CS 120 Lecture 08

Flow Control: Repetition with For Loops

(Alice In Action, Ch 4)

21 September 2012

Slides Credit: Joel Adams, Alice in Action

Flow of Control

Sequential Execution

Each instruction is executed in order they are written (after the previous one, before the next on).

Functions

Enable procedural decomposition.

Repeat statements by calling functions multiple times.

Flow of Control

Selection

Some statements are executed while others are not.

Repetition

Statements can be repeated some fixed number of time, or else can be repeated until some event signals they should not be repeated any more.

3

Flow Control

- Flow: sequence of steps for performing a user story
- Flow control statement: structure for managing flow
- Flow control statements used in previous chapters
  - **`doInOrder`**: produces a sequential execution
  - **`doTogether`**: produces a parallel execution
  - methods and functions: name a block of statements

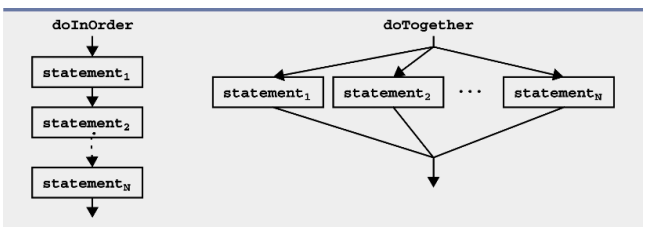


FIGURE 4-1 The flows produced by the `doInOrder` and `doTogether` statements

4

Flow Control

- Control statements introduced in the current chapter
  - **`if`**: directs program flow along one of two paths
  - **`for`**: directs flow into a fixed number of loops
  - **`while`**: directs flow into an arbitrary number of loops

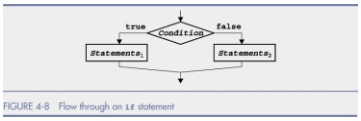


FIGURE 4-8 Flow through an `if` statement

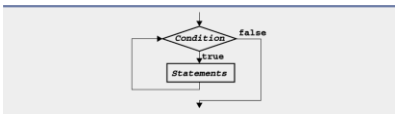


FIGURE 4-30 Flow through a `while` statement

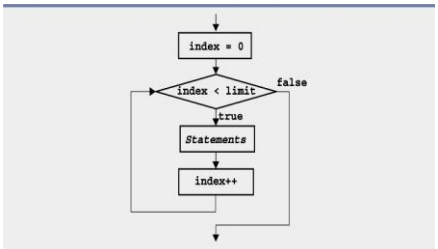


FIGURE 4-20 Flow through a `for` statement

5

Objectives

- Review using the **`if`** statement to perform some statements while skipping others
- Use the **`for`** and **`while`** statements to perform (other) statements more than once

- Lean about design patterns.

6

Introducing Repetition

- Selection:
  - The **`if`** statement executes its body 0 or 1 times based on the condition.
- Indefinite Loop
  - The **`while`** loop executes its body 0 or more times based on the condition.
- Definite loop
  - The **`for`** loop executes its body a fixed number of times.

`while` Statement Mechanics

- Structure of a **`while`** loop
  - ```
    while ( Condition ) {
        Statements
    }
    ```
- The **`while`** loop is more general than the **`for`** loop
  - Flow enters **`while`** structure if ***`Condition`*** is **`true`**
  - One statement must eventually falsify ***`Condition`*** . Otherwise, you get an infinite loop that will never ends

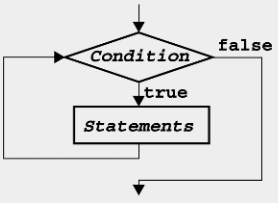


FIGURE 4-30 Flow through a **`while`** statement
    ```

Counter-Controlled Pattern

Counter-Controlled Loop

Evaluates a counter in loop's logical expression:

Pseudo-code for this pattern:

```
initialize counter
while counter > critical-value
     do controlled block statements
     decrement (increment) counter
```

9

Accumulator Pattern

Accumulator Pattern

Used to combine many values into one value.

Eg. Sum several values in a loop. Can be used with any loop pattern.

```
initialize accumulator variable(s)
loop to repeat each item:
     combine new data item into accumulator
```

10

Demo

Calculating Factorials

5! is 1 * 2 * 3 * 4 * 5 = 120

3! is 1 * 2 * 3 = 6

We can use a counter-controlled loop and accumulator pattern to calculate the results.

The accumulator is going to accumulate multiplication values: what number should we start with?

"Identity" for multiplication.

11

Introducing Definite Loop

- Refer to **`flapWings()`** method from Figure 2-16
- Enhancement: use **`for`** loop to flap wings **`numTimes`**
- Overview for implementing the enhancement
  - Open the **`flapWings()`** method
  - Adjust the **`duration`** values for the wing movements
  - Drag **`loop`** control to the top of the method and drop
  - Select **`numTimes`** for number of iterations
  - Drag the **`doInOrder`** statement into the **`for`** statement

12

Introducing Repetition (continued)

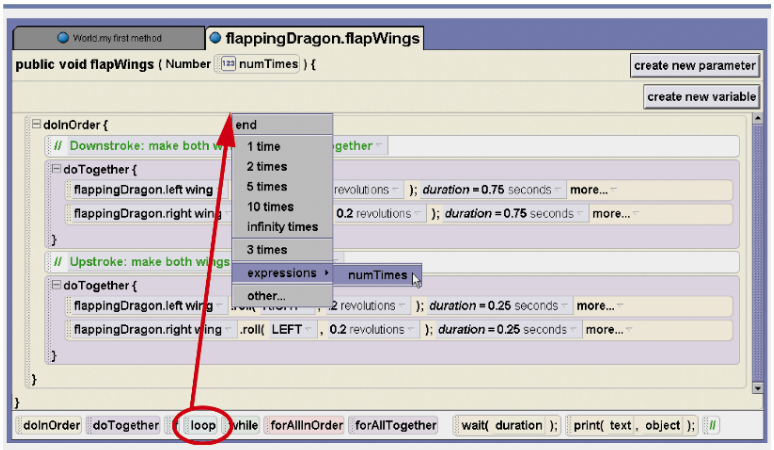


FIGURE 4-17 Dragging the `loop` control

Introducing Repetition (continued)

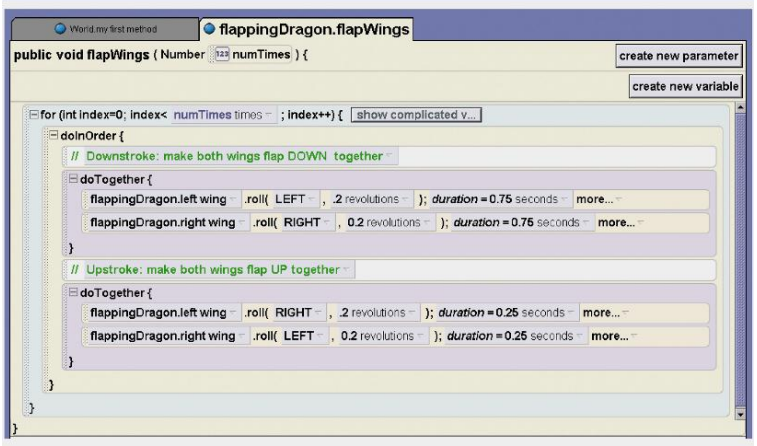


FIGURE 4-19 The revised `flapWings()` method

Mechanics of the `for` Statement

- Repeat statement execution a fixed number of times
- Example: pass 3 to `flapWings()` for 3 flaps
- Structure of the simple `for` statement
  - `for(int index = 0;index < limit;index++){`
    `Statements`
    `}`
- The `for` statement is also known as a counting loop
  - First statement in ( ) initializes the `index`
  - Second statement in ( ) checks `index` against `limit`
  - Third statement in ( ) increments the `index`

15

Mechanics of the `for` Statement (continued)

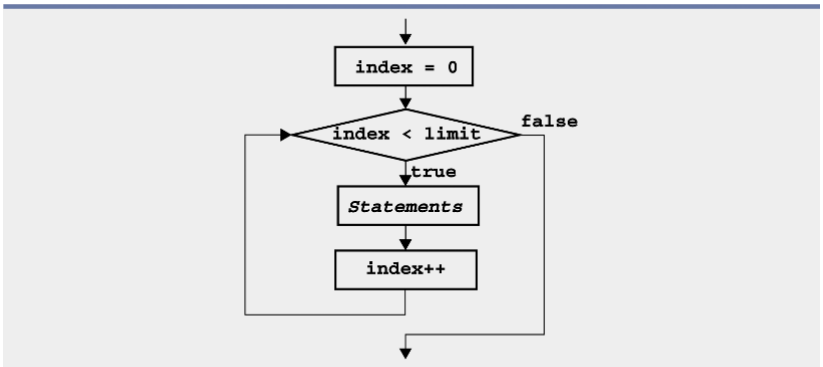


FIGURE 4-20 Flow through a `for` statement

16

Mechanics of the `for` Statement (continued)

- To test a for loop, trace the behavior with values
  - Statements are executed while **`index < numTimes`**
  - Example: send **`flapWings(3)`** to the **`dragon`** object

FIGURE 4-19 The revised `flapWings()` method

Mechanics of the `for` Statement (continued)

Step	Flow is in...	Effect	Comment
1	`index = 0;`	Initialize **`index`**	**`index`**'s value is `0`
2	`index < numTimes` `(0 < 3)`	The condition is **`true`**	Flow is directed into the loop
3	`doInOrder`	Flap wings	The first repetition
4	`index++`	Increment **`index`**	**`index`**'s value changes from `0` to `1`
5	`index < numTimes` `(1 < 3)`	The condition is **`true`**	Flow is directed into the loop
6	`doInOrder`	Flap wings	The second repetition
7	`index++`	Increment **`index`**	**`index`**'s value changes from `1` to `2`

FIGURE 4-21 Tracing the flow of `flapWings(3)`

continued

Mechanics of the `for` Statement (continued)

Step	Flow is in...	Effect	Comment
8	`index < numTimes (2 < 3)`	The condition is `true`	Flow is directed into the loop
9	`doInOrder`	Flap wings	The third repetition
10	`index++`	Increment `index`	`index`'s value changes from `2` to `3`
11	`index < numTimes (3 < 3)`	The condition is `false`	Flow is directed *out of* the loop
12	Flow leaves the `for` statement, moving to the end of the method		

FIGURE 4-21 Tracing the flow of `flapWings(3)` *(continued)*

Mechanics of the `for` Statement (continued)

- Purpose of **`show complicated version`** button
  - Change initial value of **`index`** and/or update to **`index`**
  - Example: change update to **`index+=2`**

- Simple version of **`for`** lets you modify **`limit`** value
- Note: neither version of **`for`** allows you to count down in Alice, but you can in Java.

The `while` vs `for` loop

- ***for* loop property: `limit`** must be set to a fixed value
- Circumstance when the **`for`** loop is appropriate
  - Statements are to be executed a fixed number of times
  - "Defined" ahead of time → Definite loop.
- Problem: looping when the **`limit`** value is unknown
  - e.g. how many flaps for "Dragon then descends and lands on the drawbridge"?
  - Solution: use a **`while`** statement
  - Not "defined" ahead of time → Indefinite loop.

21

Comparing the `for` and `while` Statements

- **`while`** statement is more general and can produce any type of repetition, including the for loop behavior
- **`for`** statement is used for fixed number of repetitions
  - Loop selection question to ask: "Am I counting?"
  - If yes, use a **`for`** statement; otherwise, use **`while`**
- Both loops test conditions before flow enters structure
- Both loops are bypassed if initial condition is **`false`**
- **`More examples:`**
  - **`drop a ball: continue the bounce while rebound distanceToGround >0`**
  - **`FabonaciGirl: use for to compute Fabonaci numbers and move corresponding steps`**

22

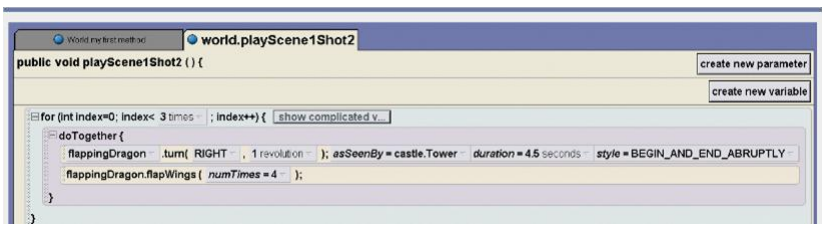
Nested Loops

- Example: three shots enhancing Scene 1 of dragon animation
  - Dragon flies toward a castle in the countryside
  - As dragon nears castle, it circles the tower three times
  - Dragon then descends and lands on the drawbridge (section 4.4)
- One way to build the first shot
  - Go to go into the **`Add Objects`** window
  - Position the dragon above the castle's drawbridge
  - Move dragon up until it is even with the castle's tower
  - Drop a dummy and then drag the dragon off-screen
  - Use **`setPointOfView()`** to properly position dragon

23

Nested Loops (continued)

- One way to build the second shot: circling the tower 3 times
  - Outer ***`for loop`*** for circling the tower 3 times
  - inner ***`for loop`*** **`in flapWings() for flapping wings 4 times in each circle`**
  - **`More Alice details`**
    - **`AsSeenBy()`** attribute revolves dragon around castle
    - Increase **`duration`** of **`turn()`** to synchronize moves
    - Set **`style`** suitable for animation



24

Fibonacci Numbers

- The original problem investigated by Fibonacci in1202
  - about how fast rabbits could breed in ideal circumstances
  - Suppose a newly-born pair of rabbits, one male, one female, are put in a field.
  - Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits.
  - Suppose that our rabbits **never die** and that the female **always** produces one new pair (one male, one female) **every month** from the second month on.
  - How many pairs will there be in one year?

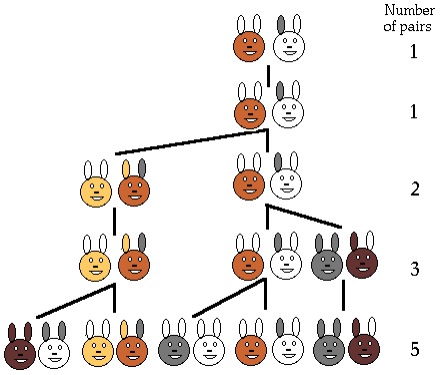
25

Fibonacci Numbers

- Analysis
  - At the end of the first month, they mate, but there is still one only 1 pair.
  - At the end of the second month the female produces a new pair, so now there are 2 pairs of rabbits in the field.
  - At the end of the third month, the original female produces a second pair, making 3 pairs in all in the field.
  - At the end of the fourth month, the original female has produced yet another new pair, the female born two months ago produces her first pair also, making 5 pairs.
  - …

26

Fibonacci Numbers in the Rabbit Growth Model



- Fibonacci numbers (sequence)
  - Number > 2 is found by adding two preceding numbers
  - Example: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

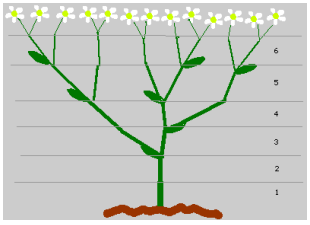
27

Fabonaci Numbers in Nature










http://britton.disted.camosun.bc.ca/fibslide/jbfibslide.htm

28

Spirals and the Fibonacci Numbers

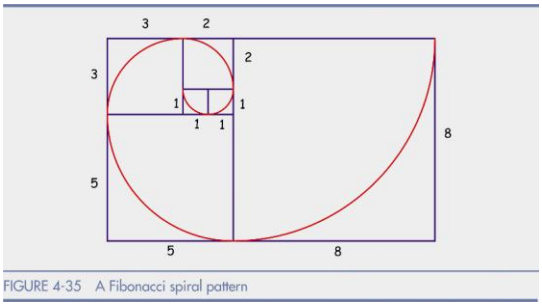
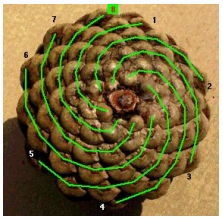
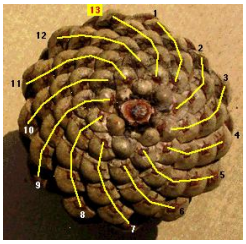


FIGURE 4-35 A Fibonacci spiral pattern





http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html

Fibonacci Number Example

- User story
  - Scene 1: girl finds an old book and reads contents
  - Scene 2: girl uses the map to locate a palm tree
  - Scene 3: girl follows spiral from tree to treasure site
- Coding spiral motion in Scene 3 is greatest challenge
  - Spiral inscribed in rectangles built from the sequence

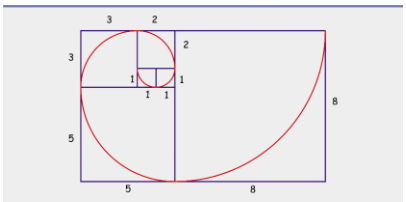


FIGURE 4-35 A Fibonacci spiral pattern

Spirals and the Fibonacci Function (continued)

- Approximating Fibonacci spiral in **`playScene3()`**
  - Have girl move 6 times
  - Distance of each move equals a corresponding Fibonacci number
  - While moving forward, girl also turns left 1/4 revolution
- **`playScene3()`** requires a **`fibonacci()`** function, which will be defined as an object method for the girl
- Save girl as **`fibonacciGirl`** for possible reuse

31

Spirals and the Fibonacci Function (continued)

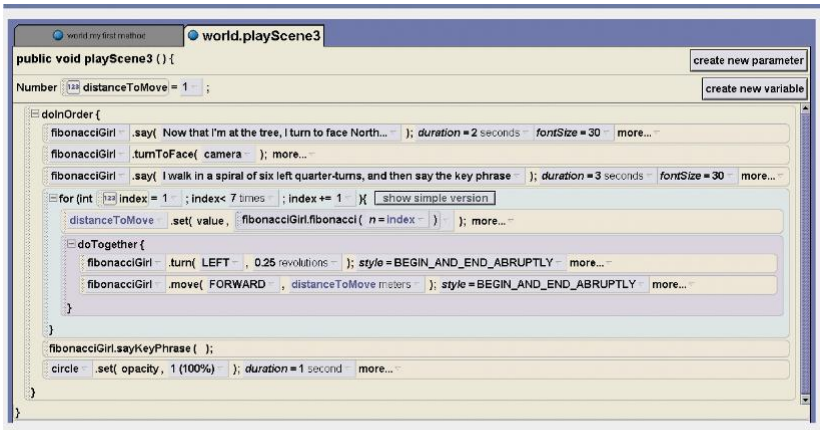


FIGURE 4-36 The **`playScene3()`** Method

32

The Fibonacci Function

- Defining the outline of **`fibonacci()`** function
  - Select girl and create a function named **`fibonacci`**
  - Create a **`Number`** parameter named **`n`**
- Formula: if n > 1, f(n) = sum of two preceding numbers
- Designing an algorithm to generate the n^{th} number
  - Create local variables: **`result`**, **`nextToLast`**, **`last`**
  - Add if statement to the function
  - If **`n == 1 or n == 2`**, **`result = 1`**
  - Otherwise calculate n^{th} value using formula in **`for`** loop
- **`fibonacci()`** calls in **`playScene3()`** specify spiral

33

The Fibonacci Function (continued)

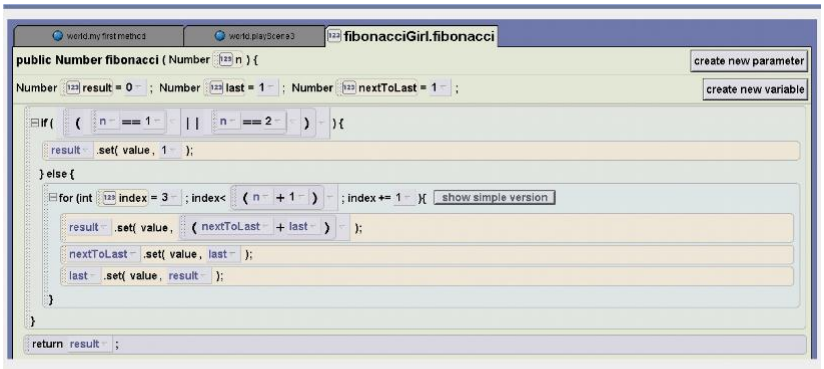


FIGURE 4-37 The **`fibonacci()`** function

34

Summary

- Flow control statement: controls the flow of statement execution
- **`if`** statement: directs flow along one of two paths based on evaluation of a condition
- **`for`** statement: repeats execution of a group of statements a fixed number of times
- **`while`** statement: repeats execution of a group of statements an appropriate number of times based on a loop continuation condition
- Design Patterns for writing programs.
  - Loops: Interactive, Counter, Sentinal
  - Accumulator Pattern

35

Student To Do's

- Readings:
  - Alice in Action, Chapter 4 (This week)
- Homework due Monday during lab time
  - Bring HW solution to demo.
- Practice, practice, practice!