v2
How many people are in your class?
30
12 : 12
12 : 10
8 : 13
5 : 4
1 : 24
12 : 25
6 : 6
6 : 6
9 : 23
3 : 20
6 : 10
1 : 10
9 : 24
12 : 29
4 : 29
9 : 30
7 : 10
8 : 18
9 : 8
8 : 15
12 : 24
9 : 20
5 : 18
2 : 9
1 : 29
5 : 6
4 : 23
9 : 28
1 : 29
11 : 27
2.0
The probability that at least two people share the same birthday in class is : 0.06666666666666667
```

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Monte's 3 Doors / Deal or No Deal Program

The screenshot shows the Eclipse IDE interface. In the top-left, there are two tabs: "PlayGame.java" and "ThreeDoors.java". The "PlayGame.java" tab is active, displaying the following code:

```
1 //  
2 * PlayGame Class - Runs the ThreeDoors class  
3 * @author Dante Anzalone  
4 * @version 2023-09 (4.29.0)  
5 */  
6 public class PlayGame  
7 {  
8     public static void main (String [] args)  
9     {  
10         ThreeDoors game = new ThreeDoors (10000);  
11         game.constructAndPlayGame();  
12     }  
13 }  
14
```

In the bottom-right corner of the code editor, there is a small terminal window showing the output of the Java application:

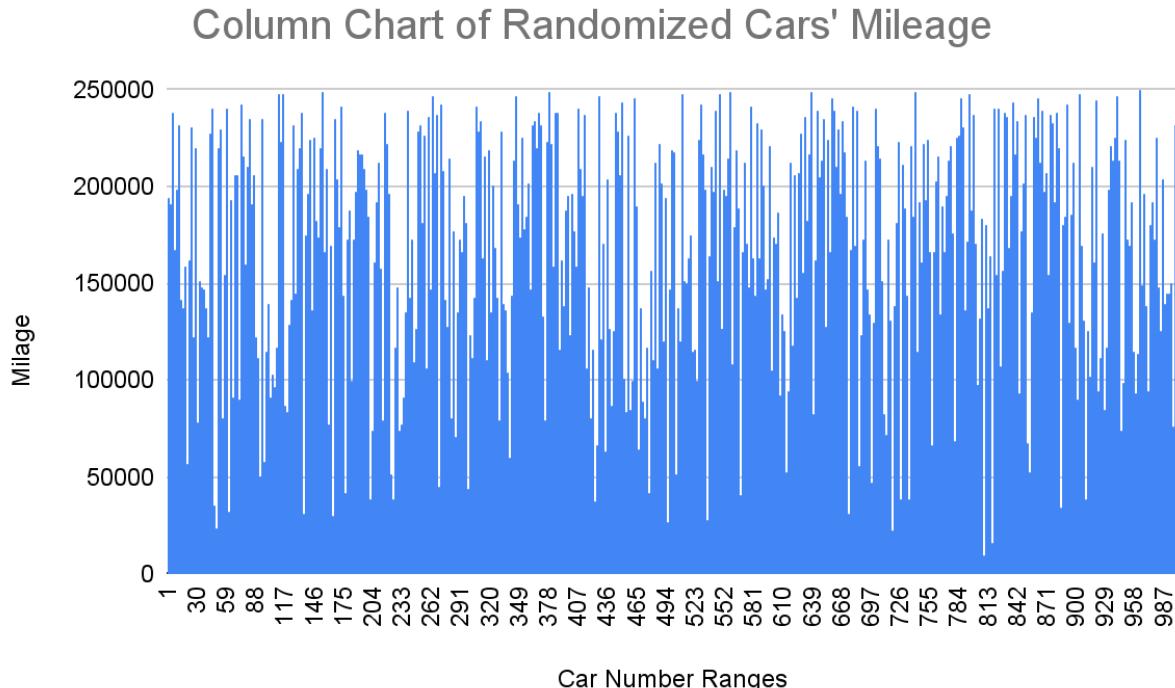
```
<terminated> PlayGame [Java Application] /Users/dante_anz/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.8.v20230831-1047/jre/bin/java (Oct 23, 2023, 5:20:37 PM - 5:20:37 PM) [pid: 7986]  
winsChangeDoor = Win probability of 66.22 that the host removes one incorrect door and the player changes their initial pick to a new door, out of the remaining doors  
winsKeepDoor = Win probability of 32.85 that player keeps the same door after getting the chance to change, when the hosts removes an incorrect door, and not taking it
```

The terminal window has tabs for "Console" and "Git Staging". At the bottom of the terminal window, there are status indicators: "Writable", "Smart Insert", and the time "3:13:62".

# Dante Anzalone's Assignments and Program Screenshots

## CSCI 3327: Probability and Applied Statistics

## Car Factory Program



The screenshot shows the Eclipse IDE interface with the following details:

- Java Editor:** The main window displays the `TestFactory.java` file. The code implements a factory pattern to produce different types of cars (Sports Car, Hatchback, Sedan, Van, Minivan, Truck) based on their color and model number.
- Terminal Output:** The bottom part of the interface shows the terminal window with the following output:

