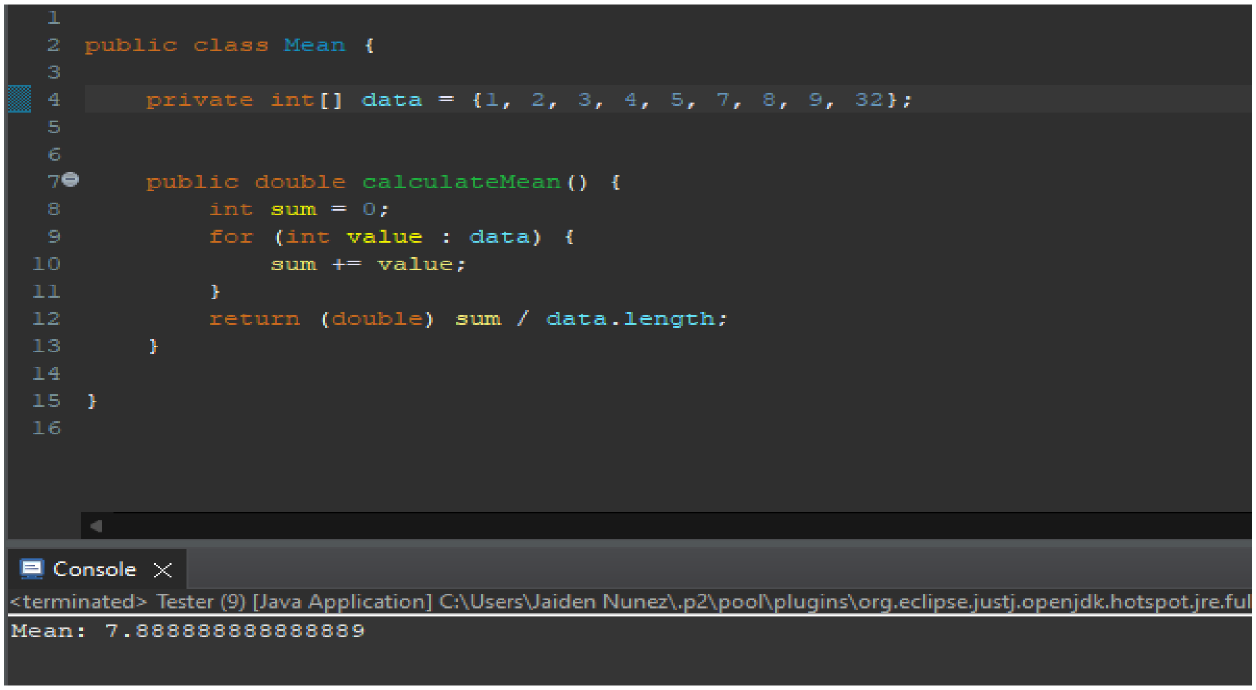
Mean:

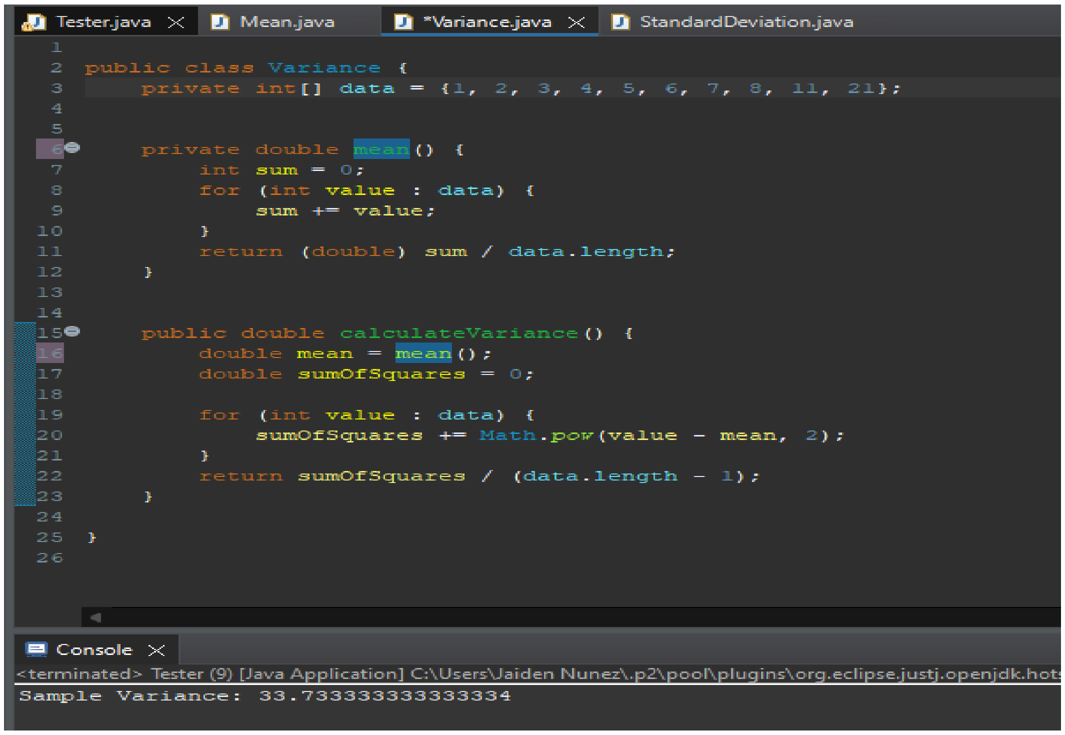


```java

public class Mean {

    private int[] data = {1, 2, 3, 4, 5, 7, 8, 9, 32};


    public double calculateMean() {
        int sum = 0;
        for (int value : data) {
            sum += value;
        }
        return (double) sum / data.length;
    }

}

