Getting Started with Java Using Alice

Declare Procedures

Objectives

This lesson covers the following objectives:

- Compare and define an animation and a scenario
- Write a storyboard
- Flowchart a storyboard
- Describe inheritance and how traits are passed from superclasses to subclasses

Objectives (cont.)

This lesson covers the following objectives:

- Describe when to implement procedural abstraction
- Demonstrate how to declare a procedure
- Identify and use procedural abstraction techniques to simplify animation development

Object Movement

Professional animators begin their process by developing a scenario—or story—that gives the animation a purpose.

Examples:

- A story representing a conflict and resolution.
- A lesson to teach a math concept.
- A process to simulate or demonstrate.
- A game to play for entertainment or training.

Scenario and Animation Examples

Defining the scenario, and the animation to represent the scenario, is the first step to programming your animation.

Scenario Type	Scenario	Animation
Story	A cat needs help to get down from a tree.	A firefighter climbs up the tree to save the cat.
Lesson	Memorizing chemistry symbols is difficult.	A timed game matches chemistry symbols with their definitions.
Process	A car has a flat tire.	A demonstration shows how to change a tire on a virtual car.
Game	An airplane must avoid objects in its path as it flies through the sky.	An interactive game maneuvers an airplane around objects in the sky.

What is a Storyboard?

A storyboard identifies the design specifications for the animation scenario: how objects appear, move, speak, interact, and so on. Once a scenario is defined, you can begin to develop the animation storyboard.

Two types of storyboards are used to plan an animation:

- Visual: A series of illustrated images that represent the main scenes of the animation.
- Textual: A detailed, ordered list of actions that each object performs in each scene of the animation.

Storyboard Formats

Storyboards are created in the following formats:

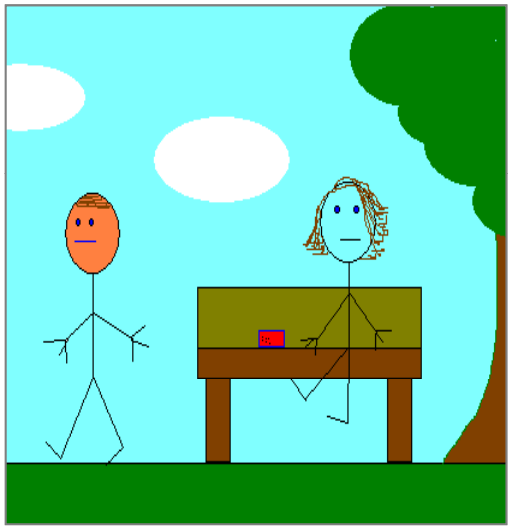
- Drawn with paper and pencil.
- Created using digital tools such as a word processing program, paint or drawing program, or presentation.
- Created using comments in the Alice 3 Code editor.

Visual Storyboards

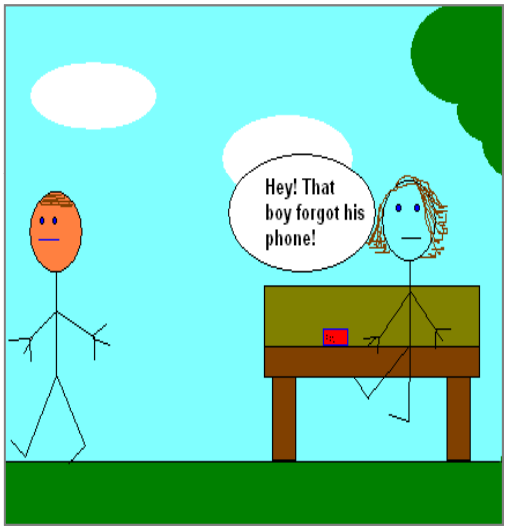
A visual storyboard helps the reader understand:

- The components of a scene.
- How the initial scene will be set-up.
- The moving and non-moving objects in a scene.
- The actions that will take place.
- The user interactions that will occur during the animation execution.

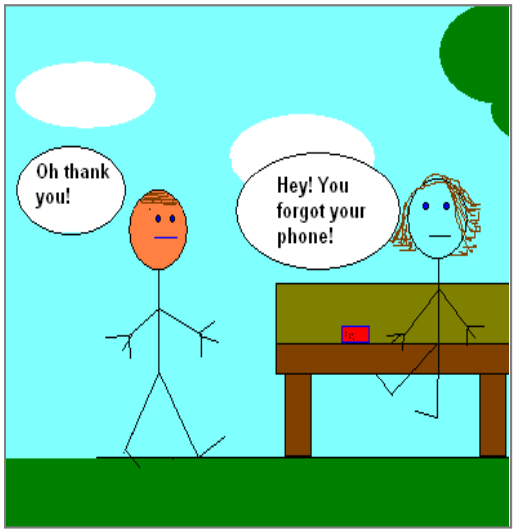
Visual Storyboard Example



Boy and girl sit on a park bench. The boy walks away, leaving his phone behind.



