Computer Simulation Laboratory

B.Tech. 5th Semester

Department: Computer Science and Engineering

Faculty of Engineering & Technology

M. S. Ramaiah University of Applied Sciences

M. S. Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	5th Semester
Name of the Laboratory	Computer Simulation Laboratory
Laboratory Code	CSC308A

List of Experiments

1. Introduction to Java simulation and Implementing a Java program for random numbers generation for given scenario
2. Simulation of a single server queue (Grocery centre problem)
3. Simulation of a Two server Queue (Able Baker Problem)
4. Discrete Distributions AND Continuous Distributions
5. Random Number generator using LCG
6. Random Variate Generator using lnverse-Transfonn Technique

    Exponential Distribution

    Uniform Distribution
7. Test for random numbers

    KS test

    Chi square Test
8. Simulation of a single server Single queue(M/M/1)

NAME: SATYAJIT GHANA REG NO: 17ETCS002159

Laboratory 1: Introduction to java simulation and Random Number Generation

1. Introduction and Purpose of Experiment

Computer simulation provides students to design and implement computer simulation models, conduct simulation experiments and evaluate system performance. This laboratory exercise will help the students to get familiar with using object-oriented simulation in Java.

Java (Structured Parallel Discrete Event Simulation in Java) system is designed to incorporate the parallel programming technology into discrete event simulations. The java system adopts the approach of augmenting a general-purpose language with essential constructs to support simulation modeling based on the process-oriented modeling technology.

Random numbers are widely used ingredient in the simulation of almost all discrete systems. Simulation languages generate random numbers that are used to generate event times and other random variables. Random number generators have applications in gambling, statistical sampling, computer simulation, cryptography, completely randomized design and other areas where producing an unpredictable result is desirable. The generation of pseudo random numbers is an important and common task in computer programming.

2. Aim and Objectives

Aim

- To use Netbeans and understand using object-oriented simulation in Java
- To develop programs generating random numbers and Understand its significance in various applications

Objectives

At the end of this lab, the student will be able to

- Explain the features and use of Netbeans IDE to develop java programs for simulation
- Edit, compile and execute java programs successfully using Netbeans IDE

  Use different random generation methods for generating random numbers

  Create java programs for generating random numbers

3. Experimental Procedure

Students are given a set of programs for generating random numbers using built-in methods. Programs should be edited, compiled and executed using Netbeans IDE.

Random number generation using inbuilt methods/manually

Ex: coin toss, die, and cards

4. Calculations/Computations/Algorithms

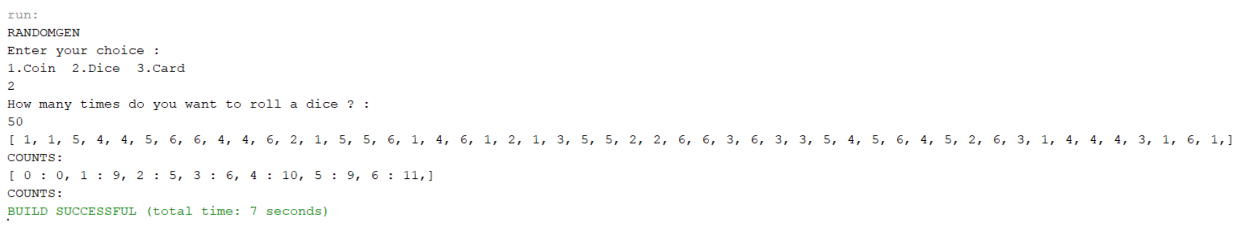
Generate a random numbers for coin flip, die and cards

5. Presentation of Results

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rnumgen;

import java.util.Random;
import java.util.Scanner;

/**
 *
 * @author Student
 */
public class RNUMGEN {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Random rgen = new Random();

        Scanner input = new Scanner(System.in);

        System.out.println("RANDOMGEN");
        System.out.println("Enter your choice : ");
        System.out.println("1.Coin\t2.Dice\t3.Card");
        int choice = input.nextInt();
        switch(choice) {
            case 1: {
                System.out.println("How many times do you want to flip a coin ? : ");
            int N = input.nextInt();
            System.out.print("[ ");
            int count = 0;
            for (int i = 0 ; i < N ; i++) {
                int val = rgen.nextInt(2);
                if (val == 0)
                    count++;
                System.out.print( val + ", ");
            }
            System.out.print("\b]");
            System.out.println("\nCOUNT 0 : " + count + " , 1 : " + (N-count) );
            break;
        }
        case 2: {
            int []count = new int[7];
            System.out.println("How many times do you want to roll a dice ? : ");
            int N = input.nextInt();
            System.out.print("[ ");
            for (int i = 0 ; i < N ; i++) {
                int val = (rgen.nextInt(6)+1);
                count[val]++;
                System.out.print( val + ", ");
            }
            System.out.print("\b]\nCOUNTS: \n");
            System.out.print("[ ");
            int j = 0;
            for (int e : count) {
                System.out.print(j + " : " + e + ", ");
                j++;
            }
            System.out.print("\b]\nCOUNTS: \n");
            break;
        }
        case 3: {
            int []count = new int[53];
            System.out.println("How many card do you want ? : ");
            int N = input.nextInt();
            System.out.print("[ ");
            for (int i = 0 ; i < N ; i++) {
                int val = (rgen.nextInt(52)+1);
                count[val]++;
                System.out.print( val + ", ");
            }
            System.out.print("\b]\nCOUNTS: \n");
            System.out.print("[ ");
            int j = 0;
```

```
                for (int e : count) {
                    System.out.print(j + " : " + e + ", ");
                    j++;
                }
                System.out.print("\b]\nCOUNTS: \n");
                break;
            }
        }
    }

}
```

6. Analysis and Discussions



```
run:
RANDOMGEN
Enter your choice :
1.Coin  2.Dice  3.Card
2
How many times do you want to roll a dice ? :
50
[ 1, 1, 5, 4, 4, 5, 6, 6, 4, 4, 6, 2, 1, 5, 5, 6, 1, 4, 6, 1, 2, 1, 3, 5, 5, 2, 2, 6, 6, 3, 6, 3, 3, 5, 4, 5, 6, 4, 5, 2, 6, 3, 1, 4, 4, 4, 3, 1, 6, 1,]
COUNTS:
[ 0 : 0, 1 : 9, 2 : 5, 3 : 6, 4 : 10, 5 : 9, 6 : 11,]
COUNTS:
BUILD SUCCESSFUL (total time: 7 seconds)
```

7. Conclusions

Random class is used to generate pseudo-random numbers in java. An instance of this class is thread-safe. The instance of this class is however cryptographically insecure. This class provides various method calls to generate different random data types such as float, double, int.

Constructors:

Random(): Creates a new random number generator

Random(long seed): Creates a new random number generator using a single long seed

The random() method returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. When you call Math.random(), under the hood, a java.util.Random pseudorandom-number generator object is created and used.

The java.util.Random class implements what is generally called a linear congruential generator (LCG). It is designed to be fast but does not meet requirements for real-time use, such as use in unique session ID generation on a web server, scientific experiments, cryptography, or lotteries and sweepstakes where a monetary stake is involved.

Laboratory 2: Simulation of a single server queue (Grocery centre problem)

1. Introduction and Purpose of Experiment

2. Aim and Objectives

3. Experimental Procedure

An ATM booth has a single machine to withdraw cash. Customers arrive at the ATM at random times that are from 1 to 8 minutes apart. Each Inter-arrival time has the same probability of occurrence and service times vary from 1 to 6 minutes with the respective probabilities of time taken for service shown in Table 3 below:

Service(in minutes)	1	2	3	4	5	6
Probability	0.10	0.20	0.30	0.25	0.10	0.05

4. Table 3

Simulate the system for arrival of 1000 customers starting with an empty ATM queue to determine the following:

i. Average waiting time of a customer
ii. Idle time of the ATM machine
iii. Average service time
iv. Average time between arrivals

Use random numbers between 1 to 1000 to determine inter arrival time, and random numbers between 1 to 100 to determine service time.

5. Algorithms

```
single_server_queue_model():
1. for each request in queue:
2.     AT[i] = AT[i-1] + IAT[i]
3.     if (AT[i] >= SE[i-1])
4.             SS[i] = AT[i-1]
5.     else SS[i] = SE[i-1]
6.     SE[i] = SS[i] + ST[i]
7.     WAIT[i] = SS[i]-AT[i]
```

8. IDLE[i] = AT[i] - SE[i-1] >= 0 ? AT[i] - SE[i1] : 0

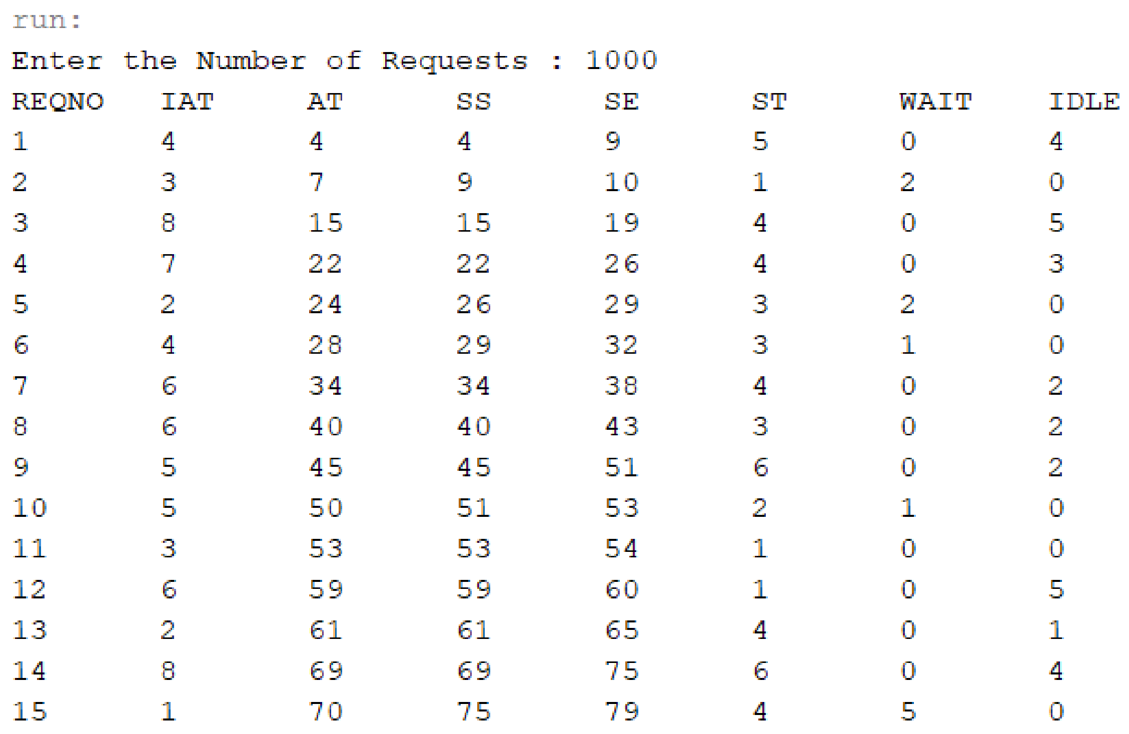
6. Presentation of Results

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab02;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.OptionalDouble;
import java.util.Random;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

/**
 *
 * @author shadowleaf
 */
public class Lab02 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the Number of Requests : ");
        Integer N = input.nextInt();

        Random rand = new Random();

        // Taking it as input
//        List<Integer> IAT = new ArrayList<>();
//        List<Integer> ST = new ArrayList<>();
//        IAT.add(0);
//        ST.add(0);
```