```

Console

```
<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Mean: 7.888888888888889
```

Variance:



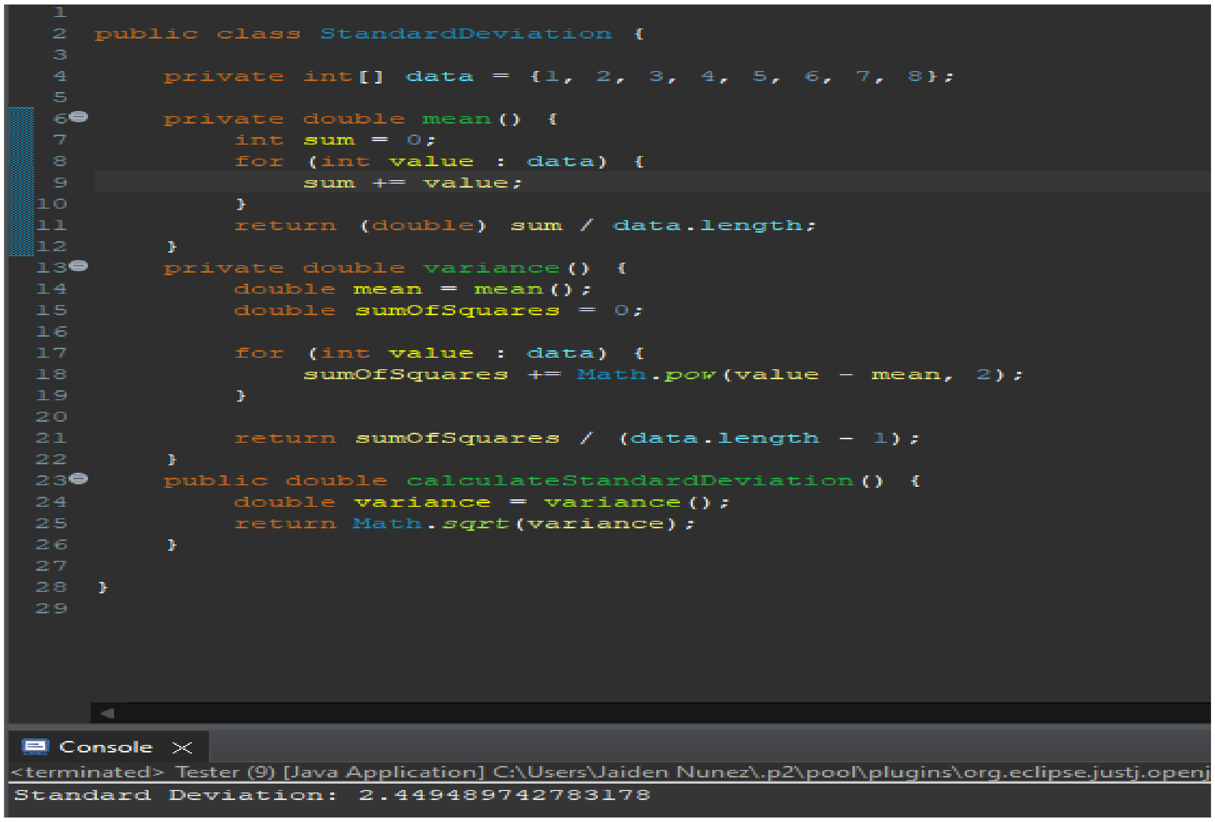
Tester.java   Mean.java   *Variance.java   StandardDeviation.java

```java

public class Variance {
    private int[] data = {1, 2, 3, 4, 5, 6, 7, 8, 11, 21};


    private double mean() {
        int sum = 0;
        for (int value : data) {
            sum += value;
        }
        return (double) sum / data.length;
    }


    public double calculateVariance() {
        double mean = mean();
        double sumOfSquares = 0;

        for (int value : data) {
            sumOfSquares += Math.pow(value - mean, 2);
        }
        return sumOfSquares / (data.length - 1);
    }

}

```

Console

```
<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hot
Sample Variance: 33.733333333333334
```

Standard Deviation

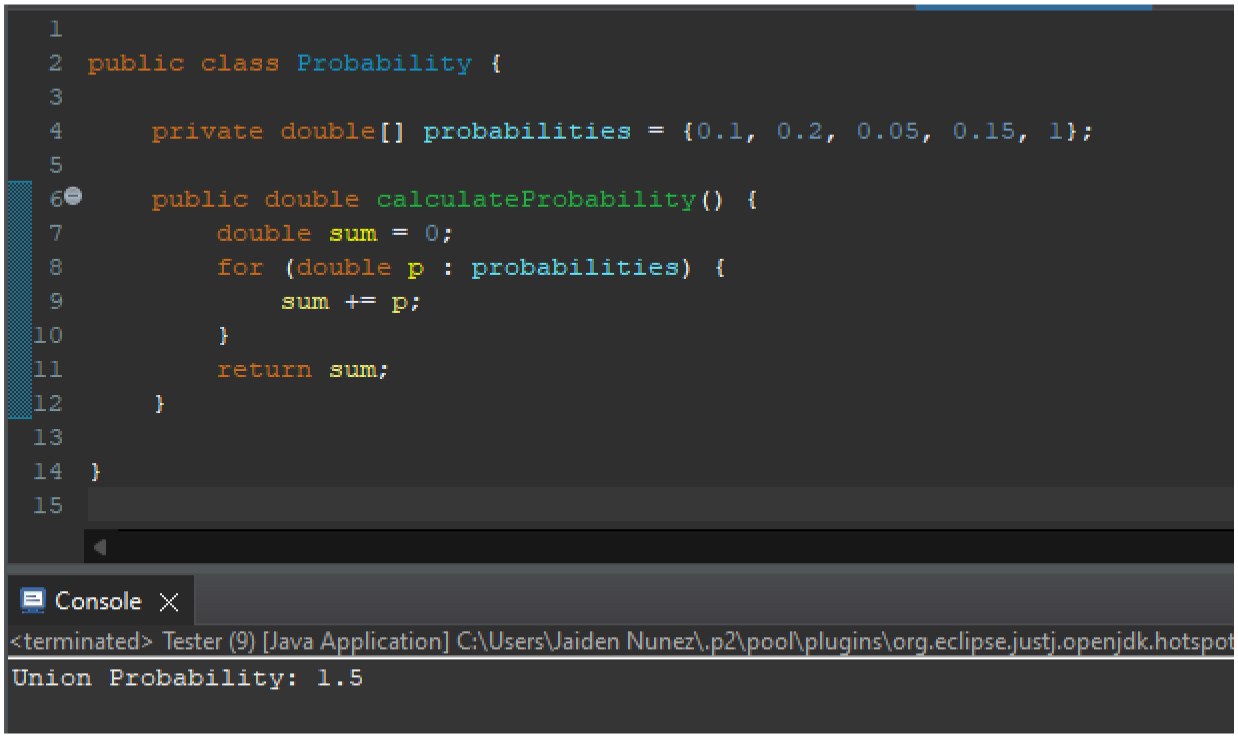


```java

public class StandardDeviation {

    private int[] data = {1, 2, 3, 4, 5, 6, 7, 8};

    private double mean() {
        int sum = 0;
        for (int value : data) {
            sum += value;
        }
        return (double) sum / data.length;
    }
    private double variance() {
        double mean = mean();
        double sumOfSquares = 0;

        for (int value : data) {
            sumOfSquares += Math.pow(value - mean, 2);
        }

        return sumOfSquares / (data.length - 1);
    }
    public double calculateStandardDeviation() {
        double variance = variance();
        return Math.sqrt(variance);
    }

}

```

Console

```
<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openj
Standard Deviation: 2.449489742783178
```

Probability:

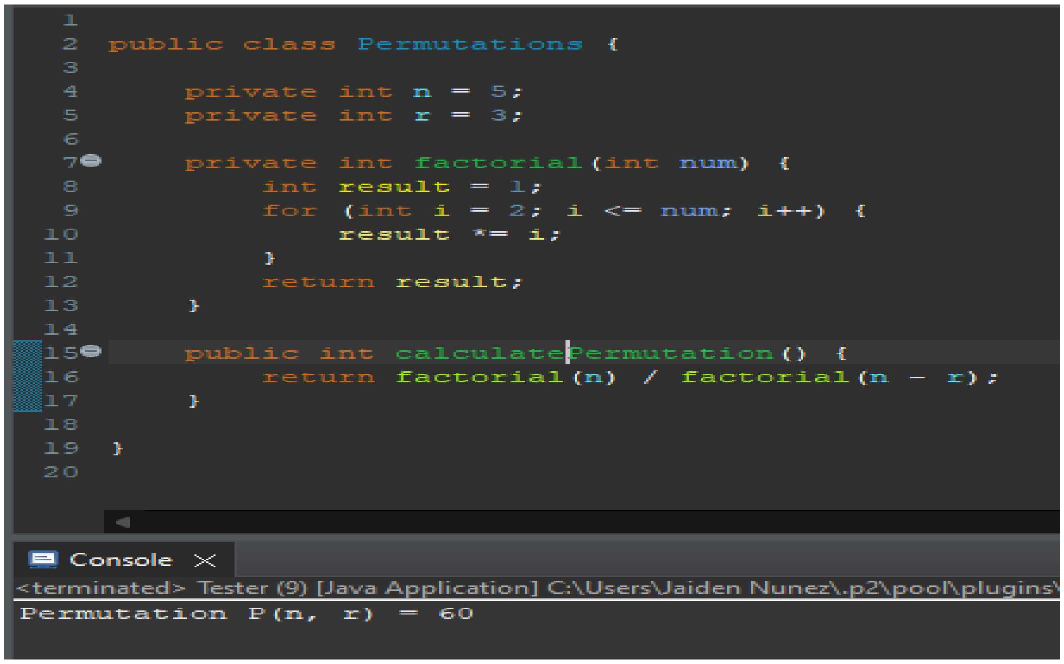


```java

public class Probability {

    private double[] probabilities = {0.1, 0.2, 0.05, 0.15, 1};

    public double calculateProbability() {
        double sum = 0;
        for (double p : probabilities) {
            sum += p;
        }
        return sum;
    }

}

```

Console

```
<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
Union Probability: 1.5
```

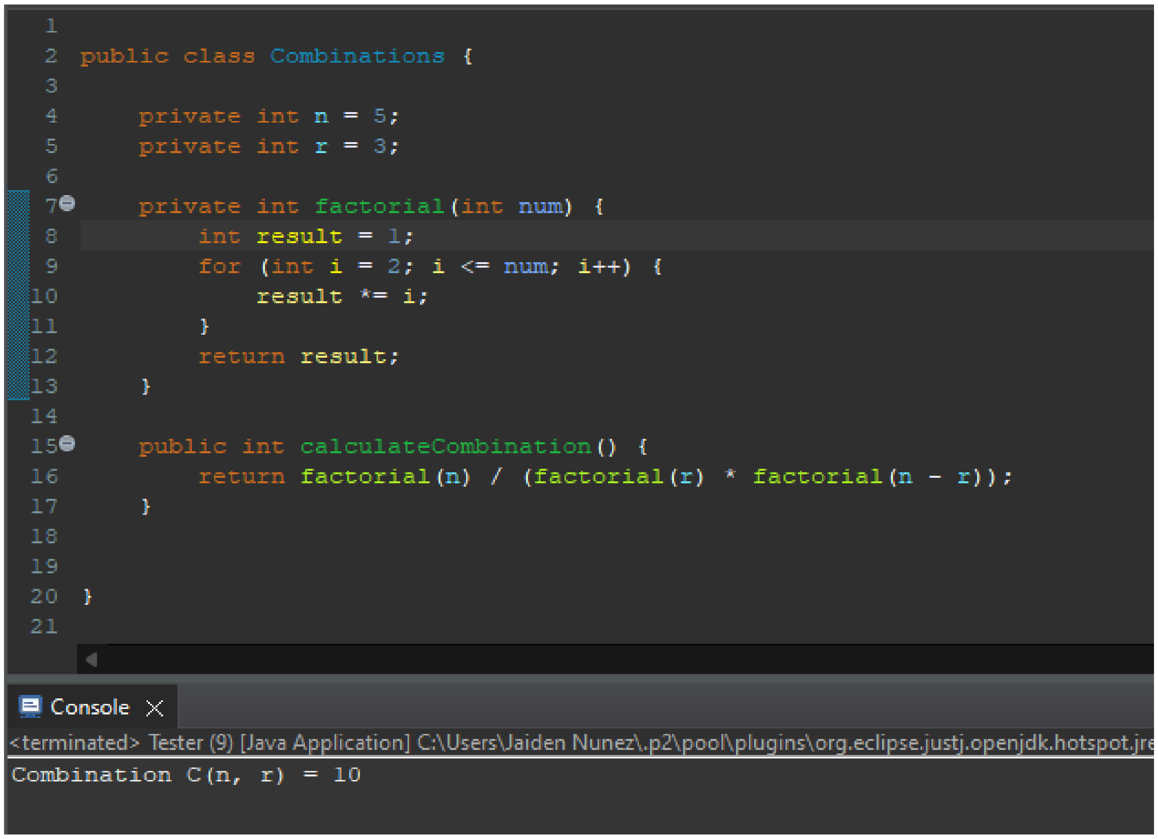
Permutations:



```
public class Permutations {

    private int n = 5;
    private int r = 3;

    private int factorial(int num) {
        int result = 1;
        for (int i = 2; i <= num; i++) {
            result *= i;
        }
        return result;
    }

    public int calculatePermutation() {
        return factorial(n) / factorial(n - r);
    }

}
```

```
<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\
Permutation P(n, r) = 60
```

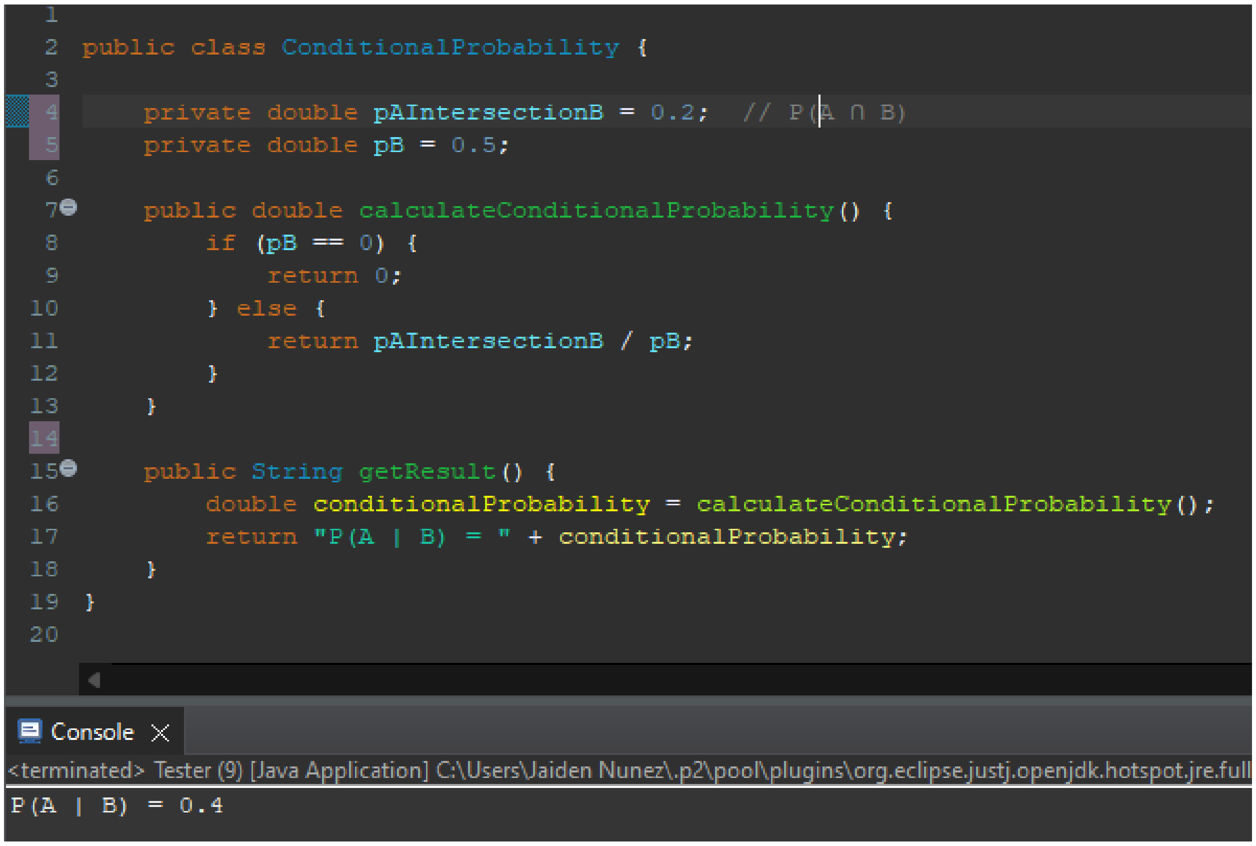
Combinations:



```
public class Combinations {

    private int n = 5;
    private int r = 3;

    private int factorial(int num) {
        int result = 1;
        for (int i = 2; i <= num; i++) {
            result *= i;
        }
        return result;
    }

    public int calculateCombination() {
        return factorial(n) / (factorial(r) * factorial(n - r));
    }


}
```

```
<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

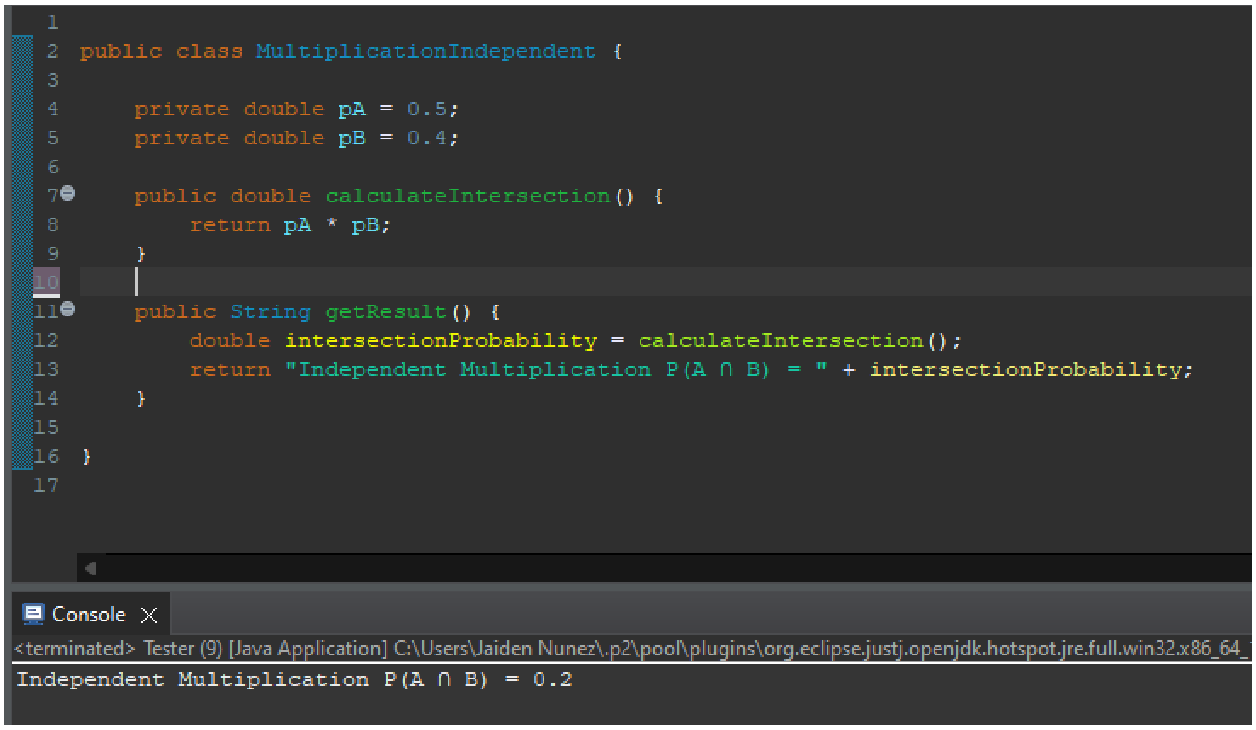


```java

public class ConditionalProbability {

    private double pAIntersectionB = 0.2;  // P(A ∩ B)
    private double pB = 0.5;

    public double calculateConditionalProbability() {
        if (pB == 0) {
            return 0;
        } else {
            return pAIntersectionB / pB;
        }
    }

    public String getResult() {
        double conditionalProbability = calculateConditionalProbability();
        return "P(A | B) = " + conditionalProbability;
    }
}

```

Console

```
<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:

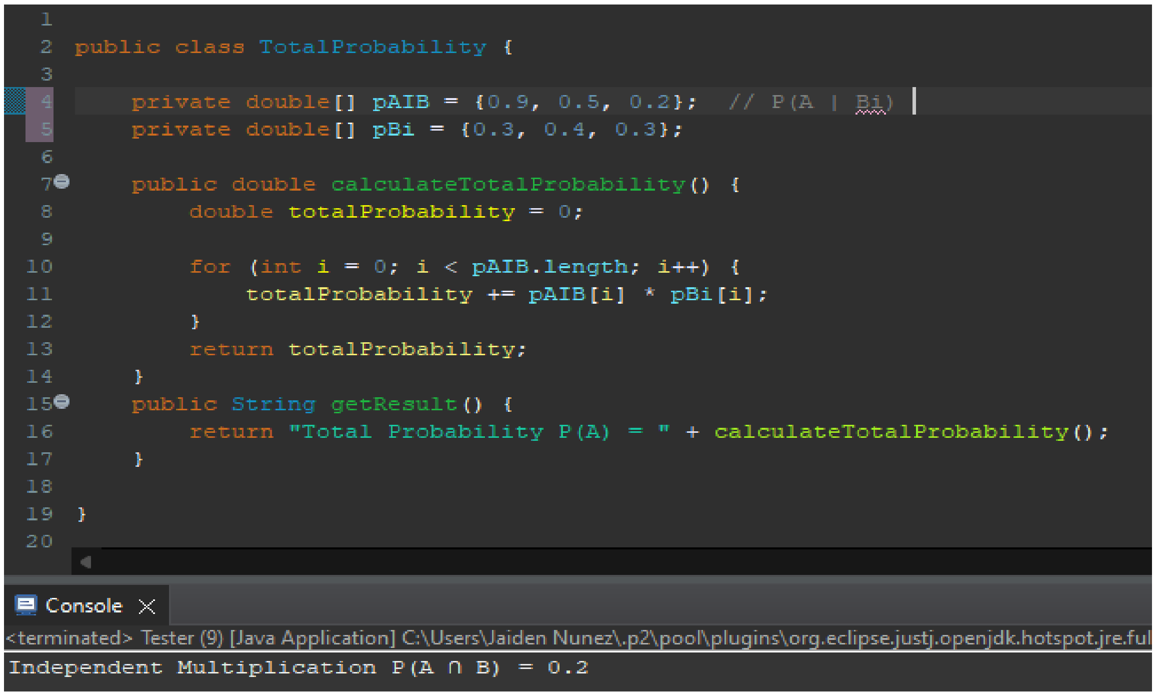


```java

public class MultiplicationIndependent {

    private double pA = 0.5;
    private double pB = 0.4;

    public double calculateIntersection() {
        return pA * pB;
    }

    public String getResult() {
        double intersectionProbability = calculateIntersection();
        return "Independent Multiplication P(A ∩ B) = " + intersectionProbability;
    }

}

```

Console

```
<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:

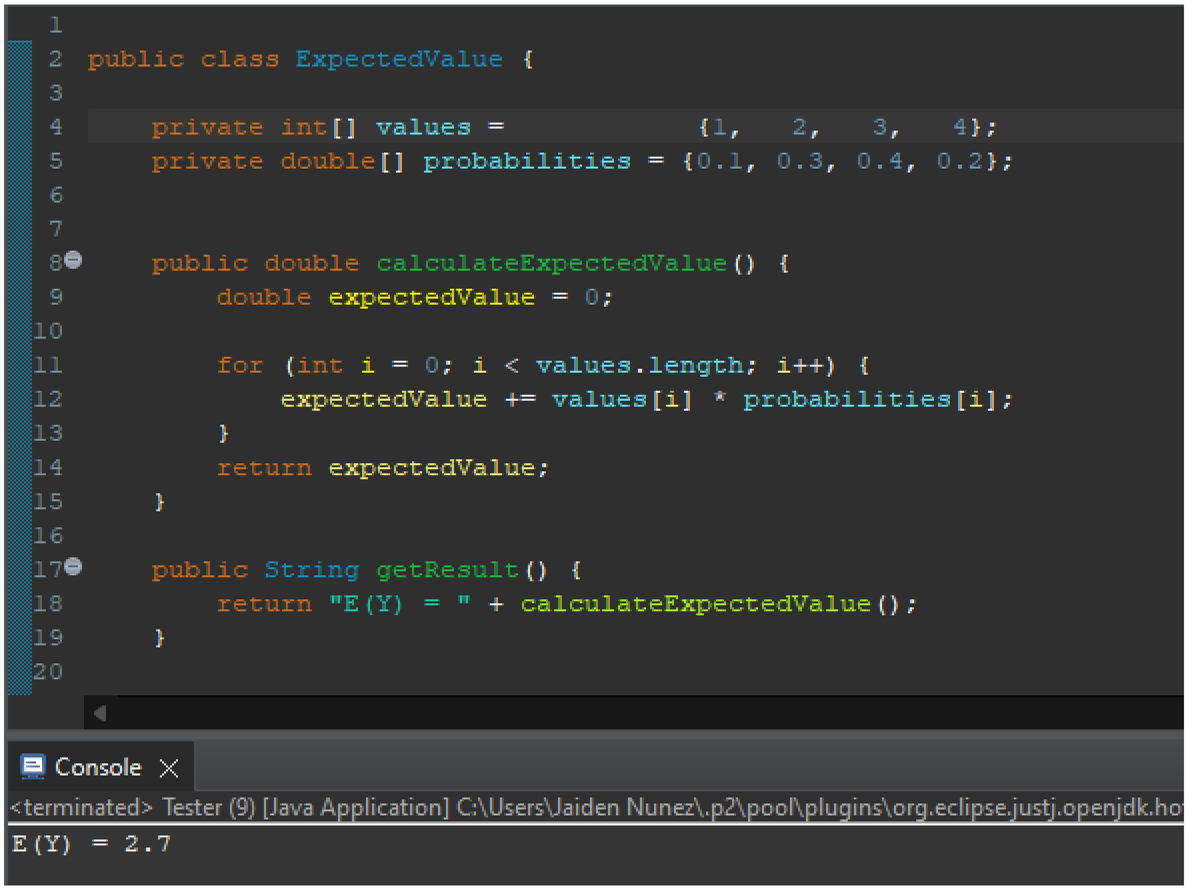


```java

public class TotalProbability {

    private double[] pAIB = {0.9, 0.5, 0.2};   // P(A | Bi)
    private double[] pBi = {0.3, 0.4, 0.3};

    public double calculateTotalProbability() {
        double totalProbability = 0;

        for (int i = 0; i < pAIB.length; i++) {
            totalProbability += pAIB[i] * pBi[i];
        }
        return totalProbability;
    }
    public String getResult() {
        return "Total Probability P(A) = " + calculateTotalProbability();
    }

}

```

Console

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

Expected Value:

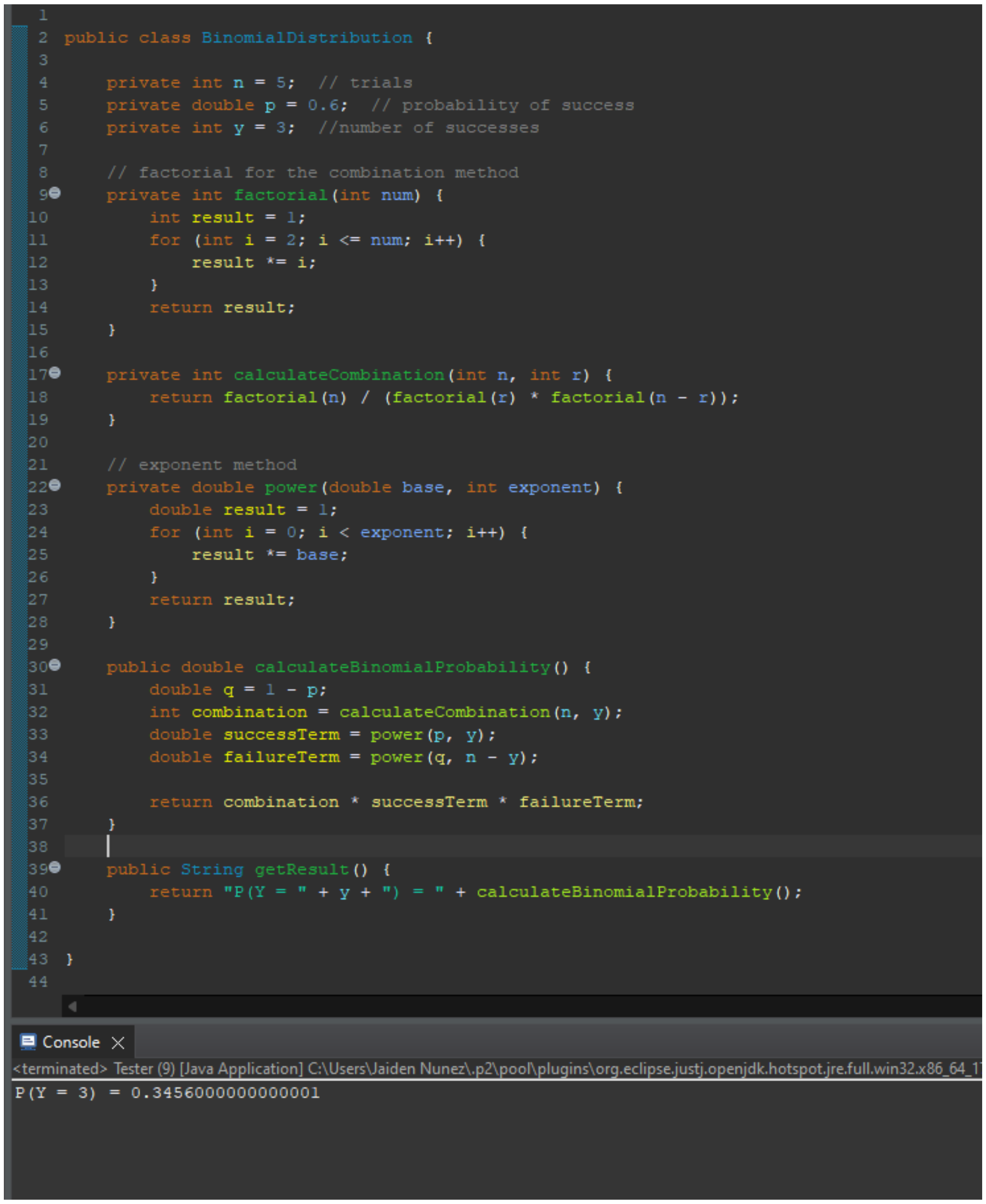


```java

public class ExpectedValue {

    private int[] values =              {1,   2,   3,   4};
    private double[] probabilities = {0.1, 0.3, 0.4, 0.2};


    public double calculateExpectedValue() {
        double expectedValue = 0;

        for (int i = 0; i < values.length; i++) {
            expectedValue += values[i] * probabilities[i];
        }
        return expectedValue;
    }

    public String getResult() {
        return "E(Y) = " + calculateExpectedValue();
    }

```

Console

