Random Number Generation

CSCI1130

Outline

Assignment 1 summary

Random number generation

Assignment 1 summary

Mean: 97

Common mistakes:

- Wrong zip file, project, package, file name
- No comments, or only one or two lines of comments
- Bad indentation
- No declaration, or wrong personal information in the declaration part
- Only show the first letter of the surname
- The output does not match the surname of the student
- Runtime exception, compile error
- Do not customize showMyName function

Appeal: email to Grace at ltang@cse.cuhk.edu.hk by 13-Oct-2018 (This Saturday)

(Pseudo) Random Number Generation

A). Math.random()

B). Using class Random

Math.random()

Parameter

- None

Return

- double: [0.0, 1.0) ; EXCLUDING 1.0
- Can be considered as a probability value

To generate a random number within [m, n)

- Apply a scaling factor of (n-m) with an offset value m

```
double m = -2.6, n = +8.3;
double value = Math.random() * (n-m) + m;
```

Class Random

import java.util.Random;

- An instance of this class can be used for generating a stream of pseudo-random numbers.

Constructor Random( )

- For creating a new random number generator object with auto-seeding.

Overloaded Constructor Random(long seed)

- For creating a new random number generator object with a long integer-type parameter seed.

Random Seed

A **random seed** is a number used for initializing a pseudo-random number generator. The seed governs the behavior of the PRNG.

PRNG objects created with the same random seed will produce identical pseudo-random number sequences.

Random Seed (Example)

```
import java.util.Random;
...
int seed = 11;
Random rngObj1 = new Random(seed);
Random rngObj2 = new Random(seed);
Random rngObj3 = new Random( );
System.out.printf("%12d\n",
rngObj1.nextInt() );
System.out.printf("%.2f\n",
rngObj1.nextFloat() );
System.out.printf("%12d\n",
rngObj2.nextInt() );
System.out.printf("%.2f\n",
rngObj2.nextFloat() );
System.out.printf("%12d\n",
rngObj3.nextInt() );
System.out.printf("%.2f\n",
rngObj3.nextFloat() );
```

```
-1158177819
0.71
-1158177819
0.71
1856327341
0.34
```

Demo by TA

TA is going to do a set of demonstrations on NetBeans NOW

You may also download RNGExample.java and try yourselves.

Random Seed Candidate

Using system time:

```
long value = System.currentTimeMillis();
```

It returns the current system time in milliseconds, which is a 64-bit long integer.

It is the time elapsed since midnight, Jan 1, 1970 UTC, e.g.

```
Current system time is  1538720316328ms on
2018.10.05 at 14:18:36 HKT
```

```
Date now = new Date();
String thisMoment = new SimpleDateFormat(
          "yyyy.MM.dd 'at' HH:mm:ss z").format(now);
System.out.println("Current system time is " +
      System.currentTimeMillis() + "ms on " + thisMoment);
```

Exercise: Allow/ Avoid Duplication

How to allow or avoid duplications between 3 random numbers?

If we allow duplication,

- Simply generate 3 random numbers independently

If we need avoid duplication,

- Perform comparisons; reject ties and re-generate

Exercise: Avoid Duplication

Draft your solution.

Communicate your idea with peers.

Share with the class.

Methods

Random rngObj = new Random(20161001);

result = rngObj.nextInt();

- Get the next pseudorandom, uniformly distributed int value from this random number generator object's sequence

result = rngObj.nextInt(10);

- Get the next pseudorandom, uniformly distributed int value within 0 to 9 (note that 10 is excluded.)

Methods

Random rngObj = new Random(20161001);

result = rngObj.nextDouble();

- Get the next pseudorandom, uniformly distributed double value between 0.0 (inclusive) and 1.0 (exclusive)

result = rngObj.nextGaussian();

- Returns the next pseudorandom, Gaussian ("normally") distributed double value with mean 0.0 and standard deviation 1.0.

Output Range Control

nextInt() returns an int in $[-2^{31}, +2^{31})$

nextInt(int bound) returns an int in [0, bound-1]

- excluding the upper bound value
- e.g.

[a, b]	rngObj.nextInt(b - a + 1) + a;
(a, b]	rngObj.nextInt(b - a) + a + 1;
[a, b)	rngObj.nextInt(b - a) + a;
(a, b)	rngObj.nextInt(b - a - 1) + a + 1 ;

Output Range Control (Examples)

[-10, 15]

```
rngObj.nextInt(26) - 10;
```

(-10, 15]

```
rngObj.nextInt(25) - 9;
```

[10, 15)

```
rngObj.nextInt(5) + 10;
```

(10, 15)

```
rngObj.nextInt(4) + 11;
```

Application: Mark Six

How to generate 6 *unique* random numbers in the range of [1 – 49]?

