

Mean:

```

1
2 public class Mean {
3
4     private int[] data = {1, 2, 3, 4, 5, 7, 8, 9, 32};
5
6
7     public double calculateMean() {
8         int sum = 0;
9         for (int value : data) {
10             sum += value;
11         }
12         return (double) sum / data.length;
13     }
14
15 }
16

```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe -Djava.library.path=C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\plugin2 -Xms128m -Xmx1024m -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djna.noincremental=true -jar C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\plugin2\lib\plugin2.jar

Mean: 7.888888888888889

Variance:

```
Tester.java × Mean.java *Variance.java × StandardDeviation.java
1
2 public class Variance {
3     private int[] data = {1, 2, 3, 4, 5, 6, 7, 8, 11, 21};
4
5
6     private double mean() {
7         int sum = 0;
8         for (int value : data) {
9             sum += value;
10        }
11        return (double) sum / data.length;
12    }
13
14
15    public double calculateVariance() {
16        double mean = mean();
17        double sumOfSquares = 0;
18
19        for (int value : data) {
20            sumOfSquares += Math.pow(value - mean, 2);
21        }
22        return sumOfSquares / (data.length - 1);
23    }
24
25 }
26
```

Console ×

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\p2\pool\plugins\org.eclipse.justi.openjdk.hot

Sample Variance: 33.733333333333334

Standard Deviation

```
1
2 public class StandardDeviation {
3
4     private int[] data = {1, 2, 3, 4, 5, 6, 7, 8};
5
6     private double mean() {
7         int sum = 0;
8         for (int value : data) {
9             sum += value;
10        }
11        return (double) sum / data.length;
12    }
13    private double variance() {
14        double mean = mean();
15        double sumOfSquares = 0;
16
17        for (int value : data) {
18            sumOfSquares += Math.pow(value - mean, 2);
19        }
20
21        return sumOfSquares / (data.length - 1);
22    }
23    public double calculateStandardDeviation() {
24        double variance = variance();
25        return Math.sqrt(variance);
26    }
27 }
28
29
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openj

Standard Deviation: 2.449489742783178

Probability:

```
1
2 public class Probability {
3
4     private double[] probabilities = {0.1, 0.2, 0.05, 0.15, 1};
5
6     public double calculateProbability() {
7         double sum = 0;
8         for (double p : probabilities) {
9             sum += p;
10        }
11        return sum;
12    }
13
14 }
15
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot

Union Probability: 1.5

Permutations:

```

1
2 public class Permutations {
3
4     private int n = 5;
5     private int r = 3;
6
7     private int factorial(int num) {
8         int result = 1;
9         for (int i = 2; i <= num; i++) {
10             result *= i;
11         }
12         return result;
13     }
14
15     public int calculatePermutation() {
16         return factorial(n) / factorial(n - r);
17     }
18 }
19
20

```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\

Permutation P(n, r) = 60

Combinations:

```

1
2 public class Combinations {
3
4     private int n = 5;
5     private int r = 3;
6
7     private int factorial(int num) {
8         int result = 1;
9         for (int i = 2; i <= num; i++) {
10             result *= i;
11         }
12         return result;
13     }
14
15     public int calculateCombination() {
16         return factorial(n) / (factorial(r) * factorial(n - r));
17     }
18
19
20 }
21

```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre

Combination C(n, r) = 10

Conditional Probability

```
1
2 public class ConditionalProbability {
3
4     private double pAIntersectionB = 0.2; //  $P(A \cap B)$ 
5     private double pB = 0.5;
6
7     public double calculateConditionalProbability() {
8         if (pB == 0) {
9             return 0;
10        } else {
11            return pAIntersectionB / pB;
12        }
13    }
14
15    public String getResult() {
16        double conditionalProbability = calculateConditionalProbability();
17        return "P(A | B) = " + conditionalProbability;
18    }
19 }
20
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
P(A | B) = 0.4

Independent Multiplication:

```
1
2 public class MultiplicationIndependent {
3
4     private double pA = 0.5;
5     private double pB = 0.4;
6
7     public double calculateIntersection() {
8         return pA * pB;
9     }
10
11    public String getResult() {
12        double intersectionProbability = calculateIntersection();
13        return "Independent Multiplication P(A ∩ B) = " + intersectionProbability;
14    }
15
16 }
17
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
Independent Multiplication P(A ∩ B) = 0.2

