

# CS-259

## Data Structures with Java

*Random Numbers and Roulette*



Instructor: **Joel Castellanos**

e-mail: [joel@unm.edu](mailto:joel@unm.edu)

web: <http://cs.unm.edu/~joel/>

Course website:

<http://cs.unm.edu/~joel/cs259/>



1

## Project 1: Roulette Project

- Due Sunday, Sept 6 at midnight.
- Write a Java program that simulates many games of Roulette using a given betting strategy.
- The program must input three numbers:
  - **amountStart**: The amount of dollars the player starts with.
  - **maxBet**: The maximum amount of a single bet.
  - **numBets**: The total number of bets to be made.
- The program must output the player's final balance.
- You must modify the strategy in some creative way.
- You must give a 5 minute, slide presentation in class on Monday analyzing the results of the given method and of your method.

-2-

## The Game of Roulette



-3-

## The Game of Roulette

Roulette is a casino gambling game.

In the game, a croupier spins a wheel in one direction, then spins a ball in the opposite direction around a tilted circular surface running around the circumference of the wheel.

The ball eventually falls on to the wheel and into one of 38 colored and numbered pockets on the wheel.

Players place bets on the winning number, the color of the pocket, whether the number is odd or even, etc. The payout odds for each type of bet is based on its probability. The casino usually imposes minimum and maximum bets.

Of the 38 colored and numbered pockets on the wheel, 18 are black, 18 are red and 2 are green. The payout for betting on black or on red is 1 to 1.

-4-

## Winning Heuristic?

1. Start with a bet of one dollar on **black**.
2. Whenever **black** is rolled, collect the payout and return to a one dollar bet on **black** for the next roll.
3. Whenever **black** is not rolled, either
  - bet on **black** again at double the pervious bet, or
  - If that doubled bet exceeds the house maximum, then return to a bet of one dollar on **black**.
4. The simulation terminates after the user specified number of rolls.

-5-

## Winning Heuristic - example

Starting Balance: \$1000.00

Bet	Result	Balance
\$1.00	not black	1000 - 1 = 999
\$2.00	not black	999 - 2 = 997
\$4.00	not black	997 - 4 = 993
\$8.00	black	993 + 8 = 1001
\$1.00	black	1001 + 1 = 1002
\$1.00	not black	1002 - 1 = 1001
\$2.00	not black	1001 - 2 = 999
\$4.00	black	999 + 4 = 1003
\$1.00	not black	1003 - 1 = 1002
\$2.00	black	1002 + 2 = 1004

-6-

## Roulette Project - Interpret your results

- Is the given scheme good to take to the casino?
  - If yes, why does it work, and why do you believe it?
  - If not, what is the flaw?
- How sensitive are these results to the initial conditions?
  - What are *good* initial conditions?
- Get Creative:
  - Modify the given heuristic
  - Invent or research a different heuristic.
  - Implement that heuristic and interpret the results.

-7-

## Roulette Grading Rubric

- Java Code (RouletteStandard.java in WebCT)
  - +15 Correctness of code.
  - +7 Readability and organization.
  - +7 Coding standards (including comments).
- Java Code (RouletteXXXX.java in WebCT)
  - +12 Correctness of code.
  - +7 Readability and organization.
  - +7 Coding standards (including comments).
- Presentation (oral + .pdf file in WebCT)
  - +10 well organized and fits within 5 minutes.
  - +25 Correctness of Conclusions.
  - +10 written and oral explanation is clear.

-8-

## java.util.Random

### Constructor: `Random()`

Creates a new random number generator. Its seed is initialized to a value based on the current time. Two `Random` objects created within the same millisecond will have the same sequence of random numbers.

### `double nextDouble()`

Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.

### `int nextInt(int n)`

Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

-9-

## Example of java.util.Random

```
import java.util.Random;
public class HelloWorldClass
{ public static void main(String[] args)
  {
    Random rand = new Random();
    for (int i=0; i<25; i++)
    { int x = rand.nextInt(6);
      double y = rand.nextDouble();
      System.out.println(i+" " +
                          x + ", " + y);
    }
  }
}
```

```
0) 5, 0.6159209125196468
1) 0, 0.08301268243756299
2) 5, 0.620922404525617
...
```

-10-

## Histogram of a Six-Sided Die

```
public static void main(String[] args)
{
    Random rand = new Random();
    int[] histogram = new int[7];

    for (int i=0; i<100; i++)
    {
        int a = rand.nextInt(6)+1;
        histogram[a]++;
    }

    for (int i=1; i<histogram.length; i++)
    {
        System.out.print(i+" ");
        for (int k=1; k<=histogram[i]; k+=1)
        {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

```
1) *****
2) *****
3) *****
4) *****
5) *****
6) *****
```

-11-

## Histogram of a Pair of Six-Sided Dice

```
public static void main(String[] args)
{
    Random rand = new Random();
    int[] histogram = new int[13];

    for (int i=0; i<500; i++)
    {
        int a = rand.nextInt(6)+1;
        int b = rand.nextInt(6)+1;
        histogram[a+b]++;
    }

    for (int i=2; i<histogram.length; i++)
    {
        for (int k=1; k<=histogram[i]; k+=1)
        {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

-12-

## Homework: Due Friday, Sept 11



Read Chapter 4, Objects and Classes through page 152:

- Introduction to Object-Oriented Programming
  - Using Predefined Classes
  - Defining Your Own Classes
  - Static Fields and Methods
  - Method Parameters
  - Object Construction
- There will be quiz questions on the reading.

-13-

## Quiz: Pair of Dice Histogram

```
1. public static void main(String[] args)
2. { Random r = new Random();
3.   int[] hist = new int[13];
4.
5.   for (int i=0; i<500; i++)
6.     { int a = r.nextInt(6)+1;
7.       int b = r.nextInt(6)+1;
8.       hist[i]++;
9.     }
10.
11.  for (int k=2; k<hist.length; k++)
12.    { for (int i=0; i<hist[k]; i+=1)
13.      { System.out.print("#");
14.        }
15.      System.out.println();
16.    }
17. }
```

This code stop at a runtime error in line:

- a) 5
- b) 6
- c) 8
- d) 12
- e) 13

-14-