Girl notices the mobile phone. She thinks, “Hey! That boy forgot his phone!”



Girl says out loud “Hey! You forgot your phone!” Boy turns around and walks back to bench. He says, “Oh! thank you!”

Textual Storyboards

A textual storyboard helps the reader understand the actions that will take place during the animation.

The moving and non-moving objects can be easily identified within the action statements, but a more detailed description may be necessary if additional programmers are also involved in implementing any scene.

An algorithm is a list of actions to perform a task or solve a problem. In computing, a textual storyboard is an algorithm.

Textual Storyboard Example 1

Program the following actions in order:

Boy and girl sit on a park bench.

Boy stands up and walks away, leaving his mobile phone on the park bench.

Girl turns to look at the phone.

Girl thinks, “Hey! That boy forgot his phone!”

Girl says out loud, “Hey! You forgot your phone!”

Boy stops and turns around.

Boy walks back to the park bench and says, “Oh! Thank you!”

Textual Storyboard Example 2

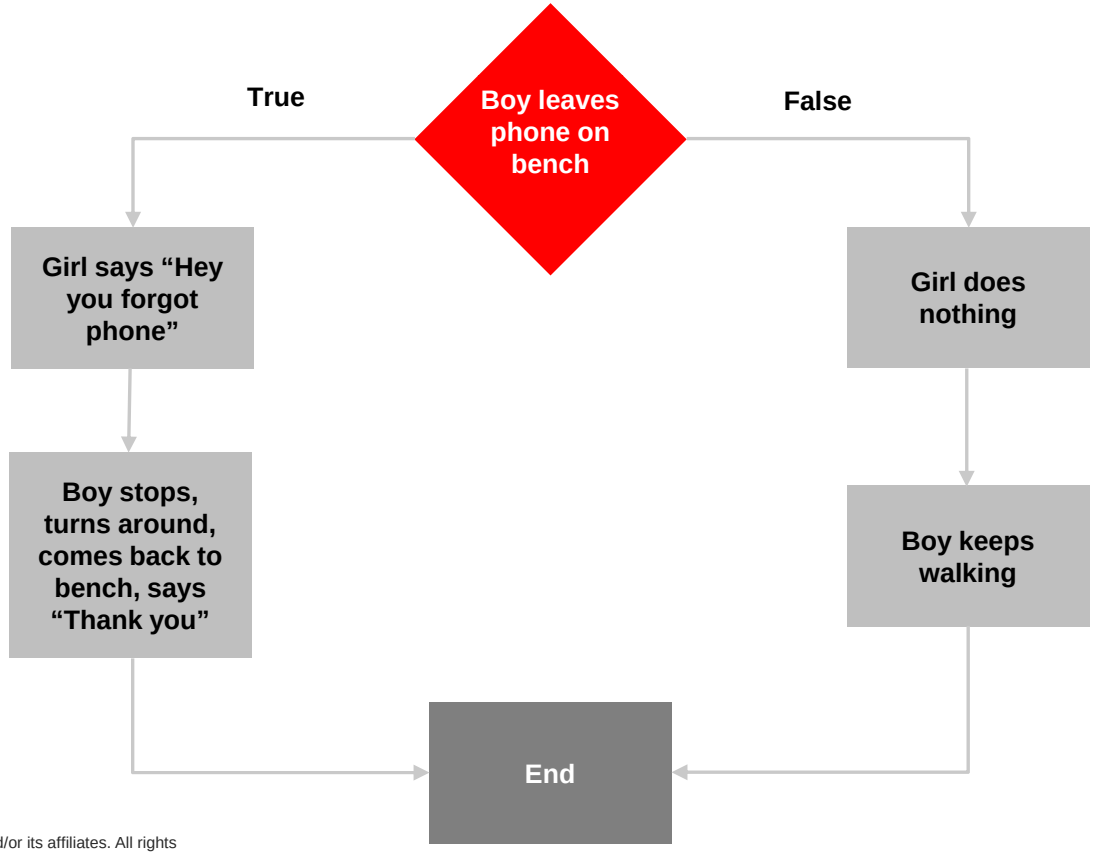
This example shows how you can develop your storyboard by first writing comments in the Code editor of your program. Afterwards, you can start to develop the animation directly from the storyboard.

Textual Storyboard Components

Component	Definition	Examples
Scene	The place (or “world” in Alice 3) where your story occurs.	Park, library, school, home
Objects	Moving or non-moving characters that you program to move and act.	Animals, cars, people, trees
Actions	Instructions for how each object should act in the scene.	Walk 2 meters, turn left, say “Hello!”
User Interactions	Ways in which the user viewing the animation can manipulate the objects in the animation.	Keyboard commands or mouse clicks to make objects move
Design Specifications	How the objects and scenery should look in the animation.	Size, position, location, color

Flowcharting a Storyboard

Flowcharting a storyboard helps you organize the flow of the animation actions and conditions.