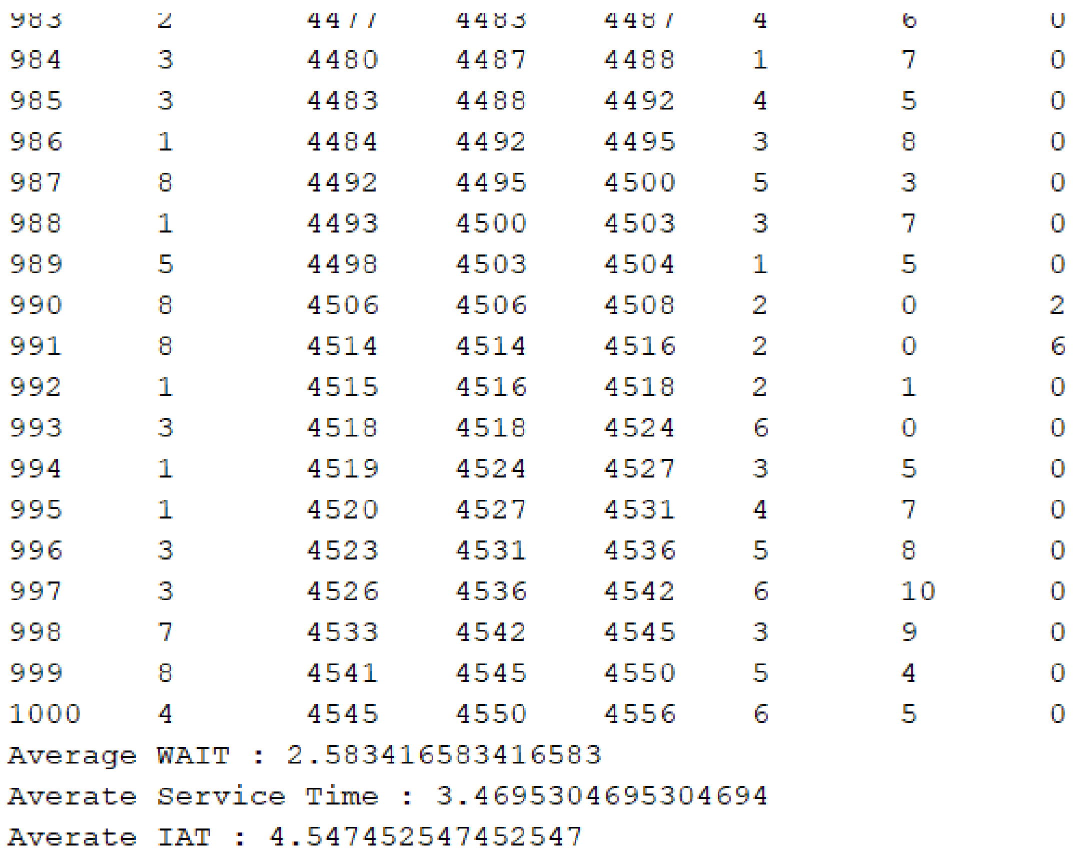
```
//
//        System.out.print("Enter IAT's : ");
//        for (int i = 0 ; i < N ; i++) {
//            IAT.add(input.nextInt());
//        }
//
//        System.out.print("Enter ST's : ");
//        for (int i = 0 ; i < N ; i++) {
//            ST.add(input.nextInt());
//        }

        // Inter Arrival Times range from 1 - 8 mins
        List<Integer> IAT = IntStream.range(0, N + 1).mapToObj(i -
> rand.nextInt(8) + 1).collect(Collectors.toList());
        // Service Time ranges from 1 - 6 mins
        List<Integer> ST = IntStream.range(0, N + 1).mapToObj(i -
> rand.nextInt(6) + 1).collect(Collectors.toList());

        Map<String, List<Integer>> SIM_TAB = new HashMap<>();

        SIM_TAB.put("IAT", IAT);
        SIM_TAB.put("ST", ST);
        SIM_TAB.put("SS", new ArrayList<>(Collections.nCopies(N + 1, 0)));
        SIM_TAB.put("AT", new ArrayList<>(Collections.nCopies(N + 1, 0)));
        SIM_TAB.put("SE", new ArrayList<>(Collections.nCopies(N + 1, 0)));
        SIM_TAB.put("WAIT", new ArrayList<>(Collections.nCopies(N + 1, 0)));
        SIM_TAB.put("IDLE", new ArrayList<>(Collections.nCopies(N + 1, 0)));

        SIM_TAB.get("AT").set(0, 0);
        for (int i = 1; i <= N; i++) {
            SIM_TAB.get("AT").set(i, SIM_TAB.get("AT").get(i - 1) + SIM_TAB.get("IAT").ge
t(i));
            if (SIM_TAB.get("AT").get(i) >= SIM_TAB.get("SE").get(i - 1)) {
                SIM_TAB.get("SS").set(i, SIM_TAB.get("AT").get(i));
            } else {
                SIM_TAB.get("SS").set(i, SIM_TAB.get("SE").get(i - 1));
            }

            SIM_TAB.get("SE").set(i, SIM_TAB.get("SS").get(i) + SIM_TAB.get("ST").get(i))
;

            SIM_TAB.get("WAIT").set(i, SIM_TAB.get("SS").get(i) - SIM_TAB.get("AT").get(i
));

            SIM_TAB.get("IDLE").set(i, SIM_TAB.get("AT").get(i) - SIM_TAB.get("SE").get(i
-1) >= 0 ? SIM_TAB.get("AT").get(i) - SIM_TAB.get("SE").get(i-1) : 0);
```

```
        }

        System.out.println("REQNO\tIAT\tAT\tSS\tSE\tST\tWAIT\tIDLE");
        for (int i = 1; i <= N; i++) {
            String out = i + "\t"
                    + SIM_TAB.get("IAT").get(i) + "\t"
                    + SIM_TAB.get("AT").get(i) + "\t"
                    + SIM_TAB.get("SS").get(i) + "\t"
                    + SIM_TAB.get("SE").get(i) + "\t"
                    + SIM_TAB.get("ST").get(i) + "\t"
                    + SIM_TAB.get("WAIT").get(i) + "\t"
                    + SIM_TAB.get("IDLE").get(i);
            System.out.println(out);
        }

        // Avg WAIT, Avg. Ser, Avg. IAT
        OptionalDouble avgWAIT = SIM_TAB.get("WAIT").stream().mapToDouble(a -
> a).average();
        OptionalDouble avgService = SIM_TAB.get("ST").stream().mapToDouble(a -
> a).average();
        OptionalDouble avgIAT = SIM_TAB.get("IAT").stream().mapToDouble(e -
> e).average();
        System.out.println("Average WAIT : " + avgWAIT.getAsDouble());
        System.out.println("Averate Service Time : " + avgService.getAsDouble());
        System.out.println("Averate IAT : " + avgIAT.getAsDouble());
    }

}
```

7. Analysis and Discussions



```
run:
Enter the Number of Requests : 1000
REQNO     IAT       AT        SS        SE        ST        WAIT      IDLE
1         4         4         4         9         5         0         4
2         3         7         9         10        1         2         0
3         8         15        15        19        4         0         5
4         7         22        22        26        4         0         3
5         2         24        26        29        3         2         0
6         4         28        29        32        3         1         0
7         6         34        34        38        4         0         2
8         6         40        40        43        3         0         2
9         5         45        45        51        6         0         2
10        5         50        51        53        2         1         0
11        3         53        53        54        1         0         0
12        6         59        59        60        1         0         5
13        2         61        61        65        4         0         1
14        8         69        69        75        6         0         4
15        1         70        75        79        4         5         0
```

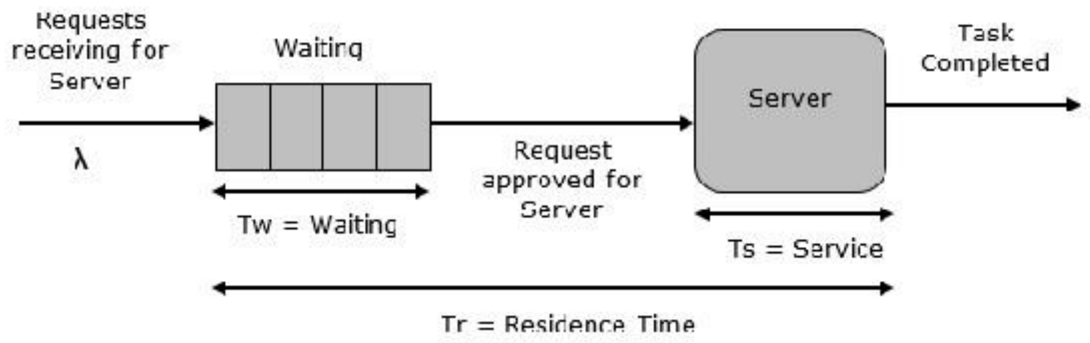


```
983       2         4477      4483      4487      4         6         0
984       3         4480      4487      4488      1         7         0
985       3         4483      4488      4492      4         5         0
986       1         4484      4492      4495      3         8         0
987       8         4492      4495      4500      5         3         0
988       1         4493      4500      4503      3         7         0
989       5         4498      4503      4504      1         5         0
990       8         4506      4506      4508      2         0         2
991       8         4514      4514      4516      2         0         6
992       1         4515      4516      4518      2         1         0
993       3         4518      4518      4524      6         0         0
994       1         4519      4524      4527      3         5         0
995       1         4520      4527      4531      4         7         0
996       3         4523      4531      4536      5         8         0
997       3         4526      4536      4542      6         10        0
998       7         4533      4542      4545      3         9         0
999       8         4541      4545      4550      5         4         0
1000      4         4545      4550      4556      6         5         0
Average WAIT : 2.583416583416583
Averate Service Time : 3.4695304695304694
Averate IAT : 4.547452547452547
```

8. Conclusions

Single Server Queue

This is the simplest queuing system as represented in the following figure. The central element of the system is a server, which provides service to the connected devices or items. Items request to the system to be served, if the server is idle. Then, it is served immediately, else it joins a waiting queue. After the task is completed by the server, the item departs.



