CS 241 — Introduction to Problem Solving and Programming

Fundamentals of Programming

Methods. . . putting it together.

Feb 9, 2005

Specification

Simulation is the modelling of a dynamic process from the real world. Often it requires an element of randomness.

Specification:

> Write a program that simulates rolling a set of dice. Given a number of dice, a number of rolls, and a roll value, run a simulation of rolling a that set of dice the specified number of times, reporting on the number of occurrences of the roll value and its frequency.

Use method `Random.nextInt(int)` to generate a random number from 1 to the given integer.

Algorithm sketch

- Input the number of dice
- Input the number of rolls
- Input the roll value to monitor
- Run the simulation to compute the number of occurrences
- Find the frequency by dividing occurrences by rolls
- Display the results

Main method

```
public class DiceGame {
    public static void main(String[] args) {
        // The number of dice to use
        int dice = DocsIO.readint("How many dice do you want to use? ");
        // The number of times we roll the dice
        int rolls = DocsIO.readint("How many rolls would you like? ");
        // The number whose frequency we are monitoring
        int number = DocsIO.readint("What number should we monitor? ");

        // The calculated occurrences of the monitored number
        int occurrences = rollDice(dice, rolls, number);
        System.out.println(number + " occurred " + occurrences +
                           " times with frequency " +
                           ((double) occurrences / rolls));

    }
    . . .
```

Zooming in

How do we do the simulation? We want roll dice based on the given information.

- Initialize the occurrences to 0
- Repeat *numberOfRolls* times
  - Roll the set of dice once
  - If the result equals the number we're monitoring, increment the occurrences
- Return the number of occurrences

Big rollDice method

```
/**
 * Roll a given number of dice a given number of times, reporting
 * the occurrences of a give value.
 * Roll the dice in a count-controlled loop (bounded by the number of
 * rolls), incrementing an accumulator each time the number occurrs.
 * @param numDice The number of dice to use.
 * @param numRolls How many times to roll the dice
 * @param monitorNumber The value whose frequency we are counting.
 * @return The integer number of occurrences of the monitored value
 */
static int rollDice(int numDice, int numRolls, int monitorNumber) {
    // The accumulator
    int occurrences = 0;
    for (int i = 0; i < numRolls; i++)
        if (rollDice(numDice) == monitorNumber)
            occurrences++;
    return occurrences;
}
```

Zooming in further

How do we simulate a single roll?

- Initialize the total value of the roll to zero
- Repeat *numberOfDice* times
  - Simulate the roll of a single die
  - Add the result to the total value
- Return the value

Medium rollDice method

```
/**
 * Roll a give number of dice once, reporting the rolled value.
 * Repeatedly roll one die in a count-countrolled loop (bounded by the
 * number of dice), incrementing an accumulator by the value of
 * the die. Return the accumlated value.
 * @param numDice The number of dice to use
 * @return The total value of all dice rolled.
 */
static int rollDice(int numDice) {
    // The accumulator
    int value = 0;
    for (int i = 0; i < numDice; i++)
        value += rollDice();
    return value;
}
```

Small rollDice method

To simulate a single roll of a single die, randomly generate a number from one to size. Use a standard method `Random.nextInt()`.

```
/**
 * Roll a single die, reporting the value.
 * Use the method Random.nextInt() to generate a new random
 * number between 1 and 6, and return that number.
 * @return The randomly generated number, from 1 to 6 inclusive.
 */
static int rollDice() {
    return Random.nextInt(6);
}
```

Dice simulation

```
ar1121: {34} java DiceGame
How many dice do you want to use? 1
How many rolls would you like? 1
What number should we monitor? 4
4 occurred 0 times with frequency 0.0
ar1121: {35} java DiceGame
How many dice do you want to use? 1
How many rolls would you like? 6
What number should we monitor? 4
4 occurred 0 times with frequency 0.0
ar1121: {36} java DiceGame
How many dice do you want to use? 1
How many rolls would you like? 6
What number should we monitor? 4
4 occurred 2 times with frequency 0.3333333333333333
```

Dice simulation

```
ar1121: {37} java DiceGame
How many dice do you want to use? 2
How many rolls would you like? 1000
What number should we monitor? 2
2 occurred 36 times with frequency 0.036
ar1121: {38} java DiceGame
How many dice do you want to use? 2
How many rolls would you like? 1000
What number should we monitor? 7
7 occurred 169 times with frequency 0.169
ar1121: {39} java DiceGame
How many dice do you want to use? 2
How many rolls would you like? 1000
What number should we monitor? 11
11 occurred 57 times with frequency 0.057
```

CS 241 — Introduction to Problem Solving and Programming

Fundamentals of Programming

Introduction to Recursion

Feb 9, 2005

Recursion

Recursion is the defining of something using the thing you are defining. A method that calls itself is recursive. A recursive method is self-referential.

Examples of recursion:

- The set of things mentioned on this slide.
- $n! = n \times (n-1)!$
- PINE: Pine is $n$ot $e$lm.

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd called with a = 30, b = 12

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd called with a = 12, b = 6

gcd called with a = 30, b = 12

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd called with a = 6, b = 0

gcd called with a = 12, b = 6

gcd called with a = 30, b = 12

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd returns 6

gcd called with a = 12, b = 6

gcd called with a = 30, b = 12

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd returns 6

gcd called with a = 30, b = 12

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd returns 6

gcd called with a = 72, b = 30

Recursion

```
static int gcd(int a, int b) {
    if (b == 0) return a;
    else return gcd(b, a % b);
}
```

gcd returns 6

Recursion

```
static int gcd(int a, int b) {
    System.out.println("gcd called with " + a + " and " + b);
    if (b == 0) {
        System.out.println("returning " + a);
        return a;
    }
    else {
        int temp = gcd(b, a % b);
        System.out.println("returning " + temp);
        return temp;
    }
}
```

```
gcd called with 72 and 30
gcd called with 30 and 12
gcd called with 12 and 6
gcd called with 6 and 0
returning 6
returning 6
returning 6
returning 6
GCD: 6
```

Loops to recursion

How might we write a method like this, using recursion?

```
static void sayAloha(int n) {
    for (;n > 0; n--)
        System.out.println("Aloha");
}
```

Loops to recursion

How might we write a method like this, using recursion?

```
static void sayAloha(int n) {
    for (;n > 0; n--)
        System.out.println("Aloha");
}

static void sayAloha(int n) {




}
```

Loops to recursion

How might we write a method like this, using recursion?

```
static void sayAloha(int n) {
    for (;n > 0; n--)
        System.out.println("Aloha");
}

static void sayAloha(int n) {
    if (n !=0) {


    }
}
```

Loops to recursion

How might we write a method like this, using recursion?

```
static void sayAloha(int n) {
    for (;n > 0; n--)
        System.out.println("Aloha");
}

static void sayAloha(int n) {
    if (n !=0) {
        System.out.println("Aloha");

    }
}
```

Loops to recursion

How might we write a method like this, using recursion?

```
static void sayAloha(int n) {
    for (;n > 0; n--)
        System.out.println("Aloha");
}

static void sayAloha(int n) {
    if (n !=0) {
        System.out.println("Aloha");
        sayAloha(n - 1);
    }
}
```