Random Numbers

• The book uses random numbers to simulate coin tosses. What other uses of random numbers can we think of?

Uses for Random Numbers

• Games
  – Games of chance such as poker or blackjack
  – Games where unexpected events occur, so that game play is different each time

• Cryptography
  – Random numbers are used to generate keys that are used to encrypt information

• Testing software
  – Certain techniques rely on randomness

Uses for Random Numbers

• Monte Carlo Simulation
  – Traffic flow
  – Stellar evolution
  – Stock market forecasting
  – Oil well exploration

• Scientific and Medical Experimentation
  – Assign subjects to treatment and control groups randomly
Uses for Random Numbers

- Polling
  - Randomly select people to poll
- Lottery
  - Randomly print and distribute lottery 'scratchers'
  - Randomly choose lotto numbers
- Contests
  - "You may have already won $1,000,000"

Math.random()

- public static double random()
  - member of class Math
- Math.random() generates a random number $r$ with a value greater than or equal to 0.0 and less than 1.0
  - Each value is equally likely
  - $0.0 \leq r < 1.0$
  - book is not very precise about this

RandomPrint

```java
// RandomPrint.java: Print random numbers in the range 0.0 <= r < 1.0

class RandomPrint {
    public static void main(String args[]) {
        int n = 10;
        System.out.println("We will print " + n + " random numbers");
        printRandomNumbers(n);
    }

    static void printRandomNumbers(int count) {
        for (int i = 0; i < count; i++) {
            System.out.println(Math.random());
        }
    }
}
```
What if we want integers?

- Math.random() produces random doubles
- What if we want integers instead?
  - For example, what if we want to get random integers ranging from 1 to 10?
  - Need to convert double r to integer i:
    - $0.0 \leq r < 1.0$ to $1 \leq i \leq 10$

Generating random integers

- Need to convert double r to integer i:
  - $0.0 \leq r < 1.0$ to $1 \leq i \leq 10$
- So, need to map doubles to integers so that each number from 1 to 10 is equally likely
  - How?

Generating random integers

- Need to map doubles to integers so that each number from 1 to 10 is equally likely
- Break up the range 0 to 1 into 10 equally sized sections, and map each section to an integer:
  - $[0.0, 0.1) \rightarrow 1$
  - $[0.1, 0.2) \rightarrow 2$
  - ...
  - $[0.9, 1.0) \rightarrow 10$
Generating random integers:

Algorithm

- Multiply r by 10
  
  \(-0.0 \leq r < 1.0 \rightarrow 0.0 \leq r < 10.0\)

- Convert r to an integer
  
  \(-0.0 \leq r < 10.0 \rightarrow 0 \leq i < 9\)

- Add 1
  
  \(-0 \leq i < 9 \rightarrow 1 \leq i \leq 10\)

Generating random integers:

Implementation

- Multiply r by 10
- Convert to an integer
- Add 1

```java
static void printRandomNumbers(int count) {
    double r;
    int value;
    for (int i = 0; i < count; i++) {
        r = Math.random();
        value = (int)(r * 10) + 1;
        System.out.println(r + "\t" + value);
    }
}
```

Generating random integers

- What if we want to pick one person out of a million as our contest winner?
- What if we want to have the computer roll dice for us in Craps?
  - What values can be produced by two dice?
A warning

- Be careful
  - Generating one random number between 2 and 12 is not the same as
  - Generating 2 random numbers between 1 and 6 and adding them
- They have different probability distributions
- You need to understand the real-world behavior you are trying to simulate

Recursion

- A recursive algorithm is one that is defined in terms of itself
  - In order to determine the result of the algorithm for some value, you must use the same algorithm with some 'smaller' value
  - There is a 'smallest' value for which the algorithm produces a result

Recursion

- General Form of recursive algorithm
  - There is a base case.
  - There is a recursive case

RecursiveAlgorithm( x )
if (stopping condition)
  // do whatever stops the algorithm and return the result
else
  // call the algorithm again
  RecursiveAlgorithm( smaller x )
Recursion Example: Factorial

- Factorial
  \[ n! = n \times (n-1) \times (n-2) \times \ldots \times 1 \]
  \[ n! = n \times (n-1)! \]
  \[ 1! = 1 \]

```java
static long factorial( int n ) {
    if ( n <= 1 )
        return 1;
    else
        return (n * factorial(n - 1));
}
```

Recursion Example: Factorial

- What's going on here?
- Call factorial(4)

  factorial(4) calls factorial(3)
  factorial(3) calls factorial(2)
  factorial(2) calls factorial(1)
  factorial(1) returns 1
  factorial(2) returns 2
  factorial(3) returns 6
  factorial(4) returns 24

Recursion Example: Exponentials

- Exponentials
  \[ n^x = n \times n \times \ldots \times n \quad \leftarrow \text{x times} \]
  \[ n^1 = n \times n^{x-1} \]
  \[ n^0 = 1 \]

```java
static long exponential( int n, int x ) {
    if ( x <= 0 )
        return 1;
    else
        return (n * exponential(n, x - 1));
}
```
Recursion Examples

- Searching
  - Dictionary
- Sorting
  - Merge Sort
- Many data structures are manipulated with recursive algorithms
  - CMPS 12B, 101

Recursion Exercise

- JBD problem 4.18
  - Write and test a method that will recursively print all the characters from 'a' through 'z'. Remember that each character is one more than the previous.
  - What is the base case?
  - What is the recursive case?