Laboratory 3: Simulation of a two server Queue (Able Baker Problem)

1. Introduction and Purpose of Experiment

2. Aim and Objectives

3. Experimental Procedure

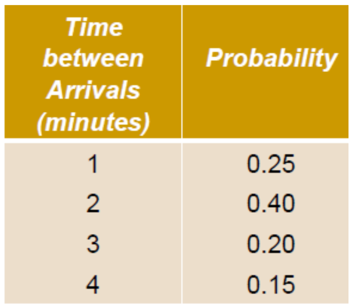
An ATM booth has a two machine to withdraw cash. Customers arrive at the ATM at random times that are from 1 to 4 minutes apart with the respective probabilities of time taken for arrival shown in Table below. Service times of machine **A**ble vary from 2 to 5 minutes and service times of machine **Baker** vary from 3 to 6 minutes with the respective probabilities of time taken for service shown in Table below:

Simulate the system for arrival of 1000 customers starting with an empty ATM queue to determine the following:

i. Average waiting time of a customer

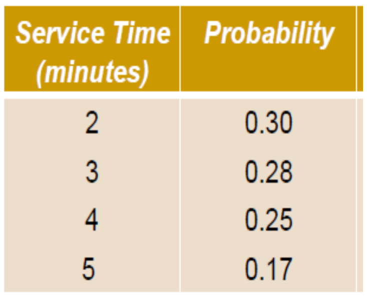
ii. Total time in the system

IAT



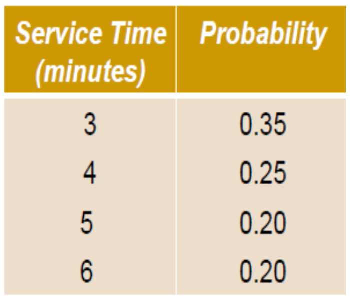
Time between Arrivals (minutes)	Probability
1	0.25
2	0.40
3	0.20
4	0.15

Able



Service Time (minutes)	Probability
2	0.30
3	0.28
4	0.25
5	0.17

Baker



Service Time (minutes)	Probability
3	0.35
4	0.25
5	0.20
6	0.20

Use random numbers between 1 to 1000 to determine inter arrival time, and random numbers between 1 to 100 to determine service time

4. Algorithms

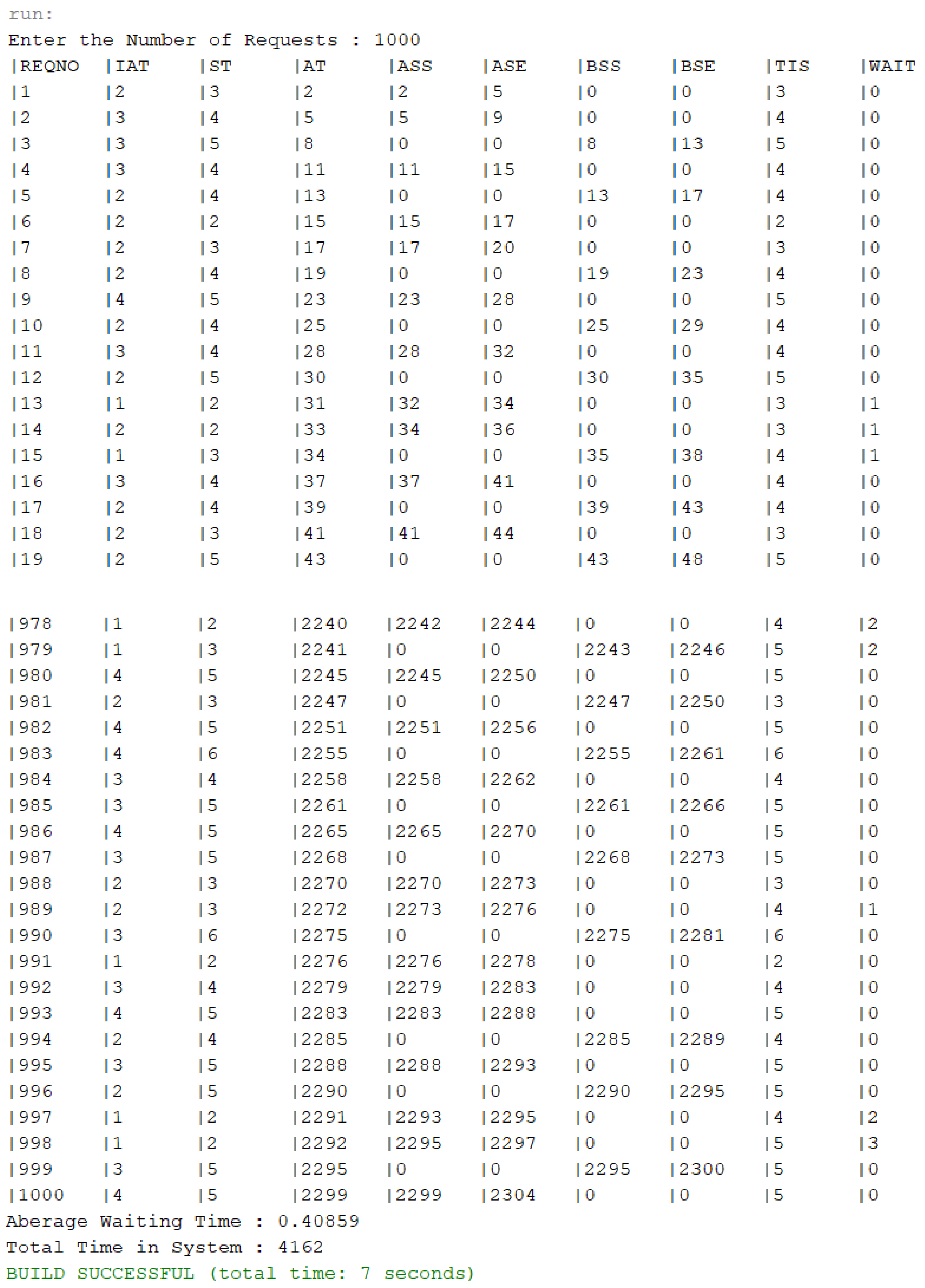
5. Presentation of Results

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```java
package lab03;

import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

/**
 *
 * @author shadowleaf
 */
public class Lab03 {

    /**
     * @param arr
     * @param e
     * @param sta
     * @param end
     */

    public static void fillArray(int arr[], int e, int sta, int end) {
        for (int i = sta ; i <= end ; i++) {
            arr[i] = e;
        }
    }

    public static int[] prepareProbDist(double prob[], int ass_prob[]) {
        int[] cumulativeProb = new int[ass_prob.length];
        int[] prob_lookup = new int[101];

        cumulativeProb[0] = (int)(prob[0] * 100);

        for (int i = 1 ; i < cumulativeProb.length ; i++) {
            cumulativeProb[i] = (int)(cumulativeProb[i-1] + prob[i] * 100);
        }

        int start = 0;
        for (int i = 0 ; i < cumulativeProb.length ; i++) {
            int end = cumulativeProb[i];
            fillArray(prob_lookup, ass_prob[i], start, end);
            start = end;
        }

        return prob_lookup;
    }

    /**
```

```
     *
     * @param args
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the Number of Requests : ");
        Integer N = input.nextInt();

        Random rand = new Random();

        double[] IATPROB = {0.25, 0.40, 0.20, 0.15};
        int[] IAT_ASS = {1, 2, 3, 4}; // IAT Associated
        int[] IAT_LOOKUP = prepareProbDist(IATPROB, IAT_ASS);

        double[] APROB = {0.30, 0.28, 0.25, 0.17};
        int[] AST_ASS = {2, 3, 4, 5};
        int[] AST_LOOKUP = prepareProbDist(APROB, AST_ASS);

        double[] BPROB = {0.35, 0.25, 0.20, 0.20};
        int[] BST_ASS = {3, 4, 5, 6};
        int[] BST_LOOKUP = prepareProbDist(BPROB, BST_ASS);

        // generate IAT, AST, BST

        int[] IAT = new int[N+1];
        int[] AST = new int[N+1];
        int[] BST = new int[N+1];

//        int[] IAT = new int[N+1];
        int[] ST = new int[N+1];
        int[] AT = new int[N+1];
        int[] ASS = new int[N+1];
        int[] ASE = new int[N+1];
        int[] BSS = new int[N+1];
        int[] BSE = new int[N+1];
        int[] TIS = new int[N+1];
        int[] WAIT = new int[N+1];

// Randomly directly generate the IAT and ST
//        for (int i = 1 ; i <= N ; i++) {
//            IAT[i] = rand.nextInt(8) + 1;
//            ST[i] = rand.nextInt(6) + 1;
//        }

// Testing :
//        int[] ST = {0, 6, 1, 5, 5, 5};
//          int[] IAT = {0, 3, 2, 3, 1, 5};

        int idxAbleLastBusy = 0;
        int idxBakerLastBusy = 0;

        for (int i = 1 ; i <= N ; i++) {

            // generate random number from 0 to 100
            int randNum = rand.nextInt(101);