Discussion NOW

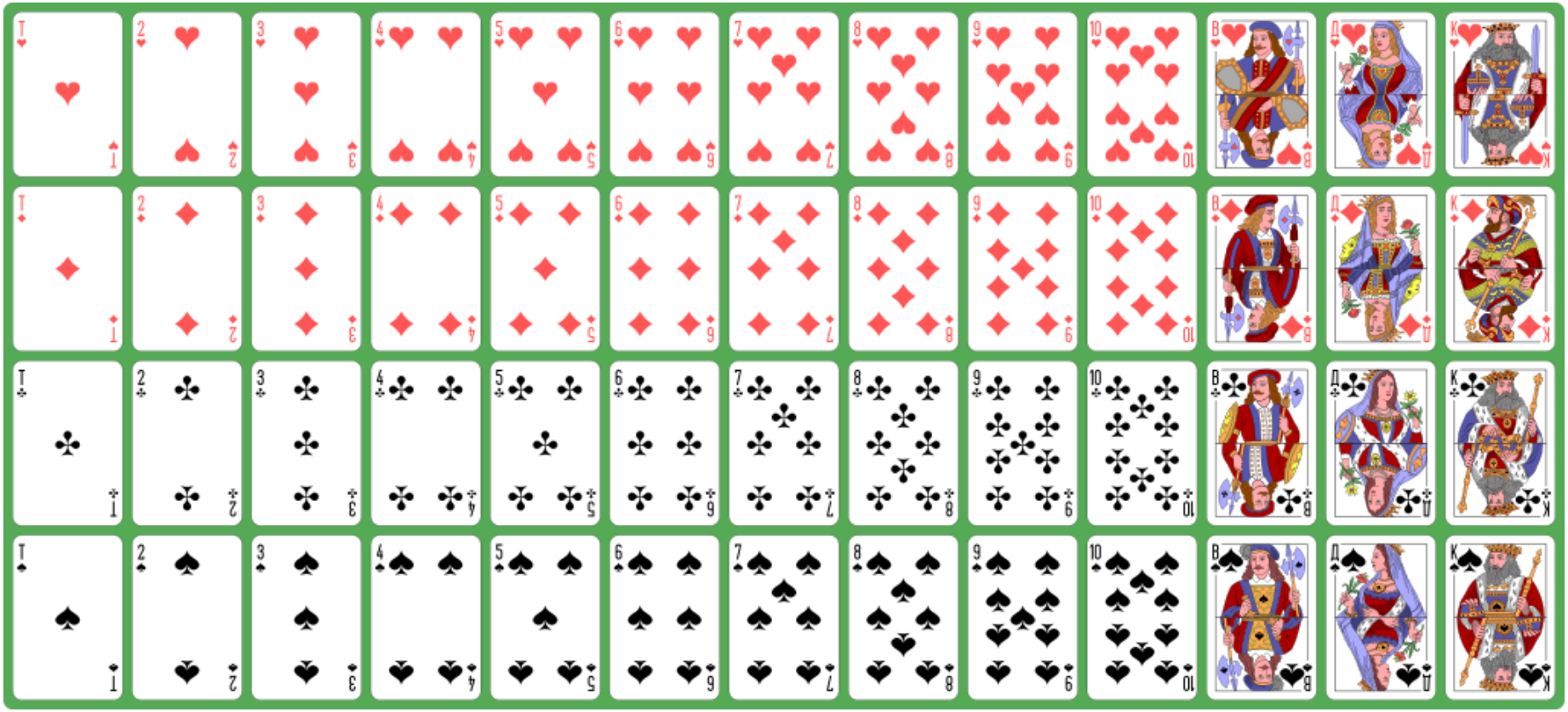
Draft your solution.

Communicate your idea with peers.

Share with the class.

Application: Shuffling

How to shuffle a deck of playing cards randomly?



Source: https://commons.wikimedia.org/wiki/File:Atlasnye_playing_cards_deck.svg

Discussion NOW

Draft your solution.

Communicate your idea with peers.

Share with the class.

Further Reading (optional)

Ziggurat algorithm

- https://en.wikipedia.org/wiki/Ziggurat_algorithm

Box–Muller transform

- https://en.wikipedia.org/wiki/Box–Muller_transform

END