Using Storyboards to Organize Your Program

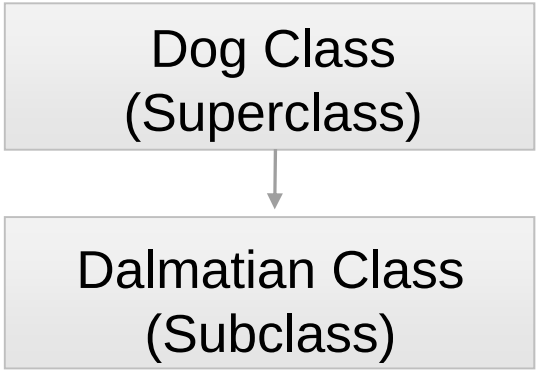
Textual storyboards can be used to generate program comment statements and organize program development.

Storyboards can also help programmers identify repetitive actions for one object, and identical actions that may be performed by multiple objects.

Inherited Characteristics

Let's examine how a Dalmatian inherits its characteristics:

- Characteristics of the Dog class (the parent class or "superclass") include four legs, two eyes, fur, and the ability to bark.
- Characteristics of the Dalmatian breed class (the child class or "subclass," which is a subset of the dog class) include white fur, black spots, and other characteristics.



Class Inheritance

Just like animals in the real world, objects in the programming world inherit the characteristics of their class, including all of the class's methods (procedures and functions).

For example, all objects within the quadruped class in Alice inherit the quadruped characteristics of four legs, a head, a body, etc. Within this quadruped superclass, subclasses exist for dogs, cats, wolves, lions, cows, etc. Each subclass adds characteristics that more specifically identify the objects within it.

Identifying Repetitive Behaviors in Your Storyboard

Understanding that subclasses belong to superclasses helps you, as a programmer, identify repetitive behaviors in your storyboard. For example:

- If you plan to program a dog, cat, and lion to walk, you should program the repetitive behavior of walking at the superclass, or quadruped, level.
- The result is that all of the subclasses (dog, cat, lion) can use the inherited characteristic of walking, and you do not have to program the repetitive walking behavior for each object individually.

Inheritance

When a dalmatian object is created, it inherits procedures, functions, and properties from the quadruped class and the dalmatian subclass that you can view in the Code editor.

Inheritance means that each subclass object inherits the methods and properties of its superclass.

Create Inherited Methods

In addition to the pre-defined methods, you can create your own methods and have them display, or be available for, any subclass object.

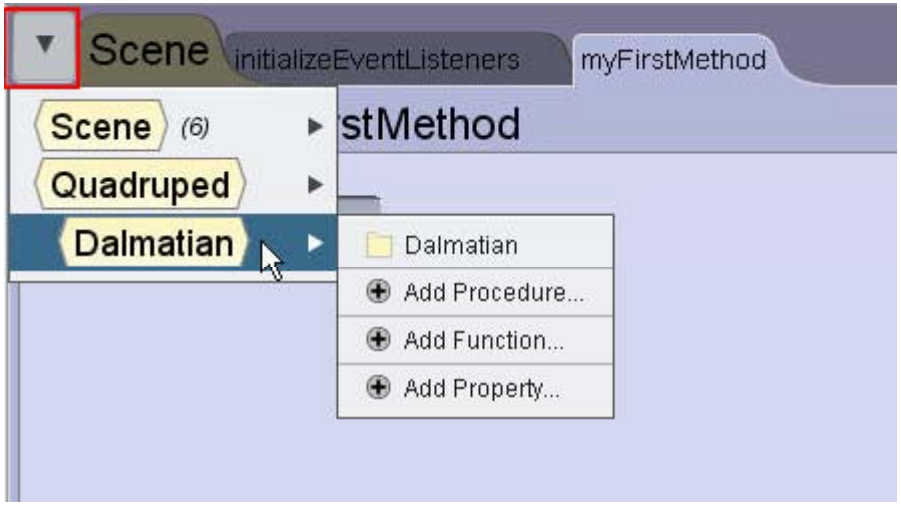
Inherited methods will always display at the top of the list of pre-defined methods once they are created.

myFirstMethod Tab

The myFirstMethod tab is displayed by default when the Code editor is opened.

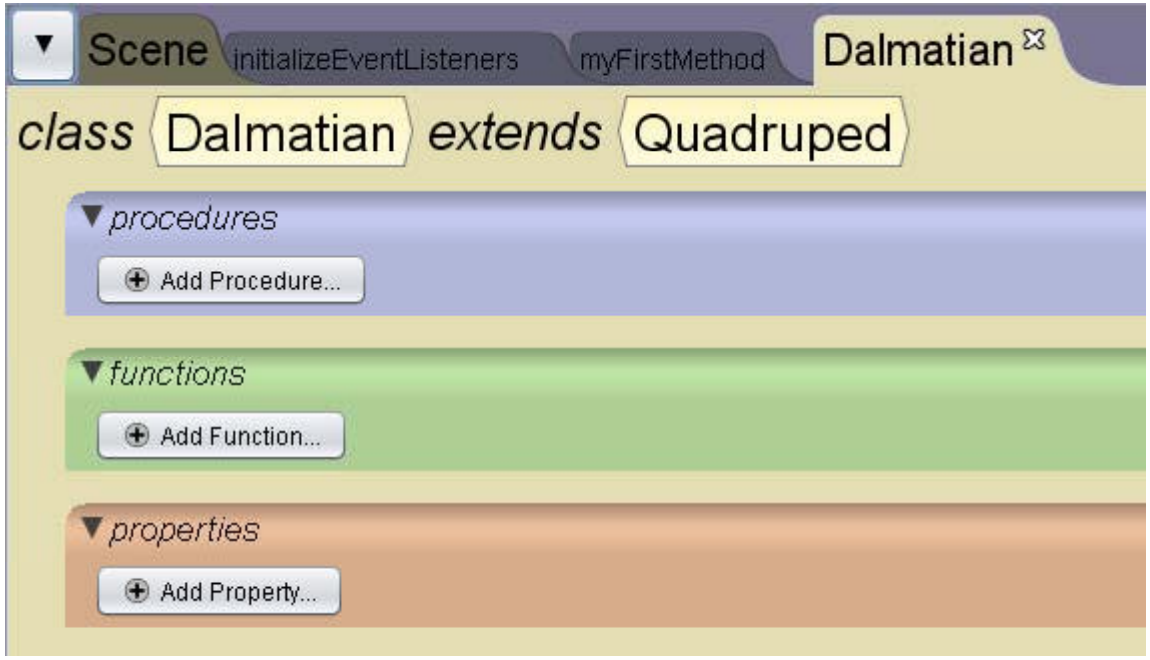
Class Hierarchy

Click the class hierarchy drop-down menu to the left of the myFirstMethod tab (indicated by a down-pointing arrow) to view the list of classes and subclasses in your animation.



View the Class Methods

Select a superclass or subclass to view the procedures, functions, and properties defined for the selected class.



Procedural Abstraction

Review the existing code or the textual storyboard to identify and plan the methods that you need to declare in your program.

Identify a repetitive behavior and create one method for it:

- Simplifies the code, making it easier to read.
- Allows many objects of a class to use the same method.
- Allows subclasses of a superclass to use the method.

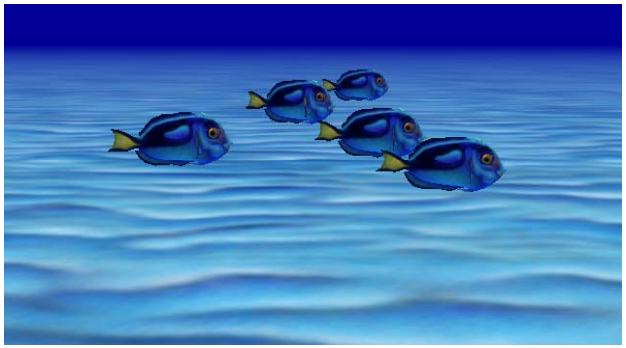
Procedural Abstraction Defined

An example of procedural abstraction is the removal of a repetitive or lengthy programming statement from myFirstMethod and the placing of it into its own method to make the code easier to read, understand, and reuse by multiple objects and multiple classes.

Procedural abstraction is the process of looking at programming code, identifying repetitive programming statements, and extracting them into their own methods, thus making the code easier to understand and reuse.

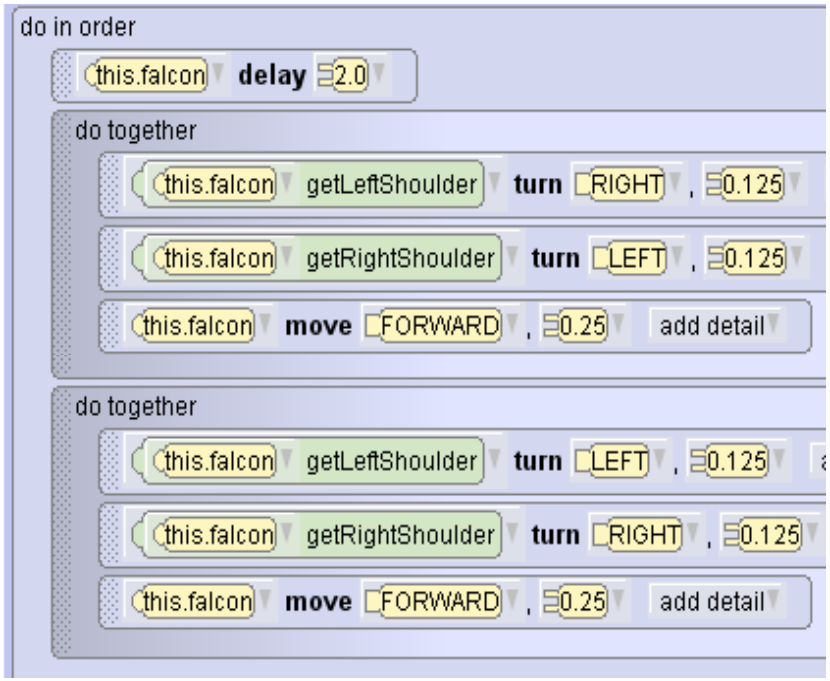
Procedural Abstraction Example 1

One or more objects may perform the same repetitive motions. The animating of one swimming fish requires the writing of many repetitive procedures which must then be repeated for each fish in the school, taking up a lot of space in myFirstMethod.