            // get a random IAT from the lookup table
            IAT[i] = IAT_LOOKUP[randNum];

            AT[i] = AT[i-1] + IAT[i];

            boolean ableIsFree = ASE[idxAbleLastBusy] <= AT[i];
            boolean bakerIsFree = BSE[idxBakerLastBusy] <= AT[i];

            if (ableIsFree) {
                // get a random service time for Able
                ST[i] = AST_LOOKUP[randNum];

                ASS[i] = AT[i]; ASE[i] = AT[i] + ST[i];
                TIS[i] = AT[i] - ASE[i]; WAIT[i] = ASS[i] - AT[i];
                idxAbleLastBusy = i;
            } else if (bakerIsFree) {
                // get a random serivce time for Baker
                ST[i] = BST_LOOKUP[randNum];

                BSS[i] = AT[i]; BSE[i] = AT[i] + ST[i];
                TIS[i] = AT[i] - BSE[i]; WAIT[i] = BSS[i] - AT[i];
                idxBakerLastBusy = i;
            } else { // neither of them is free

                // check who gets free first
                boolean isAbleFreeFirst = ASE[idxAbleLastBusy] <= BSE[idxBakerLastBusy];

                if (isAbleFreeFirst) { // able is free first
                    ST[i] = AST_LOOKUP[randNum];

                    ASS[i] = ASE[idxAbleLastBusy]; ASE[i] = ASE[idxAbleLastBusy] + ST[i];
                    TIS[i] = AT[i] - ASE[i]; WAIT[i] = ASS[i] - AT[i];

                    idxAbleLastBusy = i;
                } else { // baker is free first
                    ST[i] = BST_LOOKUP[randNum];
                    BSS[i] = BSE[idxBakerLastBusy]; BSE[i] = BSE[idxBakerLastBusy] + ST[i
];
                    TIS[i] = AT[i] - BSE[i]; WAIT[i] = BSS[i] - AT[i];

                    idxBakerLastBusy = i;
                }
            }

            // mat pucho ye bakwaas kahe kiye hum
            TIS[i] *= -1;

        }

        System.out.printf("|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t|%s\t\n", "REQNO"
, "IAT", "ST", "AT", "ASS", "ASE", "BSS", "BSE", "TIS", "WAIT");
        for (int i = 1 ; i <= N ; i++) {
            System.out.printf("|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|%s\t|%s\t\n", i,
IAT[i], ST[i], AT[i], ASS[i], ASE[i], BSS[i], BSE[i], TIS[i], WAIT[i]);
        }

        System.out.printf("Aberage Waiting Time : %.5f\nTotal Time in System : %d\n", Arr
ays.stream(WAIT).mapToDouble(a -> a).average().getAsDouble(), Arrays.stream(TIS).sum());

    }

}
```

6. Analysis and Discussions



```
run:
Enter the Number of Requests : 1000
|REQNO   |IAT     |ST      |AT      |ASS     |ASE     |BSS     |BSE     |TIS     |WAIT
|1       |2       |3       |2       |2       |5       |0       |0       |3       |0
|2       |3       |4       |5       |5       |9       |0       |0       |4       |0
|3       |3       |5       |8       |0       |0       |8       |13      |5       |0
|4       |3       |4       |11      |11      |15      |0       |0       |4       |0
|5       |2       |4       |13      |0       |0       |13      |17      |4       |0
|6       |2       |2       |15      |15      |17      |0       |0       |2       |0
|7       |2       |3       |17      |17      |20      |0       |0       |3       |0
|8       |2       |4       |19      |0       |0       |19      |23      |4       |0
|9       |4       |5       |23      |23      |28      |0       |0       |5       |0
|10      |2       |4       |25      |0       |0       |25      |29      |4       |0
|11      |3       |4       |28      |28      |32      |0       |0       |4       |0
|12      |2       |5       |30      |0       |0       |30      |35      |5       |0
|13      |1       |2       |31      |32      |34      |0       |0       |3       |1
|14      |2       |2       |33      |34      |36      |0       |0       |3       |1
|15      |1       |3       |34      |0       |0       |35      |38      |4       |1
|16      |3       |4       |37      |37      |41      |0       |0       |4       |0
|17      |2       |4       |39      |0       |0       |39      |43      |4       |0
|18      |2       |3       |41      |41      |44      |0       |0       |3       |0
|19      |2       |5       |43      |0       |0       |43      |48      |5       |0

|978     |1       |2       |2240    |2242    |2244    |0       |0       |4       |2
|979     |1       |3       |2241    |0       |0       |2243    |2246    |5       |2
|980     |4       |5       |2245    |2245    |2250    |0       |0       |5       |0
|981     |2       |3       |2247    |0       |0       |2247    |2250    |3       |0
|982     |4       |5       |2251    |2251    |2256    |0       |0       |5       |0
|983     |4       |6       |2255    |0       |0       |2255    |2261    |6       |0
|984     |3       |4       |2258    |2258    |2262    |0       |0       |4       |0
|985     |3       |5       |2261    |0       |0       |2261    |2266    |5       |0
|986     |4       |5       |2265    |2265    |2270    |0       |0       |5       |0
|987     |3       |5       |2268    |0       |0       |2268    |2273    |5       |0
|988     |2       |3       |2270    |2270    |2273    |0       |0       |3       |0
|989     |2       |3       |2272    |2273    |2276    |0       |0       |4       |1
|990     |3       |6       |2275    |0       |0       |2275    |2281    |6       |0
|991     |1       |2       |2276    |2276    |2278    |0       |0       |2       |0
|992     |3       |4       |2279    |2279    |2283    |0       |0       |4       |0
|993     |4       |5       |2283    |2283    |2288    |0       |0       |5       |0
|994     |2       |4       |2285    |0       |0       |2285    |2289    |4       |0
|995     |3       |5       |2288    |2288    |2293    |0       |0       |5       |0
|996     |2       |5       |2290    |0       |0       |2290    |2295    |5       |0
|997     |1       |2       |2291    |2293    |2295    |0       |0       |4       |2
|998     |1       |2       |2292    |2295    |2297    |0       |0       |5       |3
|999     |3       |5       |2295    |0       |0       |2295    |2300    |5       |0
|1000    |4       |5       |2299    |2299    |2304    |0       |0       |5       |0
Aberage Waiting Time : 0.40859
Total Time in System : 4162
BUILD SUCCESSFUL (total time: 7 seconds)
```

7. Conclusions

Laboratory 4: Discrete Distributions AND Continuous Distributions

1. Introduction and Purpose of Experiment

2. Aim and Objectives

3. Experimental Procedure

   Design and Implement a Java program for the following **Discrete Probability Distribution**

   1. Binomial distribution

      i. To find the number of successes in **n** independent Bernoulli trials, given that **X** has a binomial distribution

      ii. Calculate the

         1. Mean, E(X)
         2. Variance, V(X)

   2. Geometric distribution

      i. To identify the number of Bernoulli trials, *X*, to achieve the 1st success

      ii. Calculate the

         1. Mean, *E(X)*
         2. Variance *V(X)*

   3. Negative binomial distribution

      i. To identify the number of Bernoulli trials, X, until the kth success

      ii. Calculate the Mean, E(X) and Variance V(X)

   4. Develop and implement a Java program by selecting suitable distribution function for given scenario:If 40% of the assembled ink-jet printers are rejected at the inspection station. Your program should identify:

      i. Probability that the first acceptable ink-jet printer is the third one inspected. Considering each inspection as a Bernoulli trial with q=0.4 and p=0.6.

      ii. Probability that the third printer inspected is the second acceptable printer

4. Design and Implement a Java program for the following **Continuous Distribution**

   A computer repair person is “beeped” each time there is a call for service. If the number of beeps per hour is Poisson distributed (α = 2 beeps per hour). Then design and implement a Java program to determine the following.

      i. The probability of exactly three beeps in the next hour:

ii. The probability of two or more beeps in a 1-hour period:

5. Algorithms

6. Presentation of Results

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab04;

import java.util.Scanner;

/**
 *
 * @author shadowleaf
 */
public class Lab04 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        Distribution.DISTR_TYPE currDistr;

        Integer choice;

        System.out.print("Enter the Distrubution you want\n1.\tBinomial\n2.\tGeometrical\
n3.\tNegative Binomial\n4.\tPoisson\nYour Choice : ");
        choice = input.nextInt();

        Distribution distr = null;

        switch (choice) {
            case 1: {
                // Binomial
                currDistr = Distribution.DISTR_TYPE.BINOM;
                System.out.print("Enter N : ");
                Long N = input.nextLong();
                System.out.print("Enter X : ");
                Long K = input.nextLong();
                System.out.print("Enter P (success) : ");
                Double P = input.nextDouble();

                distr = new BinomialDistribution(currDistr, N, P, K);
            }
            break;
            case 2: {
                // Geometrical
                currDistr = Distribution.DISTR_TYPE.GEOMT;
                System.out.print("Enter P (success) : ");
                Double P = input.nextDouble();
                System.out.print("Enter K : ");
                Long K = input.nextLong();

                distr = new GeometricDistribution(currDistr, P, K);
            }
            break;
            case 3: {
                // Negative Binomial
                currDistr = Distribution.DISTR_TYPE.NBINOM;
                System.out.print("Enter P (success) : ");
                Double P = input.nextDouble();
                System.out.print("Enter N : ");
                Long N = input.nextLong();
                System.out.print("Enter R : ");
                Long R = input.nextLong();

                distr = new NegativeBinomialDistribution(currDistr, P, N, R);
            }
            break;
            case 4: {
                // Poisson
                currDistr = Distribution.DISTR_TYPE.POISSON;
                System.out.print("Enter lambda : ");
                Double lambda = input.nextDouble();
                System.out.print("Enter K : ");
                Long K = input.nextLong();

                distr = new PoissonDistribution(currDistr, lambda, K);
            }
            break;
            default:
                main(args);
                return;
        }
        if (distr != null) {
            System.out.printf("P( X = %d ) = %.10f\nE[X] = %.10f\nVar[X] = %.10f\n",
                    distr.getParam(), distr.getDistribution(), distr.getExpectance(), dis
tr.getVariance());
        } else {
            System.out.println("DISTR IS NULL");
        }

    }

}




/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab04;



/**
 *
 * @author shadowleaf
 */
public abstract class Distribution {

    public static enum DISTR_TYPE {
        BINOM, GEOMT, NBINOM, POISSON
    };

    public Distribution(DISTR_TYPE currentDistr) {
        this.currentDistr = currentDistr;
    }


    public DISTR_TYPE currentDistr;

    public static Long choose(Long n, Long r) {
        if (r > n/2)
            r = n - r;
        Long ans = 1l;
        for (int i = 1 ; i <= r ; i++) {
            ans *= (n - r + i);
            ans /= i;
        }

        return ans;
    }

    public abstract Long getParam();

    /**
     *
     * Calculates P(X = K) generally speaking
     *
     * @return
     */
    public abstract Double getDistribution();

    public abstract Double getExpectance();

    public abstract Double getVariance();

}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab04;

/**
 *
 * @author shadowleaf
 */
public class BinomialDistribution extends Distribution {

    Long N, K;
    Double P;

    public BinomialDistribution(DISTR_TYPE currentDistr, Long N, Double P, Long K) {
        super(currentDistr);
        this.N = N;
```

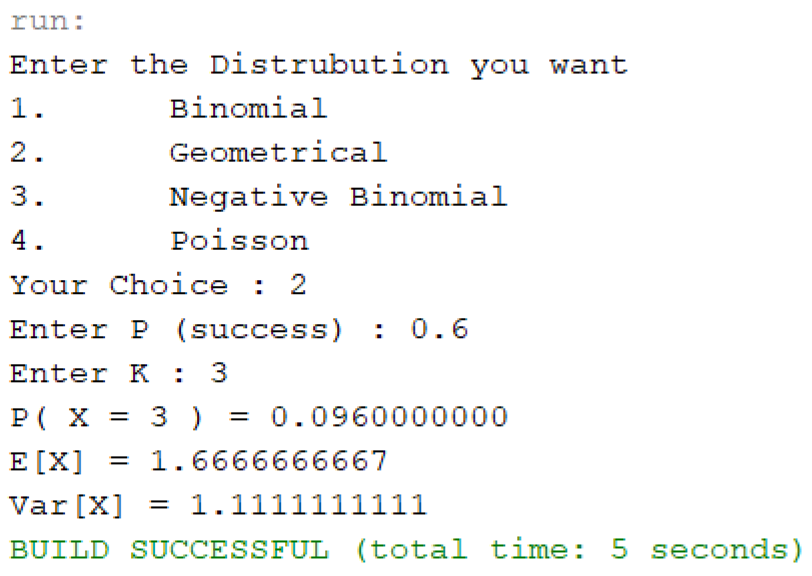
```java
        this.P = P;
        this.K = K;
    }

    public Double binomialDistribution() {
        return choose(N, K) * (double)Math.pow(P, K) * (double)Math.pow( 1 - P, N - K);
    }

    @Override
    public Double getDistribution() {
        return binomialDistribution();
    }

    @Override
    public Double getExpectance() {
        return (double)N * P;
    }

    @Override
    public Double getVariance() {
        return N * P * (1.0 - P);
    }

    @Override
    public Long getParam() {
        return K;
    }

}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab04;

/**
 *
 * @author shadowleaf
 */
public class GeometricDistribution extends Distribution {
    Double P;
    Long K;

    public GeometricDistribution(DISTR_TYPE currentDistr, Double P, Long K) {
        super(currentDistr);
        this.P = P;
        this.K = K;
    }

    public Double geometricDistribution() {
        return (double)Math.pow(1 - P, K - 1) * P;
    }

    @Override
    public Double getDistribution() {
        return geometricDistribution();
    }

    @Override
    public Double getExpectance() {
        return (1.0) / P;
    }

    @Override
    public Double getVariance() {
        return (1 - P) / (P * P);
    }

    @Override
    public Long getParam() {
        return K;
    }

}
```

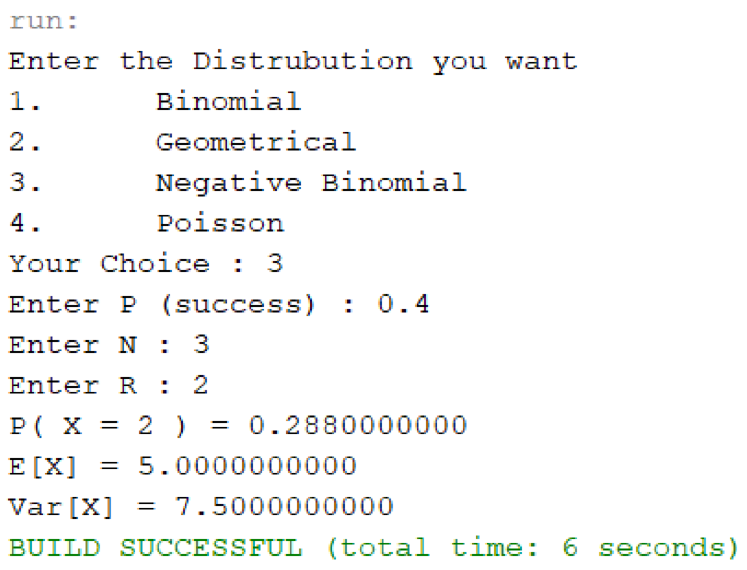
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab04;

/**
 *
 * @author shadowleaf
 */
public class NegativeBinomialDistribution extends Distribution {

    Double P;
    Long R;
    Long N;

    public NegativeBinomialDistribution(DISTR_TYPE currentDistr, Double P, Long N, Long R
) {
        super(currentDistr);
        this.P = P;
        this.R = R;
        this.N = N;
    }

    @Override
    public Double getDistribution() {
        return choose(N-1, R-1) * Math.pow(P, N-R) * (double)Math.pow(1-P, R);
    }

    @Override
    public Double getExpectance() {
        return R / P;
    }

    @Override
    public Double getVariance() {
        return (R * (1.0 - P)) / (P * P);
    }

    @Override
    public Long getParam() {
        return R;
    }

}




/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```java
package lab04;

/**
 *
 * @author shadowleaf
 */
public class PoissonDistribution extends Distribution {

    Double lambda;
    Long K;

    public PoissonDistribution(DISTR_TYPE currentDistr, Double lambda, Long K) {
        super(currentDistr);

        this.lambda = lambda;
        this.K = K;
    }

    @Override
    public Double getDistribution() {
        Double ans = Math.pow(lambda, K) * Math.exp(-lambda);
        for (Long i = 1l; i <= K; i++) {
            ans /= i;
        }

        return ans;
    }

    @Override
    public Double getExpectance() {
        return lambda;
    }

    @Override
    public Double getVariance() {
        return getExpectance();
    }

    @Override
    public Long getParam() {
        return K;
    }

}
```

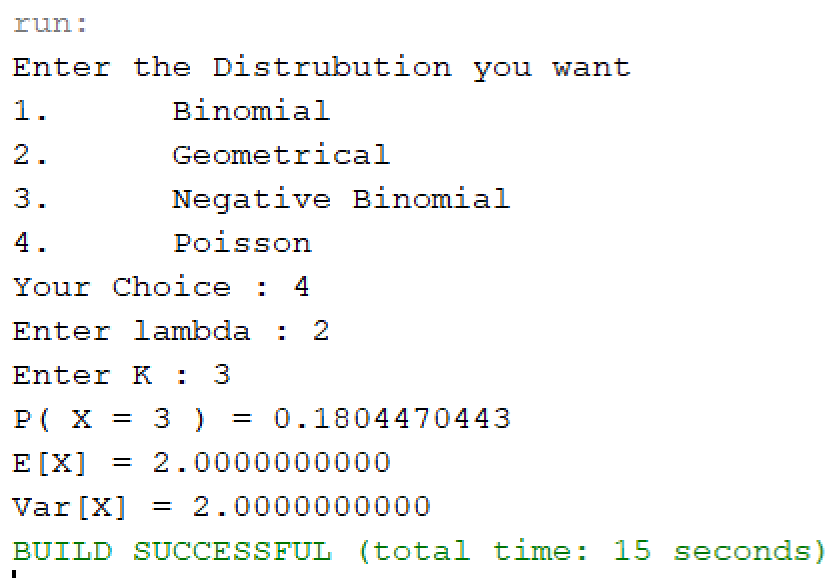
7. Analysis and Discussions



```
run:
Enter the Distrubution you want
1.      Binomial
2.      Geometrical
3.      Negative Binomial
4.      Poisson
Your Choice : 2
Enter P (success) : 0.6
Enter K : 3
P( X = 3 ) = 0.0960000000
E[X] = 1.6666666667
Var[X] = 1.1111111111
BUILD SUCCESSFUL (total time: 5 seconds)
```



```
run:
Enter the Distrubution you want
1.      Binomial
2.      Geometrical
3.      Negative Binomial
4.      Poisson
Your Choice : 3
Enter P (success) : 0.4
Enter N : 3
Enter R : 2
P( X = 2 ) = 0.2880000000
E[X] = 5.0000000000
Var[X] = 7.5000000000
BUILD SUCCESSFUL (total time: 6 seconds)
```



