User Manual Birthday Program

By: Jacob Attia

## **Overview**

The Birthday Program estimates the likelihood that in a class of a given number of people, at least two individuals share the same birthday. Instead of handling actual dates ( "MM/DD"), the program simplifies the problem by representing birthdays as integers ranging from 1 to 365. This approach simplifies random generation and comparisons.

## **How It Works**

1. **User Input:**

* The program prompts the user to enter the number of people in the class.
* The user is also asked to specify how many simulation runs should be performed.

1. **Random Birthday Generator:**

* For each simulation run, a list of Person objects is created.
* Each Person is assigned a random birthday (an integer between 1 and 365). This approach avoids complexities with date formatting and leap years.

1. **Duplicate Birthday Check:**

* The program checks if any two people in the group share the same birthday by using a Set to track encountered birthdays.
* If a duplicate is found, that run is considered a "shared birthday" occurrence.

1. **Probability Calculation:**

* After running the simulation the specified number of times, the program calculates the probability by dividing the number of runs with a shared birthday by the total number of runs.
* The result is printed to the console.

## **Program Structure**

1. Person Class

* **Purpose:**Represents an individual in the simulation.
* **Key Attribute:**
* int birthday: Stores the birthday as an integer (1–365).
* **Simplification:**Using an integer simplifies both the random birthday generation and the process of checking for duplicate birthdays.

2. BirthdaySimulation Class

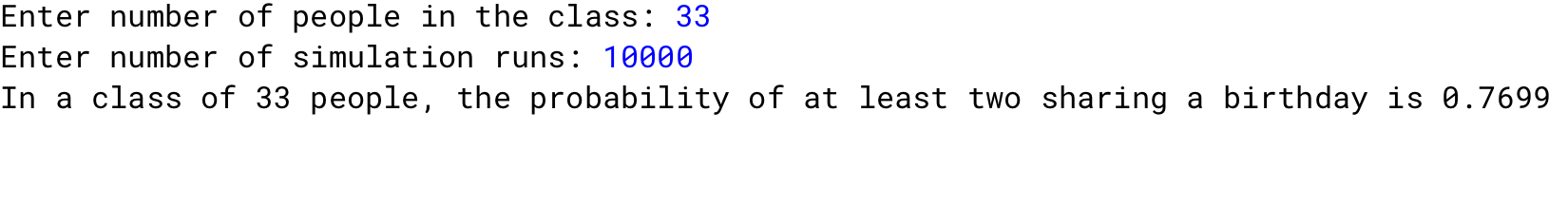
* **Purpose:**Contains the core simulation logic.
* **Key Methods:**
* hasSharedBirthday(List<Person> people):  
  Checks if there is at least one duplicate birthday in the list of Person objects.
* runSimulation(int numPeople, int runs):  
  Handles the simulation runs, creates groups of people, checks for shared birthdays, and calculates the final probability.
* **Configuration:**The number of simulation runs and the class size are configurable through method parameters, making the simulation flexible without requiring changes to the loop structure.

3. Main Class

* **Purpose:**Serves as the entry point of the application.
* **Responsibilities:**
* Collects user input for the class size and number of simulation runs.
* Calls the simulation method from the BirthdaySimulation class.
* Keeps the main method focused on object construction and control flow rather than the simulation logic itself.

**Results**

Through Scanner:



Through Method Call:

