CS101 Lecture 23

Variables
(Alice In Action, Ch 3)

Objectives

• Learn and use three types of variables
• Understand the concept of variable scope.
• Understand Datatypes: Boolean, Number, String, Object

Slides Credit: Joel Adams, Alice in Action
Variables

- Variables: named containers for storing/retrieving data
- Three types of variables
  - **Method (local):** defined for use within a method
    - e.g. to store values to be used later
  - **Parameter:** variables passed to a method
    - to make methods more general and useful
  - **Object/Class (property):** used to store an object
    - propertyTools for creating method variables and parameters
- Buttons in the upper-right corner of the method box
- Tool for creating object variables
  - Button under the property pane of the details area

![Figure 3-1](image-url)  
*The buttons to create variables and parameters*
Method/Local Variables

• Defined using create new variable button
• Information needed to define a variable
  – Name: refers to a location in memory
  – Type: refers to kind of data stored; e.g., a number
  – Initial value: starting value of specified type; e.g., 1
• Method variables are local (valid only in the method)
• Common uses of method variables
  – Computing and storing values for later usage
  – Storing values entered by a user

Example 1: Storing a Computed Value

• User story objects: nativeGirl and horse
• Objective: move nativeGirl closer to horse
• Strategy for moving the girl the right distance
  – Define variable storing distance from the girl to horse
  – Ask the girl how far she is from the horse
  – Store the girl’s reply in the variable
  – Use variable’s current value in a move() message
• Preparing to program with user-defined variables
  – Position nativeGirl and horse in a new world
  – Review types: Number, Boolean, Object, Other
Example 1: Storing a Computed Value (continued)

• Creating the first local variable
  – Click the create new variable button
  – Define Number type named distanceToHorse
  – Drag distanceToHorse to the editing area
  – Set the variable value using a placeholder value of 1
  – Click on functions pane for nativeGirl object
  – Drag distanceInFrontOf( ) onto placeholder
  – Specify the entire horse as the object argument
Example 1: Storing a Computed Value (continued)

Figure 3-3: The Create New Local Variable dialog box

Example 1: Storing a Computed Value (continued)

Figure 3-7: Setting a variable's value to a function's answer
Example 1: Storing a Computed Value (continued)

- Using `distanceToHorse` in the `move()` method
  - Drag the `move()` message into the editing area
  - Select `Forward->expressions->distanceToHorse`
- A test of method shows the girl too close to horse
- Adjusting the distance that the girl moves
  - Click the list arrow next to `distanceToHorse`
  - Select `math->distanceToHorse--> .5`
- Use functions and variables over trial-and-error
  - Behavior will adapt to repositioning of objects
Example 1: Storing a Computed Value (continued)

FIGURE 3-11  The girl too close to the horse

Example 1: Storing a Computed Value (continued)

FIGURE 3-12  Adjusting a value in a message
Example 2: Storing a User-Entered Value

- User story built around the Pythagorean Theorem
  - `skaterGirl` requests the lengths of two sides
  - User passes `edge1` and `edge2` values to `skaterGirl`
  - `skaterGirl` computes `hypotenuse` and displays value

- Overview of implementation
  - Create `skaterGirl` method `computeHypotenuse()`
  - Declare three `Number` variables for triangle sides
  - Add two `say()` statements for interactive dialog
  - Add `set()` method for `edge1`
  - Incorporate `NumberDialog` into `set()` parameter list
Example 2: Storing a User-Entered Value (continued)

- Overview of implementation (continued)
  - Add `set()` method for `edge2`
  - Incorporate `NumberDialog` into `set()` parameter list
  - Add `set()` method for `hypotenuse`
  - Drag and drop `Math.sqrt()` over placeholder value
  - Pass a mathematical expression to `Math.sqrt()`
  - Define `String` variable named `hypotString`
  - Convert hypotenuse to string and store in `hypotString`
  - Add `say()` method to display results
  - Display concatenates dialog and `hypotString` value
Example 2: Storing a User-Entered Value (continued)

![Diagram](image1.png)

FIGURE 3-25 The `computeHypotenuse()` method (final version)

Example 2: Storing a User-Entered Value (continued)

![Image](image2.png)

FIGURE 3-26 Testing `computeHypotenuse()`
Parameters

- **Argument**: value passed to an object via a message
- **Parameter**: variable that stores an argument
- Example using the `say()` message
  - The parameter is a single variable of `String` type
  - One possible argument is “Oh, hello there!”
- You may pass multiple arguments to messages
  - Example: `roll()` message requires two arguments

Example 1: Old MacDonald Had a Farm

- Objective: display song lyrics for four animals
- Use of parameters helps avoid repetitious coding
- Two parameters needed: `animal` and `noise`
- Overview of implementation
  - Create new world with `scarecrow` before `fence`
  - Create `singVerse()` method for `scarecrow`
  - Add two parameters, `animal` and `noise`
  - Add `String` types, `firstLine` and `doubleNoise`
  - Incorporate parameters and variables in `say()` methods
  - Call `singVerse()` messages in `singOldMacDonald()`
Example 1: Old MacDonald Had a Farm (continued)

Figure 3.30 The `singVerse()` method

Example 1: Old MacDonald Had a Farm (continued)

Figure 3.31 The `singOld MacDonald()` method
Example 1: Old MacDonald Had a Farm (continued)

Example 2: Jumping Fish!

- User story: fish jumps, traces an arc, returns to water
- Fish object used is an instance of Pinkminnow
- Basic elements in the design of jump() method
  - One parameter: distance
  - Local var: height, halfDist, startStopDist, angle
  - Approximate height of jump as 1/3 of the distance moved
  - Assume starting/stopping point is 1/2 distance of jump
  - Set the pitch angle to .1
- The jump() method is implemented in Figure 3-35
- Program should be tested with multiple distance values
Example 2: Jumping Fish! (continued)

Example 2: Jumping Fish! (continued)
Property Variables

- Also called instance variables or object variables or class variables
- Property variables are defined within an object
  - Use the properties pane of the object’s details area
- Values in an object variable are not shared
- How to create and use a property variable
  - Open a new world and add a wizard object
  - Click on wizard properties
  - Click create new variable
  - Define String type (in Other list) called myName
  - Initialize myName to Jim
Property Variables (continued)

• How to create and use a property variable (continued)
  – Duplicate the wizard using copy button
  – Close the Add Objects window
  – Click on the second wizard in the object tree
  – Click the properties tab in the details area
  – Change the second wizard’s name from Jim to Tim
• Testing the identity of the wizards
  – Use say () messages to display each wizard’s name
  – Argument used: “My name is “ + wizard.myName
• Check object for predefined properties; e.g., opacity
Property Variables (continued)

FIGURE 3-41  Accessing property variables

Property Variables (continued)

FIGURE 3-42  The twin wizards introduce themselves
## Summary

- **Variable**: named container for storing and retrieving values
- **Types of variables**: method variables, parameters, object variables
- **Data types**: `Number`, `Boolean`, `Object`, `String`, ...

- **Parameter**: container for an argument
- **Argument**: data passed to a method or function