```
run:
Enter the Distrubution you want
1.        Binomial
2.        Geometrical
3.        Negative Binomial
4.        Poisson
Your Choice : 4
Enter lambda : 2
Enter K : 3
P( X = 3 ) = 0.1804470443
E[X] = 2.0000000000
Var[X] = 2.0000000000
BUILD SUCCESSFUL (total time: 15 seconds)
```

8. Conclusions

1. Bernoulli Distribution

$$P(X = x) = p^x(1-p)^{1-x}$$

$$\mu = p$$
$$\sigma^2 = p(1-p)$$

2. Binomial Distribution

$$P(X = x) = nC_x p^x q^{n-x}$$

$$\mu = np$$
$$\sigma^2 = np(1-p)$$

3. Geometric Distribution

$$P(X = x) = p(1-p)^{x-1}$$

$$\mu = \frac{1}{p}$$
$$\sigma^2 = \frac{q}{p^2}$$

4. Negative Binomial Distribution

$$P(X = x) = x - 1C_{r-1}p^r(1-p)^{x-r}$$

$$\mu = \frac{r}{p}$$
$$\sigma^2 = \frac{rq}{p^2}$$

5. Poisson Distribution

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$$\mu = \lambda$$
$$\sigma^2 = \lambda$$

Laboratory 5: Random Number generator using LCG and MCG

1. Introduction and Purpose of Experiment

2. Aim and Objectives

3. Experimental Procedure

a. Develop and implement a Java program to generate pseudorandom numbers based on the linear congruential random number generator to produce a sequence of 20 integers, $X1, X2$, ... between 0 and $m - 1$ by following a recursive relationship: Use $X_0 = 27, a = 17, c = 43$, and $m = 100$.

b. Modify the above program for multiplicative congruential method to determine the period of the generator for $a = 13, m = 2^6$, and $X_0 = 1, 2, 3, 4$.

Satisfy the following property of max period (P):

i. Generate random numbers with longest possible period is $P = m = 2^b$

ii. Generate random numbers with longest possible period is $P = m / 4 = 2^{b-2}$

4. Algorithms

5. Presentation of Results

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab05;

import java.util.ArrayList;
import java.util.Arrays;

/**
 *
 * @author shadowleaf
 */
public class Lab05 {

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        {
            // X_n+1 = (aX_n + c) mod m
            // X_0 = 27
            // a = 17
            // c = 43
            // m = 100
            // n = 20

            System.out.println("LCG ----------\n");

            Integer a = 17, c = 43, m = 100, n = 20;

            ArrayList<Integer> X = new ArrayList<>();
            X.add(27);

            System.out.printf("X0 = %d, a = %d, c = %d, m = %d, n = %d\n", X.get(0), a, c
, m, n);

            for (int i = 0; i < n; i++) {
                X.add((a * X.get(i) + c) % m);
            }

            Integer i = 0;
            for (Integer e : X) {
                System.out.printf("X%d = %d, ", i++, e);
            }
            System.out.println("\n");
        }

        {
            System.out.println("MCG ----------\n");
            // a = 13
            // m = 2^6
            // X_0 = 1, 2, 3, 4
            Integer a = 13;
            Integer m = 1 << 6;
            // important
            ArrayList<Integer> C = new ArrayList(Arrays.asList(17, 0));

            for (Integer c : C) {

            ArrayList<ArrayList<Integer>> X = new ArrayList<>();
            X.add(new ArrayList(Arrays.asList(1))); // X0 = 1
            X.add(new ArrayList(Arrays.asList(2))); // X0 = 2
            X.add(new ArrayList(Arrays.asList(3))); // X0 = 3
            X.add(new ArrayList(Arrays.asList(4))); // X0 = 4

                for (int i = 0; i < X.size(); i++) {
                    Integer X0 = X.get(i).get(0);
                    // if X0 is even c should not be zero
                    if ( (X0 % 2 == 0) && (c == 0) ) {
                        continue;
                    }
                    System.out.printf("X0 = %d, a = %d, c = %d, m = %d\n", X0, a, c, m);
                    Integer j = 0;
                    while (true) {
                        Integer Xnew = (a * X.get(i).get(j++) + c) % m;
                        X.get(i).add(Xnew);
                        if (Xnew.equals(X0)) {
                            System.out.printf("P = %d\n", j);
                            Integer k = 0;
                            for (Integer e : X.get(i)) {
                                System.out.printf("X%d = %d, ", k++, e);
                            }
                            System.out.println("\n");
                            break;
                        }
                    }
                }
            }
        }
    }

}
```

6. Analysis and Discussions

```
run:
LCG ----------

X0 = 27, a = 17, c = 43, m = 100, n = 20
X0 = 27, X1 = 2, X2 = 77, X3 = 52, X4 = 27, X5 = 2, X6 = 77, X7 = 52, X8 = 27,
X9 = 2, X10 = 77, X11 = 52, X12 = 27, X13 = 2, X14 = 77, X15 = 52, X16 = 27,
X17 = 2, X18 = 77, X19 = 52, X20 = 27,

MCG ----------

X0 = 1, a = 13, c = 17, m = 64
P = 64
```

```
X0 = 1, X1 = 30, X2 = 23, X3 = 60, X4 = 29, X5 = 10, X6 = 19, X7 = 8, X8 = 57,
X9 = 54, X10 = 15, X11 = 20, X12 = 21, X13 = 34, X14 = 11, X15 = 32, X16 = 49,
X17 = 14, X18 = 7, X19 = 44, X20 = 13, X21 = 58, X22 = 3, X23 = 56, X24 = 41,
X25 = 38, X26 = 63, X27 = 4, X28 = 5, X29 = 18, X30 = 59, X31 = 16, X32 = 33,
X33 = 62, X34 = 55, X35 = 28, X36 = 61, X37 = 42, X38 = 51, X39 = 40, X40 = 25,
X41 = 22, X42 = 47, X43 = 52, X44 = 53, X45 = 2, X46 = 43, X47 = 0, X48 = 17,
X49 = 46, X50 = 39, X51 = 12, X52 = 45, X53 = 26, X54 = 35, X55 = 24, X56 = 9,
X57 = 6, X58 = 31, X59 = 36, X60 = 37, X61 = 50, X62 = 27, X63 = 48, X64 = 1,

X0 = 2, a = 13, c = 17, m = 64
P = 64
X0 = 2, X1 = 43, X2 = 0, X3 = 17, X4 = 46, X5 = 39, X6 = 12, X7 = 45, X8 = 26,
X9 = 35, X10 = 24, X11 = 9, X12 = 6, X13 = 31, X14 = 36, X15 = 37, X16 = 50,
X17 = 27, X18 = 48, X19 = 1, X20 = 30, X21 = 23, X22 = 60, X23 = 29, X24 = 10,
X25 = 19, X26 = 8, X27 = 57, X28 = 54, X29 = 15, X30 = 20, X31 = 21, X32 = 34,
X33 = 11, X34 = 32, X35 = 49, X36 = 14, X37 = 7, X38 = 44, X39 = 13, X40 = 58,
X41 = 3, X42 = 56, X43 = 41, X44 = 38, X45 = 63, X46 = 4, X47 = 5, X48 = 18,
X49 = 59, X50 = 16, X51 = 33, X52 = 62, X53 = 55, X54 = 28, X55 = 61, X56 = 42,
X57 = 51, X58 = 40, X59 = 25, X60 = 22, X61 = 47, X62 = 52, X63 = 53, X64 = 2,

X0 = 3, a = 13, c = 17, m = 64
P = 64
X0 = 3, X1 = 56, X2 = 41, X3 = 38, X4 = 63, X5 = 4, X6 = 5, X7 = 18, X8 = 59,
X9 = 16, X10 = 33, X11 = 62, X12 = 55, X13 = 28, X14 = 61, X15 = 42, X16 = 51,
X17 = 40, X18 = 25, X19 = 22, X20 = 47, X21 = 52, X22 = 53, X23 = 2, X24 = 43,
X25 = 0, X26 = 17, X27 = 46, X28 = 39, X29 = 12, X30 = 45, X31 = 26, X32 = 35,
X33 = 24, X34 = 9, X35 = 6, X36 = 31, X37 = 36, X38 = 37, X39 = 50, X40 = 27,
X41 = 48, X42 = 1, X43 = 30, X44 = 23, X45 = 60, X46 = 29, X47 = 10, X48 = 19,
X49 = 8, X50 = 57, X51 = 54, X52 = 15, X53 = 20, X54 = 21, X55 = 34, X56 = 11,
X57 = 32, X58 = 49, X59 = 14, X60 = 7, X61 = 44, X62 = 13, X63 = 58, X64 = 3,

X0 = 4, a = 13, c = 17, m = 64
P = 64
X0 = 4, X1 = 5, X2 = 18, X3 = 59, X4 = 16, X5 = 33, X6 = 62, X7 = 55, X8 = 28,
X9 = 61, X10 = 42, X11 = 51, X12 = 40, X13 = 25, X14 = 22, X15 = 47, X16 = 52,
X17 = 53, X18 = 2, X19 = 43, X20 = 0, X21 = 17, X22 = 46, X23 = 39, X24 = 12,
X25 = 45, X26 = 26, X27 = 35, X28 = 24, X29 = 9, X30 = 6, X31 = 31, X32 = 36,
X33 = 37, X34 = 50, X35 = 27, X36 = 48, X37 = 1, X38 = 30, X39 = 23, X40 = 60,
X41 = 29, X42 = 10, X43 = 19, X44 = 8, X45 = 57, X46 = 54, X47 = 15, X48 = 20,
X49 = 21, X50 = 34, X51 = 11, X52 = 32, X53 = 49, X54 = 14, X55 = 7, X56 = 44,
X57 = 13, X58 = 58, X59 = 3, X60 = 56, X61 = 41, X62 = 38, X63 = 63, X64 = 4,

X0 = 1, a = 13, c = 0, m = 64
P = 16
X0 = 1, X1 = 13, X2 = 41, X3 = 21, X4 = 17, X5 = 29, X6 = 57, X7 = 37, X8 = 33,
X9 = 45, X10 = 9, X11 = 53, X12 = 49, X13 = 61, X14 = 25, X15 = 5, X16 = 1,

X0 = 3, a = 13, c = 0, m = 64
P = 16
X0 = 3, X1 = 39, X2 = 59, X3 = 63, X4 = 51, X5 = 23, X6 = 43, X7 = 47, X8 = 35,
X9 = 7, X10 = 27, X11 = 31, X12 = 19, X13 = 55, X14 = 11, X15 = 15, X16 = 3,
```

7. Conclusions

The max period(P) is:

1 For m a power of 2, say $m = 2^b$, and $c \neq 0$, the longest possible period is $P = m = 2^b$, which is achieved provided that c is relatively prime to m (that is, the greatest common factor of c and m is 1), and a = 1 + 4k, where k is an integer.

2 For m a power of 2, say $m = 2^b$, and c = 0, the longest possible period is $P = m / 4 = 2^{b-2}$, which is achieved provided that the seed X_0 is odd and the multiplier, a, is given by a = 3 + 8k or a = 5 + 8k, for some k = 0, 1,...

