**Participation 3**
- Participation 3 is now posted.
- Participation points are weighted to be 5% of your grade.
- Turning in the participation survey AND answering questions in class are important for your grade!

**Scores**
- Do check ALL of your scores.
- After the final, points won’t be changed.
- Two week window – no more!!
- If there is a problem, GET IT FIXED NOW!
- Pass Fail is really P/D/F. The D affects your GPA. But you do get credit for the class with a D
- Today is the last day to drop

**Labs**
- The inclass portion of the lab is best demonstrated – as you will be told what to fix (and hence a perfect score is likely).
- Some of you are really lax about attending labs. We have 110 points (200 if you are taking the optional final) left in the semester (3 homework, 3 inclass, and the final project). With the inclass and homework, we are building skills. If you don’t get the early assignments done, the later ones will be extremely difficult.
- Attendance is good – but performance is what determines your grade

**Chapter 12:**
Decision and Repetition Structures

Starting Out with Alice:
A Visual Introduction to Programming
First Edition
by Tony Gaddis

**Boolean Variables**
- Boolean Variables hold only two values
  - True
  - False

- Initial Value is usually set to be "TRUE"
- called a “default” setting
- How do we assign a value?

**Naming**
- Better if named to sound like a question:
  - wantCookie rather than cookieDesire
  - isDone rather than finishFlag
  - tooClose rather than closenessMeasure

- Write code (using a boolean) to alternate “bunny” going forward/backward
Boolean Functions

- **Boolean Functions** _return_ one value from the choice of true/false

- Also primitive functions that are Boolean
  - User prompted to answer Yes (true) or No (false)

---

If/Else

- **If/Else** is a _decision structure_
- Instruction tests a _condition_ (which gives a Boolean value) and then based on a true value does task “A” or task “B” if the value was false

---

**If/Else**

- BOTH sets of actions will NOT be performed … only ONE can be!
- The penguin falling or speaking are instructions that are _conditionally executed_
  - They do NOT always execute
  - Executed only under certain conditions
- With the embarrassed person (in the bonus inclass for last week), you had to have code for each alternative to be embarrassed and disappear.

---

**Decision Structure**

- If/Else Tile
- Penguin falls down into the hole
- Penguin says “Drats!”
-ctPenquin is decremented by 1
- End If

---

**Single-Alternative Decision Structures**

- **If/Else** is a _dual-alternative_ decision structure
  - Two paths of execution: one following TRUE and the other following FALSE
- **Single-alternative** decision structures are similar to the dual-alternative
  - The ELSE part is empty!
Is this correct?
if a.height < b.height
    mistletoe.moveto b
else
    mistletoe.moveto a
    wait 1 second
    a.color set red
    a.isShowing set to false

Can we simplify?
if a.height < b.height
    mistletoe.moveto b
else
    mistletoe.moveto a
    wait 1 second
    if a.height < b.height
        a.color set red
        a.isShowing set to false
    else
        b.color set red
        b.isShowing set to false

Single-Alternative Decision Structures

Start
Penguin turns to face hole

Hole wider than penguin?
T
Penguin walks to center of hole

F
Penguin falls into hole
End

Nested If/Else Instructions

An If/Else instruction is placed inside another If/Else instruction or nested.
The inner If/Else executes only if the outer If/Else is true.
Like following directions

Give a case that demonstrates the difference?
if a < b and a < c
    bug move up 1 meter
if a < b or a < c
    bug move up 1 meter
if a == b and a == c
    if a == b or a == c

Is there a difference between the following?
isShorter set to b < 10
if isShorter
    move mistletoe to b
if b < 10
    move mistletoe to b
Give a case that demonstrates the difference

\[
\begin{align*}
    s &\text{ set to } 0 \\
    \text{if } x > 0 & \quad s \text{ set to } 0 \\
    \text{increment } s & \quad \text{if } x > 0 \\
    \text{increment } s & \quad \text{if } y > 0 \\
    \text{increment } s & \quad \text{else if } y > 0
\end{align*}
\]

Simplify the following

1. not (x > y)
2. not (x > y and y > 0)
3. not (x! = 0 or y! = 0)
4. not (country == "USA" and state != "HI" and state != "AK")

What is the difference?

\[
\begin{align*}
    \text{if } age < 18 & \quad \text{isOld set to } age < 18 \\
    \text{isOld set to false} & \quad \text{else} \\
    \text{isOld set to true}
\end{align*}
\]

Relational Operators

- Binary...operate on two pieces of data
- Can use to create conditions in an If/Else instruction
- Alice uses "a" and "b" for placeholders
  - Drag the desired tile and replace either the "a" or "b" with a value...such as an object

Logical Operators

- Tests more than true/false...can do complex testing!
  - Test two conditions to see if they BOTH are true!
  - Or ONLY one condition is true!
  - Or NEITHER condition is true!
Examples: Write the pseudocode for the following

- Object disappear if object is taller than 5 or wider than 2.
- Do some action if the “score” is 1 or 3.
- Do some action if the game score is less than 5 and the car color is blue.

The Loop Instruction

- Loop instructions cause one or more other instructions to repeat or loop a certain number of times.

The Loop Instruction

- The Loop instruction has an empty slot where instructions can be inserted.
- The number of times to loop can be a variable.

Computing the Number of Times to Repeat

- What if you want the something to loop and NOT have to specify a particular number of repetitions?
  - For example...getting a ball to roll across the screen, regardless of where the ball is initially placed
  - Use the soccerBall's distance to function to calculate the distance
  - Loop uses integers (whole numbers) and decimal portions are discarded
  - Objects loop 4 times, not 4.8; 3 times, not 3.4

Infinite Loops

- Infinite Loops...the loop NEVER stops!
  - Use for objects that shouldn't stop!
- If placed in consecutive order...the next instructions will NEVER occur, since the loop NEVER ends!
- Place an infinite loop into a Do Together structure with other items.

The While Instruction

- While instruction is a loop that repeats as long as a Boolean condition is true
- Second repetition structure
- Called conditional loop...since the loop is controlled by a condition
a **While** Instruction for tennis action

```
while distance > 0
  racket turns forward
  ball moves forward

Loop's condition is tested before each repetition of the loop
First, it tests condition
```

**Example:**

- Let's do an exercise routine which we repeat
- We want to slide five times to the left, then jump, then slide five times to the right, then jump.
- If we let distance alternate between -1 and +1 it is quite slick!

```
distance set to 1
Loop infinity times
  distance set value to ( ( distance * -1 ) )
Loop 5 times
  blueBallerina move left distance meters
  blueBallerina move up 0.5 meters
  blueBallerina move down 0.5 meters
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley