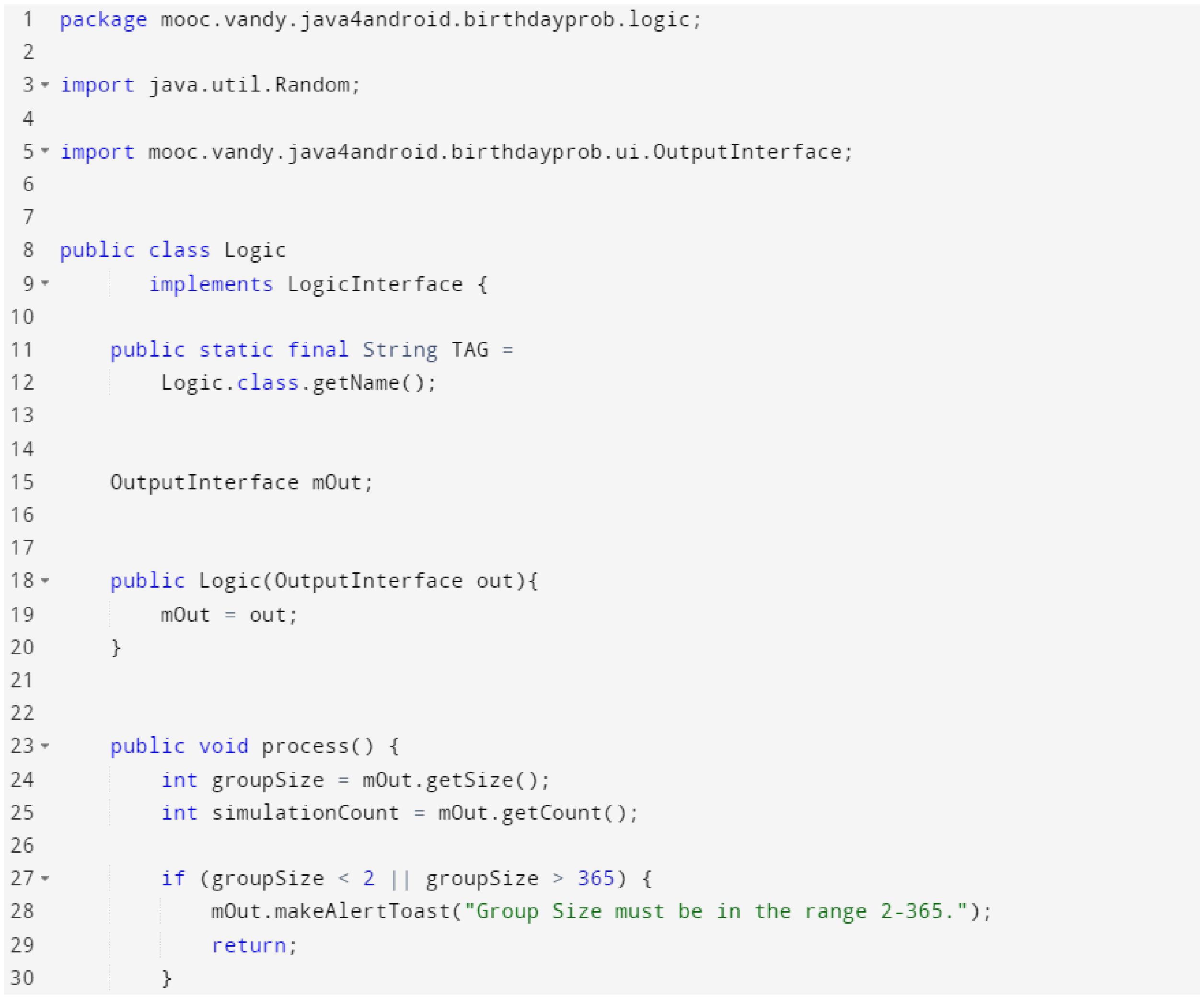
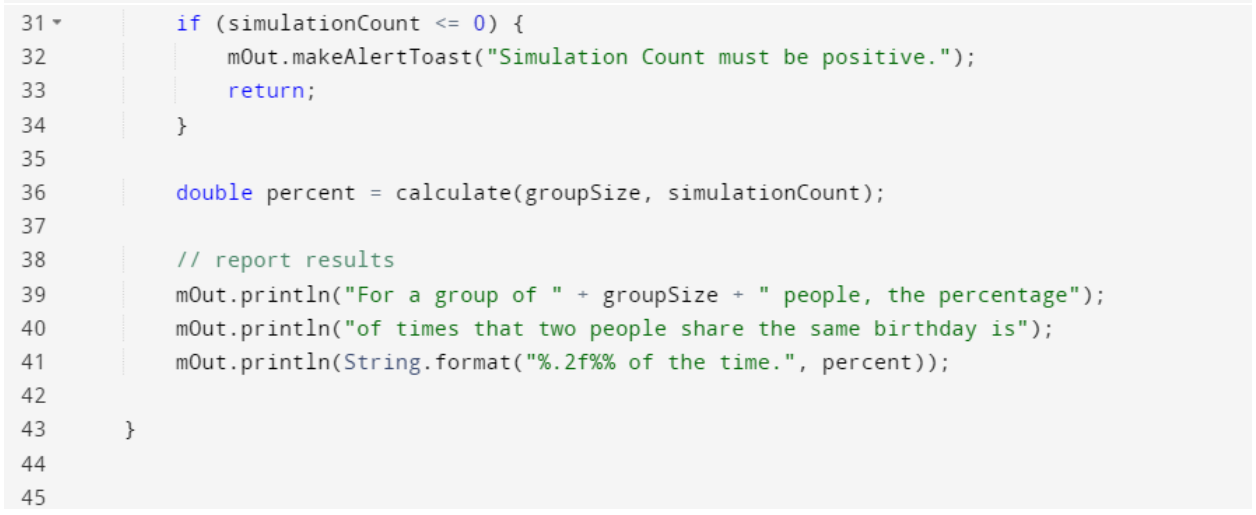
Introduction:

In probability theory, the birthday problem concerns the probability that, in a set of n randomly chosen people, some pair of them will have the same birthday. By the pigeonhole principle, the probability reaches 100% when the number of people reaches 366 (since there are 365 possible birthdays, excluding February 29th). We would need 183 people (half of 365) to reach a 50% probability. However, 99% probability is reached with just 57 people and 50% probability with just 23 people. These conclusions assume that each day of the year (except February 29) is equally probable for a birthday.

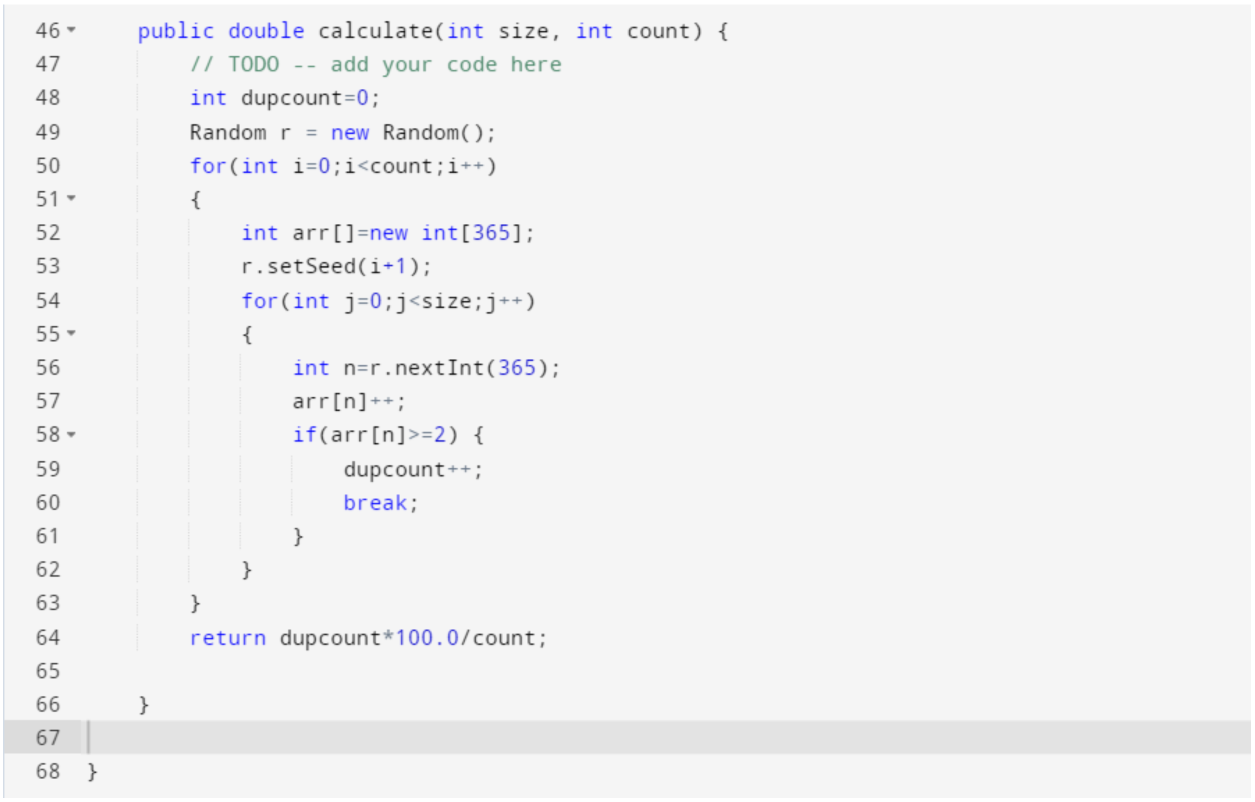
Given a group of 20 people and performing 10,000 simulation runs, your program should report that 41.24% of the time there were two people that shared the same birthday.



```java
package mooc.vandy.java4android.birthdayprob.logic;

import java.util.Random;

import mooc.vandy.java4android.birthdayprob.ui.OutputInterface;


public class Logic
        implements LogicInterface {

    public static final String TAG =
        Logic.class.getName();


    OutputInterface mOut;


    public Logic(OutputInterface out){
        mOut = out;
    }


    public void process() {
        int groupSize = mOut.getSize();
        int simulationCount = mOut.getCount();

        if (groupSize < 2 || groupSize > 365) {
            mOut.makeAlertToast("Group Size must be in the range 2-365.");
            return;
        }
```



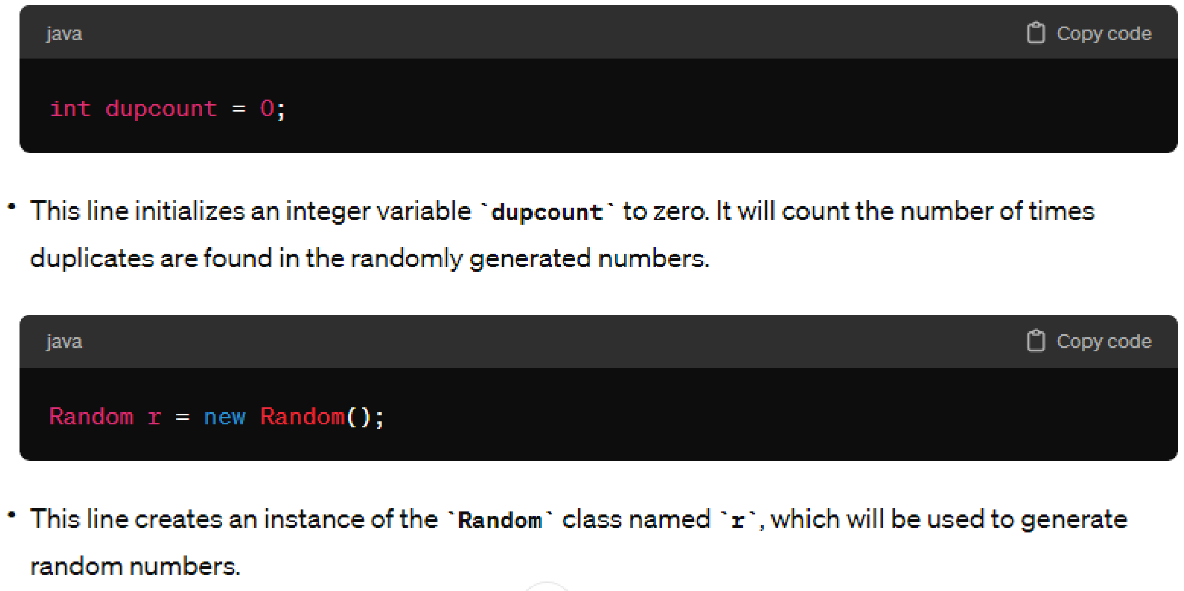
```java
        if (simulationCount <= 0) {
            mOut.makeAlertToast("Simulation Count must be positive.");
            return;
        }

        double percent = calculate(groupSize, simulationCount);

        // report results
        mOut.println("For a group of " + groupSize + " people, the percentage");
        mOut.println("of times that two people share the same birthday is");
        mOut.println(String.format("%.2f%% of the time.", percent));

    }


```



```java
    public double calculate(int size, int count) {
        // TODO -- add your code here
        int dupcount=0;
        Random r = new Random();
        for(int i=0;i<count;i++)
        {
            int arr[]=new int[365];
            r.setSeed(i+1);
            for(int j=0;j<size;j++)
            {
                int n=r.nextInt(365);
                arr[n]++;
                if(arr[n]>=2) {
                    dupcount++;
                    break;
                }
            }
        }
        return dupcount*100.0/count;

    }

}
```

The process function check for the correct group size after processing the group size it will do the calculation using the calculate function from the size and count value.

The calculate function process the algorithm.



```java
int dupcount = 0;
```

- This line initializes an integer variable `dupcount` to zero. It will count the number of times duplicates are found in the randomly generated numbers.

```java
Random r = new Random();
```

- This line creates an instance of the `Random` class named `r`, which will be used to generate random numbers.

```java
for (int i = 0; i < count; i++) {
```

- This line starts a loop that will run `count` times. This loop generates random numbers and checks for duplicates within a group of numbers of size `size`.

```java
int arr[] = new int[365];
```

- This line creates a new integer array named `arr` with a length of 365. This array will be used to track the occurrence of each random number within the group.

```java
r.setSeed(i + 1);
```

- This line sets the seed of the random number generator `r` to `i + 1`. It ensures that different sequences of random numbers are generated in each iteration of the loop.

java Copy code

```java
for (int j = 0; j < size; j++) {
```

- This line starts a nested loop that will run `size` times. Inside this loop, random numbers are generated and checked for duplicates within the group.

java Copy code

```java
int n = r.nextInt(365);
```

- This line generates a random integer `n` between 0 (inclusive) and 365 (exclusive).

java Copy code

```java
arr[n]++;
```

- This line increments the count of the element `n` in the `arr` array, indicating that the number `n` has been generated.

java Copy code

```java
if (arr[n] >= 2) {
    dupcount++;
    break;
}
```

- This block of code checks if the count of the element `n` in the `arr` array is greater than or equal to 2, indicating that a duplicate has been found within the same group of random numbers. If a duplicate is found, `dupcount` is incremented, and the loop is broken to move to the next iteration.

java Copy code

```java
return dupcount * 100.0 / count;
```

- This line calculates the percentage of sets where duplicates were found by multiplying `dupcount` by 100.0 and dividing by `count`. It then returns this percentage as a double value.