CSCI1530 Computer Principles and Java Programming

Tutorial 5
Generate random numbers

Zheng Qingqing
SHB 911
qqzheng@cse.cuhk.edu.hk

Content

- Generating random numbers in Java
- Test exercise
- Q&A for Assignment 2

Generate Random Numbers in Java

Why random?

There are so many targets for random generation:

- In games– to imitate dice, luck, scene weather
- Randomly decide teams
- Decide who to get takeaway food
- Make random selections: which restaurant to go
- Imitate probability, in some algorithms

Generate random numbers in Java

- Use pre-defined Math class methods
  Math.random()

- public static double random()
  - **Parameter:** null
  - **Return value:** a double value
  - **Functionality:** Return a random value in [0,1), Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

Generate random numbers in Java

- Or Use pre-defined Random object methods
  - nextInt()
  - nextInt(int bound)
- public int nextInt()
  - **Functionality:** Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence.
- public int nextInt(int bound)
  - **Functionality:** Returns a pseudorandom, uniformly distributed int value between zero (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Usage

```
// Random Object-based usage
import java.util.*; // beginning of the program
Random obj;
obj = new Random();
int i = obj.nextInt(); // one of +/-2.1 billion
int j = obj.nextInt(50); // [0, 49]; 50 excluded

// Math Class-based usage; import not needed
double a;
a = Math.random(); // [0.0, 1.0)
```

Examples -- run 3 times

generate random number in [0.0,1.0)

```java
package javaapplication1;
/**
 *
 * @author Zheng Qingqing
 */
public class JavaApplication1 {

    public static void main(String[] args) {
        // TODO code application logic here
        double a;
        double b;
        a = Math.random();
        b = Math.random();
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }

}
```

```
Output - JavaApplication1 (run)
run:
a = 0.7793515280752702
b = 0.9087157053616814
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
Output - JavaApplication1 (run)
run:
a = 0.1590763459200195
b = 0.11329805345051425
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
Output - JavaApplication1 (run)
run:
a = 0.0783087434359453
b = 0.9433609224723921
BUILD SUCCESSFUL (total time: 0 seconds)
```

Different return each time!

Examples -- run 3 times

generate random integer in $[-2^{32}, 2^{32}-1)$

```java
package javaapplication1;
import java.util.*;
/**
 *
 * @author Zheng Qingqing
 */
public class JavaApplication1 {

    public static void main(String[] args) {
        // TODO code application logic here
        int c;
        int d;
        Random obj = new Random();  //new a Random object
        c = obj.nextInt();
        d = obj.nextInt();
        System.out.println("c = "+c);
        System.out.println("d = "+d);
    }

}
```

Output - JavaApplication1 (run)

```
run:
c = -2070687198
d = 139255272
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
c = 676205521
d = -816116493
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
c = 1311992650
d = 220223634
BUILD SUCCESSFUL (total time: 0 seconds)
```

Different return each time!

Examples -- run 3 times

generate random integer in [0,bound-1]

```java
package javaapplication1;
import java.util.*;
/**
 *
 * @author Zheng Qingqing
 */
public class JavaApplication1 {

    public static void main(String[] args) {
        // TODO code application logic here
        int c;
        int d;
        Random obj = new Random();  //new a Random object
        c = obj.nextInt(101);       //random integer in [0,100]
        d = obj.nextInt(450);       //random integer in [0,449]
        System.out.println("c = "+c);
        System.out.println("d = "+d);
    }

}
```

Output - JavaApplication1 (run)

```
run:
c = 88
d = 80
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
c = 74
d = 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
c = 53
d = 301
BUILD SUCCESSFUL (total time: 0 seconds)
```

Different return each time!

Scaling

- Usually we do not just want random numbers falling in [0,1).

So we need to make random numbers lie in a longer or shorter range by multiplying them by a scale factor.

- Multiply or divide the return:

a in [0,1) -> 100* a in [0, 100)

b in [0,1) -> (-1)* b in (-1, 0]

Scaling

Then we can also add offset to move the range in R

- Add offset:

  a in [0,1) -> 100 + a in [100, 101)

  b in [0,1) -> (-1) + b in [-1, 0)

- Combine the two ways:

  a in [0,1) -> 100 * a+1 in [1,101)

  b in [0,1) -> 2 * b + 10 in [10,12)

Examples

generate random number in [0,100)

```java
public class JavaApplication1 {
    public static void main(String[] args)
    {
        double a = 0;
        double b = 0;

        a = Math.random();
        b = Math.random();
        a = a * 100;
        b = b * 100;
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }
}
```

Output - JavaApplication1 (run)

```
run:
a = 5.609956211588529
b = 48.31335170540295
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
a = 61.45476537590121
b = 73.00900016674228
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
a = 28.388316624816177
b = 50.94243645387455
BUILD SUCCESSFUL (total time: 0 seconds)
```

Your results will be also different!

Examples

generate random number in bound

```java
package javaapplication1;
/**
 *
 * @author Zheng Qingqing
 */
public class JavaApplication1 {

    public static void main(String[] args) {
        // TODO code application logic here
        double a;
        double b;
        a = Math.random()*10+2;        //random number in [2,12)
        b = Math.random()*(-5)+10;      //random number in (5,10]
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }

}
```

Output - JavaApplication1 (run)

```
run:
a = 7.085447657717045
b = 6.907943269029362
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
a = 3.4401661296205113
b = 8.225632293958396
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
a = 3.2052220118351142
b = 5.817640144549264
BUILD SUCCESSFUL (total time: 0 seconds)
```

Your results will be also different!

Truncation

- Truncation :

  to cut the tail of the number, for example,

  3.3 -> 3, 5.3333 -> 5

- Code:

```
double d = 3.3;
int  i =  (int)  d;
```

In this case, we use (int) to convert a double(or float) to an integer.

Examples

generate random integers in [0,100)

```java
public class JavaApplication1 {
    public static void main(String[] args)
    {
        double a = 0;
        double b = 0;

        a = Math.random();
        b = Math.random();
        a = a * 100;
        b = b * 100;
        int intA = (int) a;
        int intB = (int) b;
        System.out.println("a = "+intA);
        System.out.println("b = "+intB);
    }
}
```

Output - JavaApplication1 (run)

```
run:
a = 37
b = 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
a = 88
b = 21
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - JavaApplication1 (run)

```
run:
a = 64
b = 91
BUILD SUCCESSFUL (total time: 0 seconds)
```

Your results will be different!

Summary

- Math.random() – a Math class-based way to generate random numbers.
- nextInt()/nextInt(int bound) – a Random Object-based way to generate random integers.
- Scaling and truncation – useful to change the range of random values.

Test exercise

The end

Thank you!