Procedural Abstraction Example 2

Sometimes, a procedure needed to perform an action is not available by default. For example, a bird needs to fly, but a fly procedure is not available for bird objects.



When Procedural Abstraction Occurs

Procedural abstraction can occur before or after programming statements are created.

However, by developing the storyboard first, the programmer can more easily identify the procedures that will be needed before the programming begins.

Examples of When to Declare a Procedure

Declare a procedure when:

- Motions do not have a default procedure, such as a bird flying.
- Motions need to be used by multiple objects or classes, such as all quadrupeds hopping up and down.
- Singular motions require many programming statements, such as a person moving body parts to walk.

Declaring a Procedure Example 1

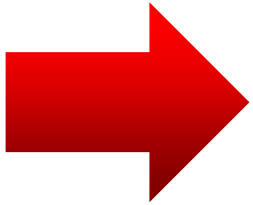
The bird flies by, turning its shoulders and moving forward simultaneously. This repetitive motion can be extracted into its own flying procedure.

Do together:

- *Bird turns right shoulder backward*
- *Bird turns left shoulder backward*
- *Bird moves forward*

Do together:

- *Bird turns right shoulder forward*
- *Bird turns left shoulder forward*
- *Bird moves forward*

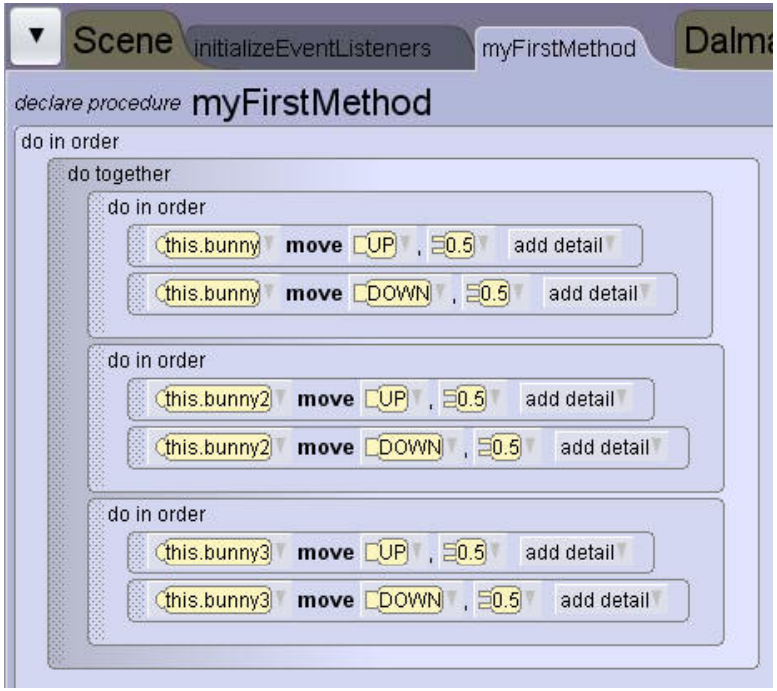


Do together:

- *Bird flies*
- *Bird moves forward*

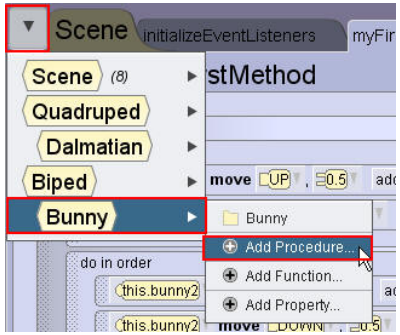
Declaring a Procedure Example 2

Each bunny moves up and down in order to simulate a hopping motion. This repetitive motion used by all bunny objects could be extracted into its own hop procedure.



Steps to Declare a Procedure

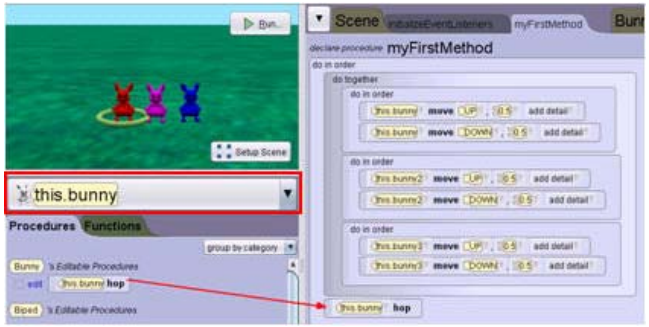
1. From the class hierarchy, select the class that should inherit the procedure. All subclasses will then inherit the procedure as well. (These are indented underneath their superclass.)
2. Click the Add Procedure... button.



3. Specify a name for the procedure and then click OK.
4. A new tab with the name of the procedure will open. Write the code for the procedure. Add the new procedure to myFirstMethod right away so that when you test the animation in myFirstMethod, you are also testing the animation in the new procedure.

Steps to Add Procedure to myFirstMethod Tab Before Programming

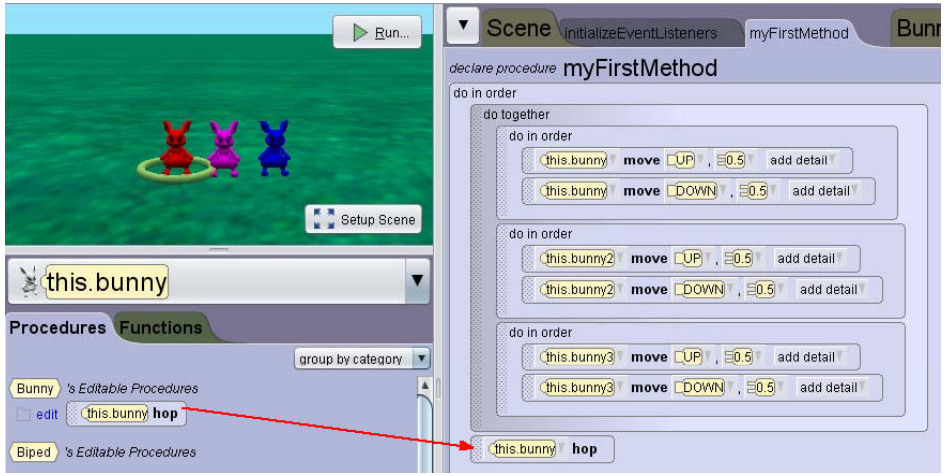
1. Click the myFirstMethod tab.
2. Select the instance for which you are coding the procedure from the instance menu.
3. In the Procedures tab, locate the declared procedure and drag it into myFirstMethod.



4. Go back to the procedure by clicking the tab at the top of the Code editor with the procedure's name.

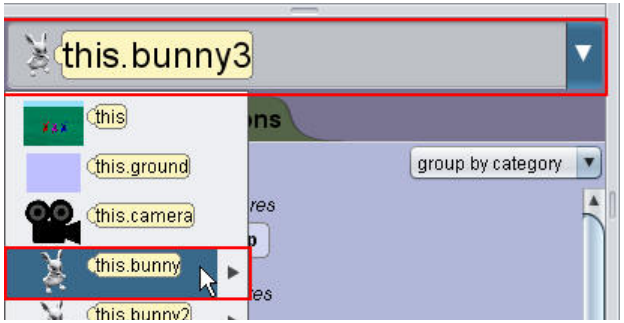
Steps to Add a Declared Procedure to Your Code

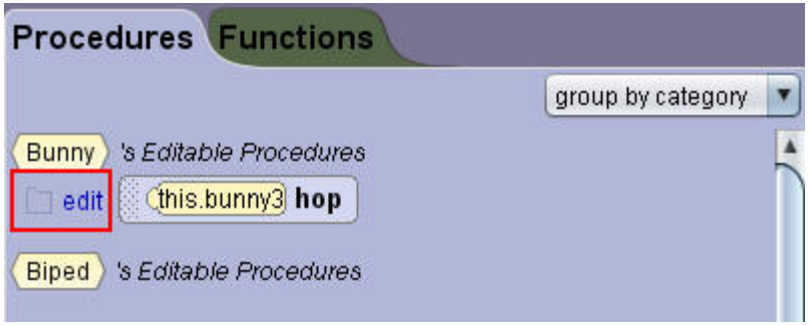
1. Select the instance of the object to which you wish to add the declared procedure.
2. Find the declared procedure listed under the Procedures tab in the methods panel. The declared procedures are available in the procedures tab for all objects that inherit them.
3. Drag the declared procedure to the code.



Steps to Access and Edit Declared Procedures

1. In the instance menu, select the instance where the procedure was declared.
2. Click Edit to the left of the procedure name.

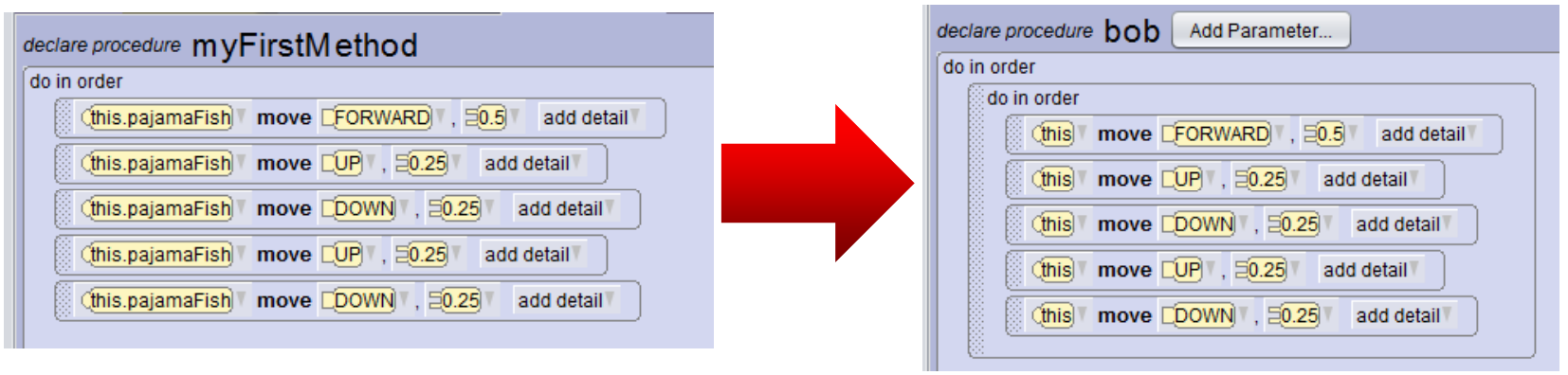




3. Create or edit the programming instructions for the procedure.
4. Click the Run button to test the procedure and debug as necessary.
5. When finished editing the procedure, click the myFirstMethod tab to return.

Identify Opportunities for Declared Procedures

As you program, continually identify opportunities to declare procedures using procedural abstraction techniques. For example, in myFirstMethod, the fish bobs up and down repeatedly. A separate “bob” procedure should be declared as a result.



Procedural Abstraction and Using the Clipboard

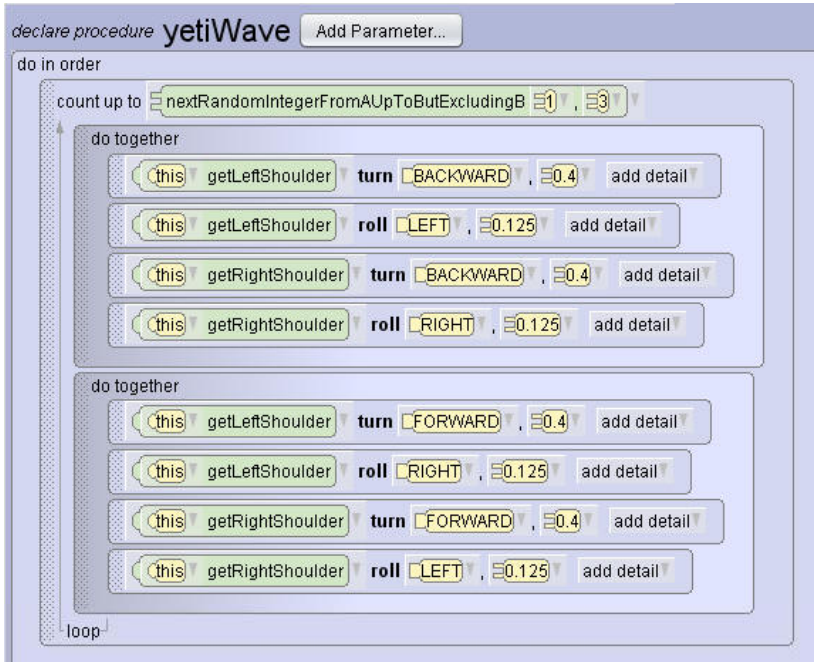
After you have written many programming instructions in the myFirstMethod tab, you may determine that the code would serve your program better if it were in a declared procedure.

To save time, you can drag the programming instructions onto the clipboard icon. Then, after creating the declared procedure, you can drag the programming instructions from the clipboard into the declared procedure.

Use Inherited Procedures

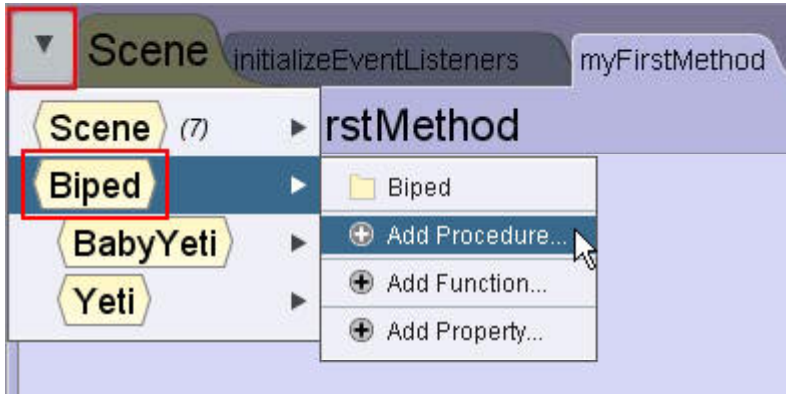
Procedures declared at the superclass level are available to the other objects in that class.

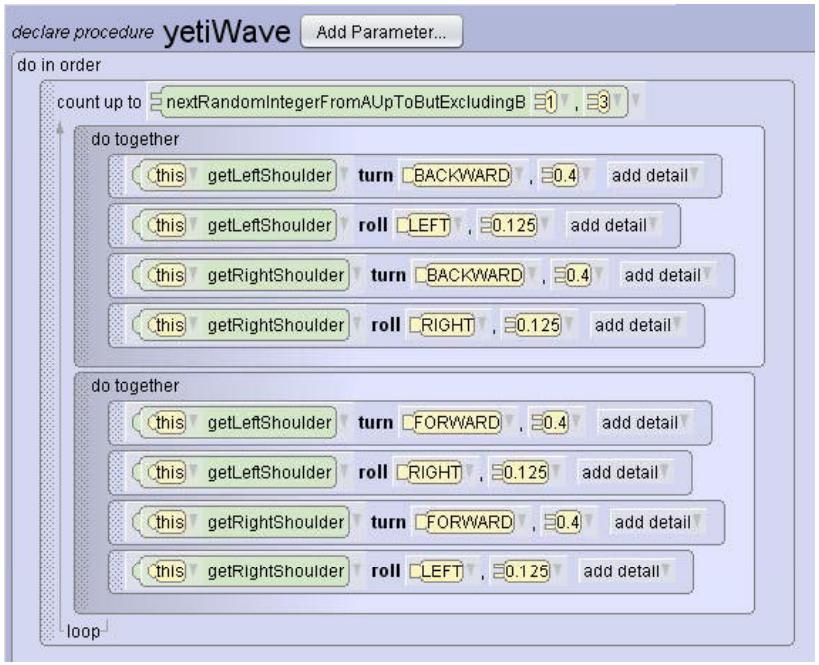
For example, a "yetiWave" procedure created to make a baby Yeti wave his arms in the air could be used to make an adult Yeti's arms wave in the air.



Declare Procedure at Superclass Level

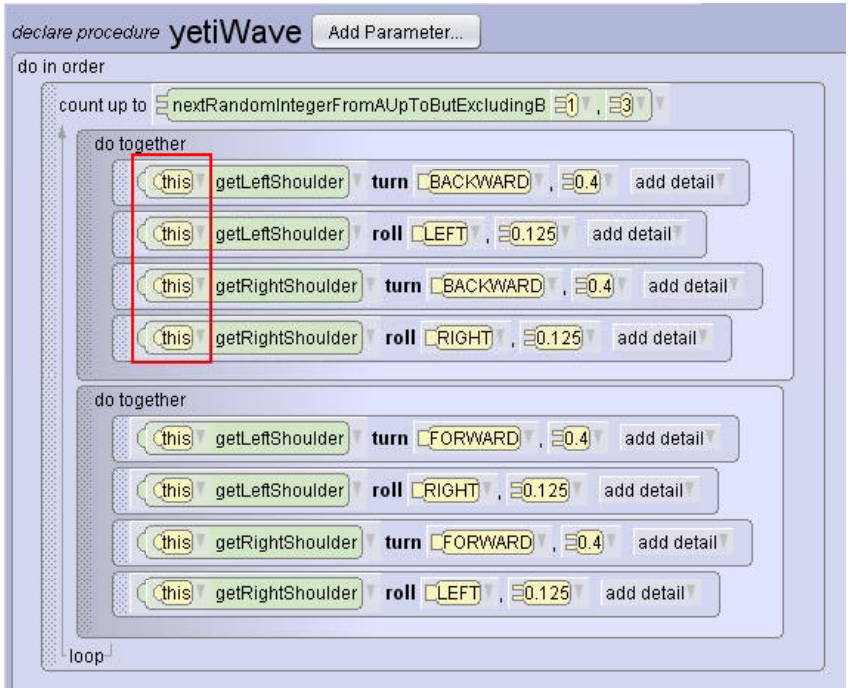
Declare the "yetiWave" procedure at the biped superclass level so that the baby Yeti and the adult Yeti can both use the procedure.





this Object Identifier

When a declared procedure is created, the object identifier, *this*, is used to indicate that the instance calling the procedure is *this*.



this Object Identifier Example

If the baby yeti is selected in the instance menu, then the YetiWave procedure is added to myFirstMethod; *this* in the declared procedure refers to the baby yeti.

If the adult yeti is selected in the instance menu, and the YetiWave procedure is added to myFirstMethod, *this* in the declared procedure now refers to the adult yeti.

In essence, *this* always refers to the instance of the class that is calling the procedure.

Summary

In this lesson, you should have learned how to:

- Compare and define an animation and a scenario
- Write a storyboard
- Flowchart a storyboard
- Describe inheritance and how traits are passed from superclasses to subclasses

Summary (cont.)

In this lesson, you should have learned how to:

- Describe when to implement procedural abstraction
- Demonstrate how to declare a procedure
- Identify and use procedural abstraction techniques to simplify animation development