3 For m a prime number and c = 0, the longest possible period is P = m - 1, which is achieved provided that the multiplier, a, has the property that the smallest integer k such that $a^k - 1$ is divisible by m is k = m - 1,

Laboratory 6: Random Variate Generator using Inverse-Transfonn Technique

1. Introduction and Purpose of Experiment

2. Aim and Objectives

3. Experimental Procedure

   Design and implement a Java program to determine a sequence of 10 *random variates* X *by generating a sequence of random numbers* R *using the flowing distributions:*

   i. Uniform distribution in the interval [10, 20]
   ii. Exponential distribution with mean value 5
   iii. Normal distribution with mean 3.1 and sigma 0.6

4. Algorithms

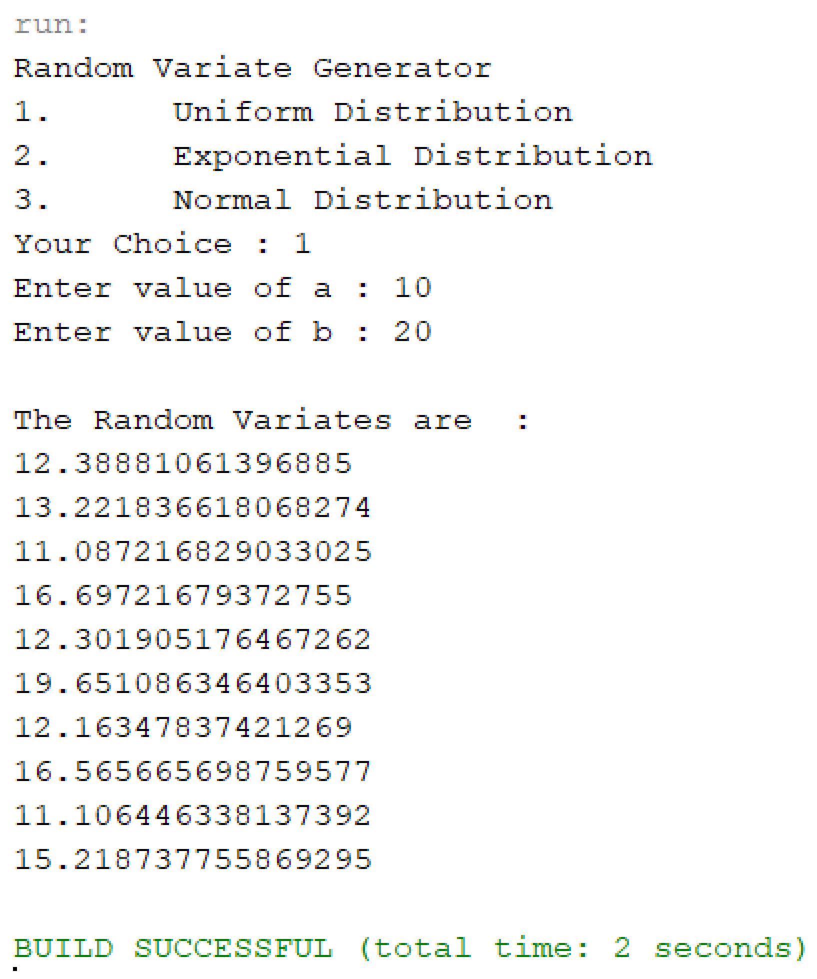
5. Presentation of Results

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab06;

import java.util.Random;
import java.util.Scanner;

/**
 *
 * @author shadowleaf
 */
public class Lab06 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Random random = new Random();
```

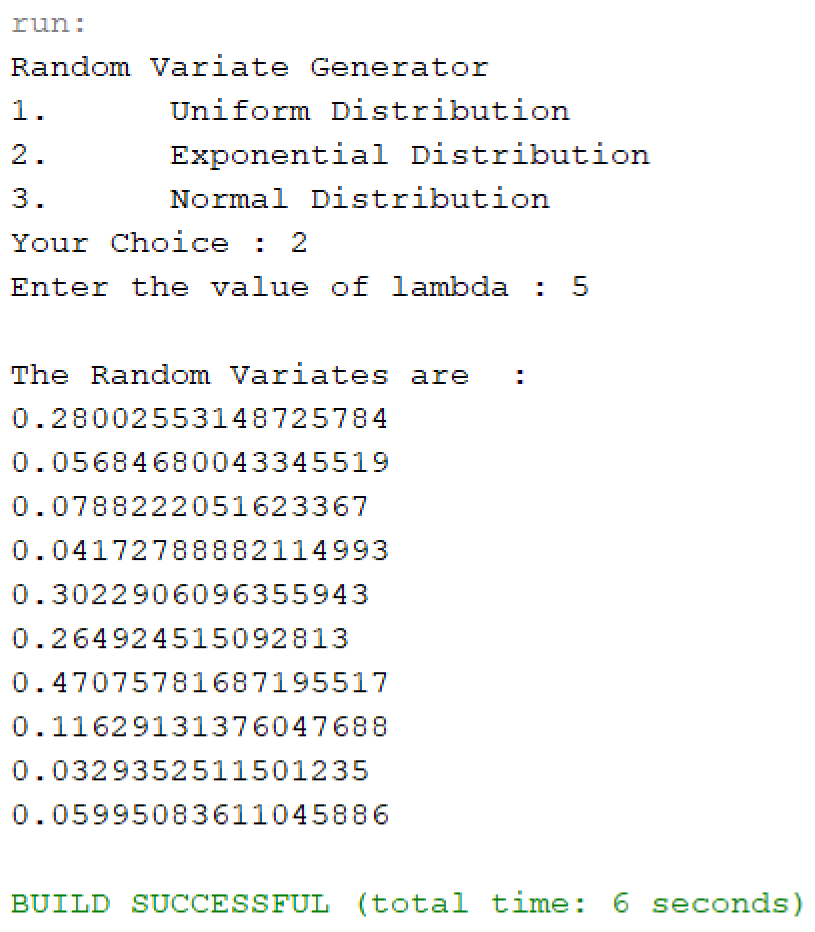
```
int N = 10;
double []R = new double[N];
double []X = new double[N];

for (int i = 0 ; i < R.length ; i++) {
    R[i] = random.nextDouble();
}

boolean done = false;

System.out.print("Random Variate Generator\n"
        + "1.\tUniform Distribution\n"
        + "2.\tExponential Distribution\n"
        + "3.\tNormal Distribution\n"
        + "Your Choice : ");
switch(input.nextInt()) {
    case 1: {
        int a, b;
        System.out.print("Enter value of a : ");
        a = input.nextInt();
        System.out.print("Enter value of b : ");
        b = input.nextInt();

        for (int i = 0 ; i < X.length ; i++) {
            X[i] = a + (b-a) * R[i];
        }

        done = true;
    }
        break;
    case 2: {
        double lambda;
        System.out.print("Enter the value of lambda : ");
        lambda = input.nextDouble();
        for (int i = 0 ; i < X.length ; i++) {
            X[i] = Math.log(R[i]) / (-lambda);
        }

        done = true;
    }
        break;
    case 3: {
        double meu, sigma;
        System.out.print("Enter the value of meu : ");
        meu = input.nextDouble();
```

```
                System.out.print("Enter the value of sigma : ");
                sigma = input.nextDouble();
                for (int i = 0 ; i < X.length ; i++) {
                    X[i] = (Math.pow(R[i], 0.135) - Math.pow(1-R[i], 0.135) ) / 0.1975;
                    X[i] = X[i] * sigma + meu;
                }

                done = true;
            }
                break;
        }

        if (done) {
            System.out.println("\nThe Random Variates are  : ");
            for (int i = 0 ; i < X.length ; i++) {
                System.out.print(X[i]+"\n");
            }
            System.out.println();
        }

    }

}
```

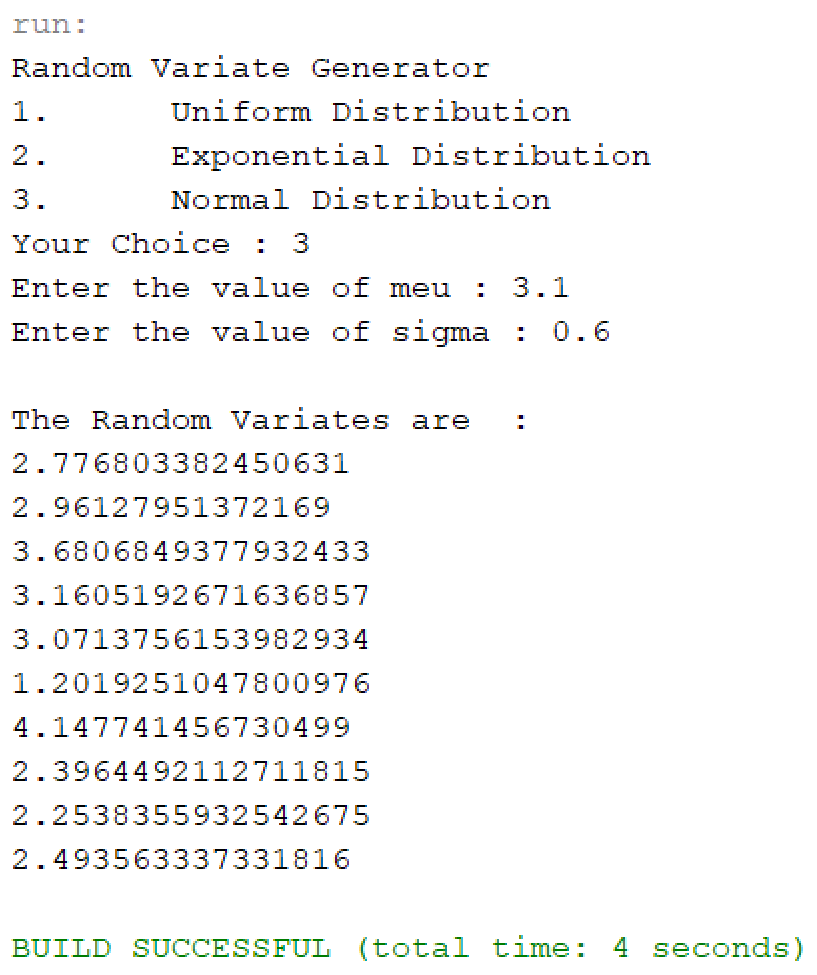
6. Analysis and Discussions



```
run:
Random Variate Generator
1.        Uniform Distribution
2.        Exponential Distribution
3.        Normal Distribution
Your Choice : 1
Enter value of a : 10
Enter value of b : 20

The Random Variates are  :
12.38881061396885
13.221836618068274
11.087216829033025
16.69721679372755
12.301905176467262
19.651086346403353
12.16347837421269
16.565665698759577
11.106446338137392
15.218737755869295

BUILD SUCCESSFUL (total time: 2 seconds)
```



```
run:
Random Variate Generator
1.      Uniform Distribution
2.      Exponential Distribution
3.      Normal Distribution
Your Choice : 2
Enter the value of lambda : 5

The Random Variates are  :
0.28002553148725784
0.05684680043345519
0.078822205162336 7
0.04172788882114993
0.3022906096355943
0.264924515092813
0.47075781687195517
0.11629131376047688
0.0329352511501235
0.05995083611045886

BUILD SUCCESSFUL (total time: 6 seconds)
```



```
run:
Random Variate Generator
1.      Uniform Distribution
2.      Exponential Distribution
3.      Normal Distribution
Your Choice : 3
Enter the value of meu : 3.1
Enter the value of sigma : 0.6

The Random Variates are  :
2.776803382450631
2.96127951372169
3.6806849377932433
3.1605192671636857
3.0713756153982934
1.2019251047800976
4.147741456730499
2.3964492112711815
2.2538355932542675
2.493563337331816

BUILD SUCCESSFUL (total time: 4 seconds)
```

7. Conclusions

Standard Normal Distribution

The inverse CDF of the standard normal distribution is approximated as:

$$X = F^{-1}(R) = \frac{R^{0.135} - (1-R)^{0.135}}{0.1975}$$

Example usage for $X \sim Nor(\mu, \sigma^2)$, $Z \sim Nor(0,1)$

Then take $X \leftarrow \mu + \sigma Z$

Taking example of $X \sim Nor(3, 16)$ and $R = 0.59$

Then

$$X = \mu + \sigma Z = 3 + 4\Phi^{-1}(0.59) = 3 + 4(0.2275) = 3.91$$

Exponential Distribution

$$X = -\frac{\ln(1-R)}{\lambda}$$

Or

$$X = -\frac{\ln(R)}{\lambda}$$

Uniform Distribution

$$X = a + (b-a)R$$

Laboratory 7: Tests for Random Numbers using Frequency Tests

1. Introduction and Purpose of Experiment

2. Aim and Objectives

3. Experimental Procedure

K-S Test

Design and implement a Java program to test the generated random numbers 0.44, 0.81, 0.14, 0.05, 0.93 for uniformity by using the Kolmogorov-Smirnov test with the level of significance α= 0.10

Chi-Square test

A public opinion poll surveyed a random sample of 1000 voters. Respondents were classified by gender (male or female) and by voting preference (BJP, Congress and AAP). Results are shown below:

	Programing language Preferences			Row total
	BJP	Congress	AAP	
Male	200	150	50	400
Female	250	300	50	600
Column total	450	450	100	1000

Design and implement a Java program to conduct chi-square test with $\alpha = 0.05$ level of significance and determine if there is a gender gap. Identify whether the men's preferences differ significantly from the women's preferences.

4. Algorithms

```
KS-Test

-> Rank the N random numbers in ascending order.

-> Calculate D+ as max(i/N-Ri) for all i in(1, N)

-> Calculate D- as max(Ri-((i-1)/N)) for all i in(1, N)

-> Calculate D as max(D+, D-)

-> If D>D(alpha)

    Rejects Uniformity

   else
```

It fails to reject the Null Hypothesis.

Chi-Square Test

First calculate the expected frequencies for the groups, then determining whether the division of the groups, called the observed frequencies, matches the expected frequencies.

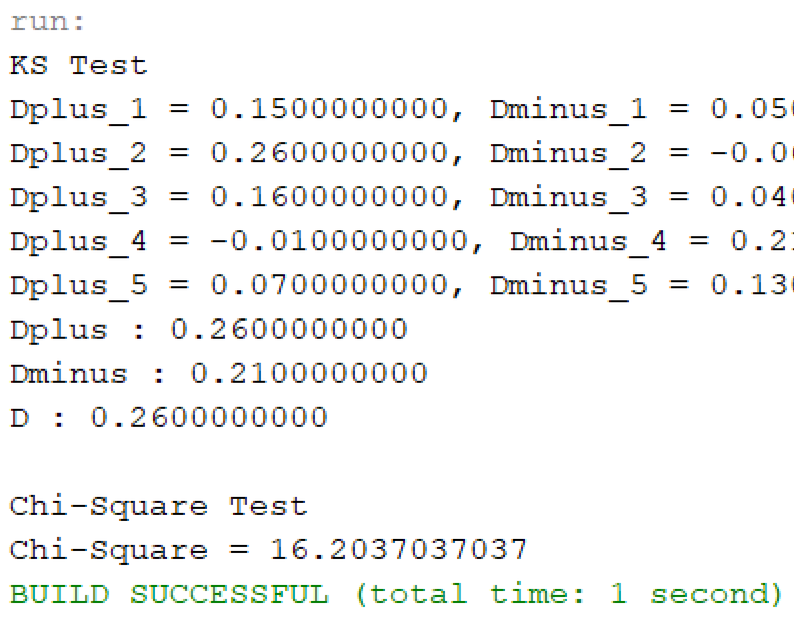
The result of the test is a test statistic that has a chi-squared distribution and can be interpreted to reject or fail to reject the assumption or null hypothesis that the observed and expected frequencies are the same.

5. Presentation of Results

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package lab07;

import java.util.Arrays;

/**
 *
 * @author shadowleaf
 */
public class Lab07 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        {
            System.out.println("KS Test");

            Double[] rn = {0.44, 0.81, 0.14, 0.05, 0.93};
            Arrays.sort(rn);
            Integer N = rn.length;

            Double Dplus = Double.MIN_VALUE;
            Double Dminus = Double.MIN_VALUE;

            Double i = 1.0;
            for (int idx = 0; idx < N; idx++) {
                Double newDplus = ((i / N) - rn[idx]);
                Double newDminus = rn[idx] - ((i - 1) / N);

                Dplus = Math.max(Dplus, newDplus);
                Dminus = Math.max(Dminus, newDminus);

//              System.out.printf("newDplus = %.10f, newDminus = %.10f\n", newDplus, newDmi
nus);
                System.out.printf("Dplus_%d = %.10f, Dminus_%d = %.10f\n", idx + 1, newDp
lus, idx + 1, newDminus);

                i += 1.0;
            }

            Double D = Math.max(Dplus, Dminus);

            System.out.printf("Dplus : %.10f\nDminus : %.10f\nD : %.10f\n", Dplus, Dminus
, D);

            Double alpha = 0.05;
        }

        {
            System.out.println("\nChi-Square Test");

            Integer[][] data_obs = {{200, 150, 50}, {250, 300, 50}};

            Integer nrows = data_obs.length;
            Integer ncols = data_obs[0].length;

            Integer[] col_total = new Integer[ncols];
            Integer[] row_total = new Integer[nrows];

            Integer total = 0;

            int row, col;

            // calculate col totals
            for (col = 0; col < col_total.length; col++) {
                    col_total[col] = 0;
                    for (row = 0; row < nrows; row++) {
                        col_total[col] += data_obs[row][col];
                    }

                    total += col_total[col];
                }

                // calculate row totals
                for (row = 0; row < row_total.length; row++) {
                    row_total[row] = 0;
                    for (col = 0; col < ncols; col++) {
                        row_total[row] += data_obs[row][col];
                    }
                }

//              System.out.println(Arrays.toString(row_total));
//              System.out.println(Arrays.toString(col_total));
//              System.out.println(total);
                Double[][] data_exp = new Double[nrows][ncols];

                // calculate expected values
                for (row = 0; row < nrows; row++) {
                    for (col = 0; col < ncols; col++) {
                        data_exp[row][col] = col_total[col] * row_total[row] / (double) total
;
//                      System.out.printf("%.5f ", data_exp[row][col]);
                    }
//                  System.out.println();
                }

                Double chi_sqr = 0.0;

                // calculate chi square
                for (row = 0; row < nrows; row++) {
                    for (col = 0; col < ncols; col++) {
                        Double Oi = (double) data_obs[row][col];
                        Double Ei = data_exp[row][col];
                        chi_sqr += (Oi - Ei) * (Oi - Ei) / Ei;
                    }
                }

                System.out.printf("Chi-Square = %.10f\n", chi_sqr);

                Integer dfree = (nrows - 1) * (ncols - 1);
```

```
            Double alpha = 0.05;

        }
    }

}
```

6. Analysis and Discussions



```
run:
KS Test
Dplus_1 = 0.1500000000, Dminus_1 = 0.0500000000
Dplus_2 = 0.2600000000, Dminus_2 = -0.0600000000
Dplus_3 = 0.1600000000, Dminus_3 = 0.0400000000
Dplus_4 = -0.0100000000, Dminus_4 = 0.2100000000
Dplus_5 = 0.0700000000, Dminus_5 = 0.1300000000
Dplus : 0.2600000000
Dminus : 0.2100000000
D : 0.2600000000

Chi-Square Test
Chi-Square = 16.2037037037
BUILD SUCCESSFUL (total time: 1 second)
```

7. Conclusions

KS-Test

Kolmogorov–Smirnov test a very efficient way to determine if two samples are significantly different from each other. It is usually used to check the uniformity of random numbers. Uniformity is one of the most important properties of any random number generator and Kolmogorov–Smirnov test can be used to test it.

The Kolmogorov–Smirnov test may also be used to test whether two underlying one-dimensional probability distributions differ. It is a very efficient way to determine if two samples are significantly different from each other.

The Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples.

H0(Null Hypothesis): Null hypothesis assumes that the numbers are uniformly distributed between 0-1. If we are able to reject the Null Hypothesis, this means that the numbers are not uniformly distributed between 0-1. Failure to reject the Null Hypothesis although does not necessarily mean that the numbers follow the uniform distribution.

Chi-square Test

Chi-square Test for Feature Extraction:

Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi-square scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population.

$$X^2 = \frac{(Observed\ frequency - Expected\ frequency)^2}{Expected\ frequency}$$