Total Probability:

```
1
2 public class TotalProbability {
3
4     private double[] pAIB = {0.9, 0.5, 0.2}; // P(A | Bi)
5     private double[] pBi = {0.3, 0.4, 0.3};
6
7     public double calculateTotalProbability() {
8         double totalProbability = 0;
9
10        for (int i = 0; i < pAIB.length; i++) {
11            totalProbability += pAIB[i] * pBi[i];
12        }
13        return totalProbability;
14    }
15    public String getResult() {
16        return "Total Probability P(A) = " + calculateTotalProbability();
17    }
18 }
19 }
20
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe -Djava.library.path=C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe -jar C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe
Independent Multiplication $P(A \cap B) = 0.2$

Expected Value:

```
1
2 public class ExpectedValue {
3
4     private int[] values = {1, 2, 3, 4};
5     private double[] probabilities = {0.1, 0.3, 0.4, 0.2};
6
7
8     public double calculateExpectedValue() {
9         double expectedValue = 0;
10
11        for (int i = 0; i < values.length; i++) {
12            expectedValue += values[i] * probabilities[i];
13        }
14        return expectedValue;
15    }
16
17    public String getResult() {
18        return "E(Y) = " + calculateExpectedValue();
19    }
20 }
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe -Djava.library.path=C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe -jar C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe
E(Y) = 2.7

Binomial Distribution:

```
1
2 public class BinomialDistribution {
3
4     private int n = 5; // trials
5     private double p = 0.6; // probability of success
6     private int y = 3; //number of successes
7
8     // factorial for the combination method
9     private int factorial(int num) {
10         int result = 1;
11         for (int i = 2; i <= num; i++) {
12             result *= i;
13         }
14         return result;
15     }
16
17     private int calculateCombination(int n, int r) {
18         return factorial(n) / (factorial(r) * factorial(n - r));
19     }
20
21     // exponent method
22     private double power(double base, int exponent) {
23         double result = 1;
24         for (int i = 0; i < exponent; i++) {
25             result *= base;
26         }
27         return result;
28     }
29
30     public double calculateBinomialProbability() {
31         double q = 1 - p;
32         int combination = calculateCombination(n, y);
33         double successTerm = power(p, y);
34         double failureTerm = power(q, n - y);
35
36         return combination * successTerm * failureTerm;
37     }
38
39     public String getResult() {
40         return "P(Y = " + y + ") = " + calculateBinomialProbability();
41     }
42 }
43
44
```

Console X

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_11.0.22-b11\jre\bin\java.exe -Djava.library.path=C:\Users\Jaiden Nunez\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_11.0.22-b11\jre\bin\plugin2\bin -jar C:\Users\Jaiden Nunez\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_11.0.22-b11\jre\bin\plugin2\bin\java.exe
P(Y = 3) = 0.3456000000000001

Tester Class

```
1
2 public class Tester {
3
4     public static void main(String[] args) {
5
6         Mean calculator = new Mean();
7         System.out.println("Mean: " + calculator.calculateMean());
8
9         Variance calculator1 = new Variance();
10        System.out.println("Sample Variance: " + calculator1.calculateVariance());
11
12        StandardDeviation calculator2 = new StandardDeviation();
13        System.out.println("Standard Deviation: " + calculator2.calculateStandardDeviation());
14
15        Probability calculator3 = new Probability();
16        System.out.println("Union Probability: " + calculator3.calculateProbability());
17
18        Permutations calculator4 = new Permutations();
19        System.out.println("Permutation P(n, r) = " + calculator4.calculatePermutation());
20
21        Combinations calculator5 = new Combinations();
22        System.out.println("Combination C(n, r) = " + calculator5.calculateCombination());
23
24        ConditionalProbability calculator6 = new ConditionalProbability();
25        System.out.println(calculator6.getResult());
26
27        MultiplicationIndependent calculator7 = new MultiplicationIndependent();
28        System.out.println(calculator7.getResult());
29
30        TotalProbability calculator8 = new TotalProbability();
31        System.out.println(calculator8.getResult());
32
33        ExpectedValue calculator9 = new ExpectedValue();
34        System.out.println(calculator9.getResult());
35
36        BinomialDistribution calculator10 = new BinomialDistribution();
37        System.out.println(calculator10.getResult());
38
39    }
40 }
41
42
```