```
<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.ho
E(Y) = 2.7
```

Binomial Distribution:



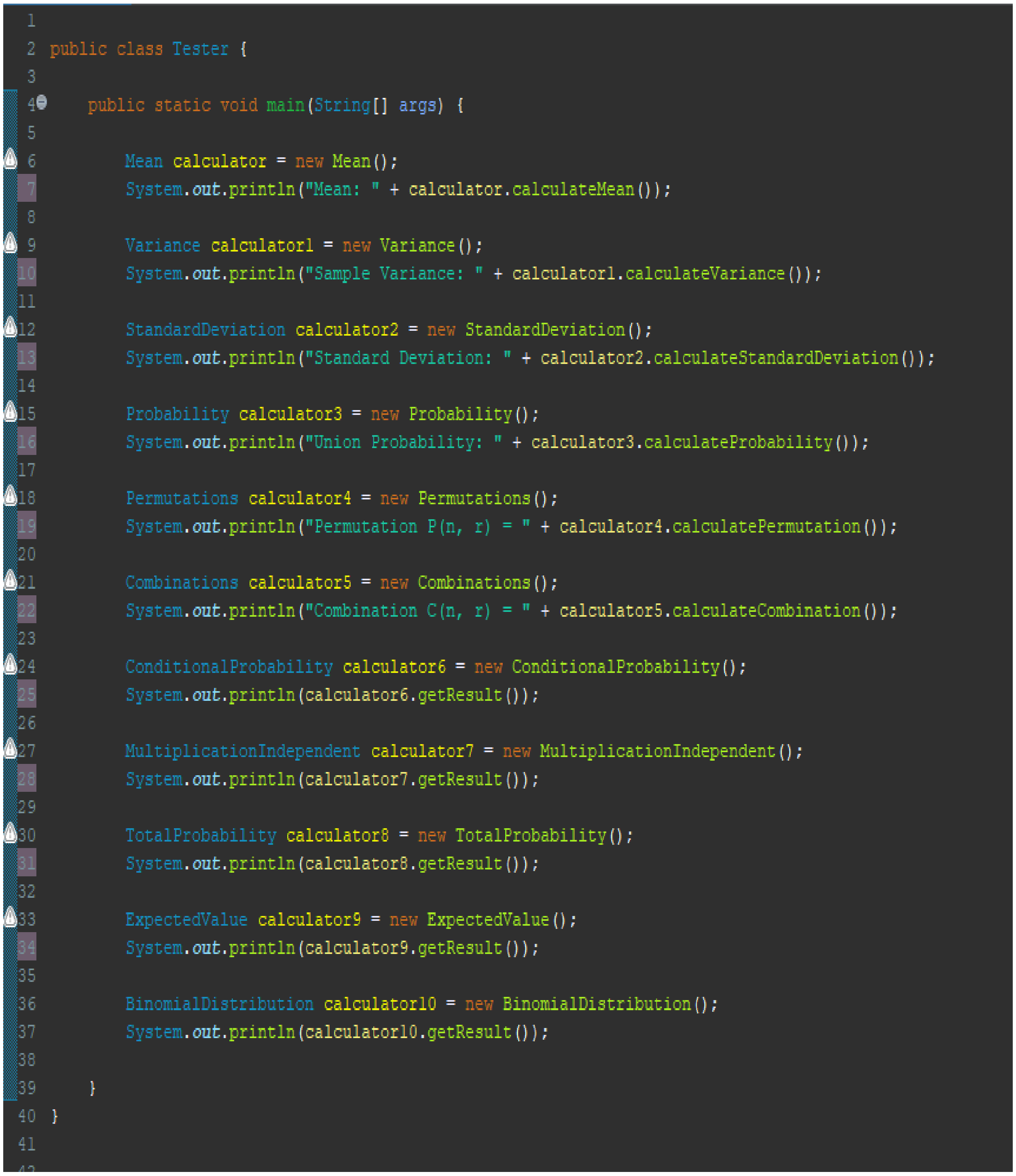
```java

public class BinomialDistribution {

    private int n = 5;  // trials
    private double p = 0.6;  // probability of success
    private int y = 3;  //number of successes

    // factorial for the combination method
    private int factorial(int num) {
        int result = 1;
        for (int i = 2; i <= num; i++) {
            result *= i;
        }
        return result;
    }

    private int calculateCombination(int n, int r) {
        return factorial(n) / (factorial(r) * factorial(n - r));
    }

    // exponent method
    private double power(double base, int exponent) {
        double result = 1;
        for (int i = 0; i < exponent; i++) {
            result *= base;
        }
        return result;
    }

    public double calculateBinomialProbability() {
        double q = 1 - p;
        int combination = calculateCombination(n, y);
        double successTerm = power(p, y);
        double failureTerm = power(q, n - y);

        return combination * successTerm * failureTerm;
    }

    public String getResult() {
        return "P(Y = " + y + ") = " + calculateBinomialProbability();
    }

}

```

Console

<terminated> Tester (9) [Java Application] C:\Users\Jaiden Nunez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1

```
P(Y = 3) = 0.3456000000000001
```

Tester Class



```java

public class Tester {

    public static void main(String[] args) {

        Mean calculator = new Mean();
        System.out.println("Mean: " + calculator.calculateMean());

        Variance calculator1 = new Variance();
        System.out.println("Sample Variance: " + calculator1.calculateVariance());

        StandardDeviation calculator2 = new StandardDeviation();
        System.out.println("Standard Deviation: " + calculator2.calculateStandardDeviation());

        Probability calculator3 = new Probability();
        System.out.println("Union Probability: " + calculator3.calculateProbability());

        Permutations calculator4 = new Permutations();
        System.out.println("Permutation P(n, r) = " + calculator4.calculatePermutation());

        Combinations calculator5 = new Combinations();
        System.out.println("Combination C(n, r) = " + calculator5.calculateCombination());

        ConditionalProbability calculator6 = new ConditionalProbability();
        System.out.println(calculator6.getResult());

        MultiplicationIndependent calculator7 = new MultiplicationIndependent();
        System.out.println(calculator7.getResult());

        TotalProbability calculator8 = new TotalProbability();
        System.out.println(calculator8.getResult());

        ExpectedValue calculator9 = new ExpectedValue();
        System.out.println(calculator9.getResult());

        BinomialDistribution calculator10 = new BinomialDistribution();
        System.out.println(calculator10.getResult());

    }
}

```