```
terminated> TestFactory (1) [Java Application] /Users/dante_anz/Downloads/p2pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.8.v20230831-1047/jre/bin/java (Oct 30, 2023, 8:00:49 PM - 8:00:50 PM) [pid: 91701]
Sports Car, 1, Pink, 49949
Hatchback, 22, Green, 49949
Convertible, 45, Red, 76835
Hatchback, 21, Blue, 200776
Sedan, 14, Red, 64152
Sedan, 29, Grey, 59939
Van, 51, Grey, 172517
Minivan, 45, Red, 46446
Truck, 6, Red, 154101
Convertible, 41, Black, 131666
Convertible, 42, Red, 60075
Sports Car, 12, Green, 64314
Convertible, 35, Yellow, 76693
Van, 49, Blue, 58499
Van, 31, Pink, 57885
```

Dante Anzalone's Assignments and Program Screenshots  
CSCI 3327: Probability and Applied Statistics

Monte Carlo Simulation (Hands)

The screenshot shows the Eclipse IDE interface with two Java files open: HandEvaluator.java and PlayGame.java. The HandEvaluator.java file contains code for a Monte Carlo simulation of poker hands. The PlayGame.java file contains the main method and other utility functions. Below the editor is a terminal window showing the output of the PlayGame application, which calculates the probability of various poker hands from 10,000 simulations.

```
HandEvaluator.java
PlayGame.java
SortingAlgorithms.java
Card.java

173     }
174     clearHand();
175     drawHand(handSize);
176     sortHand();
177     result = royalFlush();
178     if (result == true)
179     {
180         royalFlush++;
181     }
182     clearHand();
183
184
185     runAmount--;
186 }
187 probability = pairCount / hands;
188 System.out.printf("The probability of getting a pair in %s hands is %s\n", hands, probability);
189 probability = threeKind / hands;
190 System.out.printf("The probability of getting a three kind in %s hands is %s\n", hands, probability);
191 probability = fourKind / hands;
192 System.out.printf("The probability of getting a four kind in %s hands is %s\n", hands, probability);
193 probability = containStraight / hands;
194 System.out.printf("The probability of getting a straight in %s hands is %s\n", hands, probability);
195 probability = containsFlush / hands;
196 System.out.printf("The probability of getting a flush in %s hands is %s\n", hands, probability);
197 probability = fullHouse / hands;
198 System.out.printf("The probability of getting a full house in %s hands is %s\n", hands, probability);
199 probability = straightFlush / hands;
200 System.out.printf("The probability of getting a straight Flush in %s hands is %s\n", hands, probability);
201 probability = royalFlush / hands;
202 System.out.printf("The probability of getting a royal Flush in %s hands is %s\n", hands, probability);
203
204
205 }
206 /**
207  * clearHand Method - Clears the current hand for a new shuffle
208 */
209 public void clearHand() {
}

Console X Problems Debug Shell Git Repositories Git Staging
terminated> PlayGame () [Java Application] /Users/dante/Downloads/poker/src/org/eclipse/jdt/internal/openjdk/hotspot/jre/bin/java (Oct 30, 2023, 11:18:40 PM - 11:18:43 PM) [pid: 9335]
The probability of getting a pair in 10000 hands is 0.8637
The probability of getting a three kind in 10000 hands is 0.1628
The probability of getting a four kind in 10000 hands is 0.0115
The probability of getting a straight in 10000 hands is 0.0844
The probability of getting a flush in 10000 hands is 0.051
The probability of getting a full house in 10000 hands is 0.1597
The probability of getting a straight Flush in 10000 hands is 0.0037
The probability of getting a royal Flush in 10000 hands is 0.0
```