


Basic Concepts

- **Class:** ‘blueprint’ or template for creating an object
 - describes features and actions
- **Object:** any thing that can be identified as unique from other things
 - has **properties:**
 - width, height, color, location
 - can perform actions (**methods**):
 - associated actions it can perform
 - tasks it can carry out



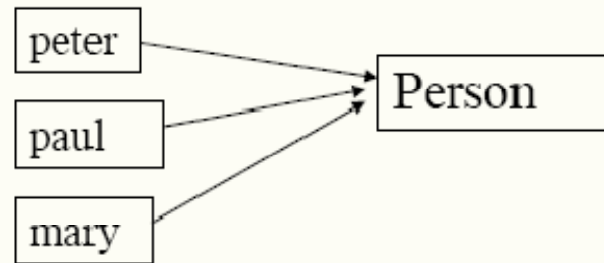


Vocabulary

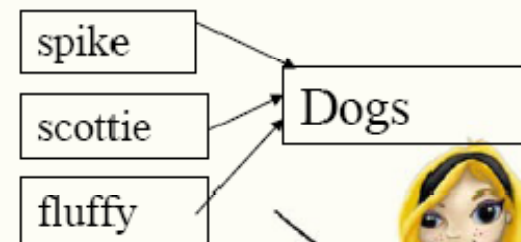
- The action of using the class to create an object is called *instantiating the class*.
 - The new object is called an *instance of the class*.
 - **Virtual world:** an animation is implemented in Alice as a *virtual world*
 - An *object* is a unique entity within a virtual world that is the result of instantiating a class.
- 

Class

- 🌐 Objects are categorized into classes



- 🌐 Each object is an **instance** of the class.




- 🌐 An Alice class is a “master object” stored on the disk; you command Alice to make copies and put them in your world.





Methods

- Program: *a set of instructions that tell the computer what to do*
 - An **instruction** is a statement that executes (is carried out by the computer at runtime).
 - In Object Oriented Programming, an instruction targeting an object is said to be a *method*.
 - To get an object to perform a behavior, send that object a message that asks it to produce that behavior
- 

Control Structure

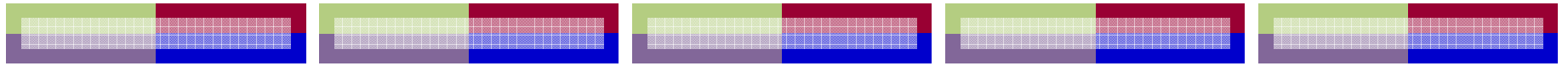
🌐 A **control structure** is a statement that controls which instructions are executed and in what order.

▶ In previous worlds, we used:

💡 Do in order

💡 Do together

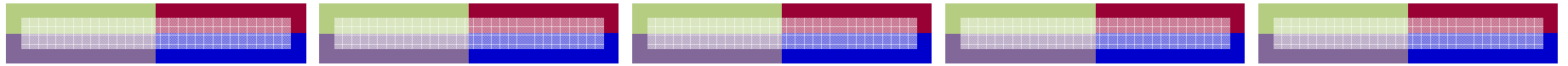




Methods

- Programs consist of a set of statements
- Method: behavior-producing message
- Objects have predefined methods for basic tasks
- Methods may also be created by Alice developers
- Two reasons for building your own methods
 - To organize your story into more manageable pieces
 - To provide an object with additional behaviors






World Methods for Scenes and Shots

- Scene: segment of a story
- Shot: part of a scene from a given camera position
- User stories can be divided into scenes and shots
 - A convenient technique for completing a project
- Divide and conquer approach to building user stories
 - Break a big problem into smaller problems
 - Solve each of the smaller problems
 - Combine the smaller problems into a solution





Methods for Scenes

- Scenario: develop a user story with three scenes
 - Convention for naming methods
 - Name should be a verb or verb phrase
 - Name should describe what the method does
 - Creating the first new method
 - Select the **world** object
 - Click the **create new method** in the details area
 - Enter **playScene1** in the **New Method** dialog box
 - Check new method by sending **say()** to **ground**
 - First test fails because **my_first_method()** is empty
- 



Methods for Scenes (continued)

- How to fix the first bug
 - Click on the tab for **my_first_method**
 - Drag a **doInOrder** control to the top of the pane
 - Click on **world** in the *object tree*
 - Drag **playScene1()** into the **doInOrder** statement
- Extend technique used to build **playScene1()**
 - Add two methods: **playScene2()**, **playScene3()**
 - New method sends a **say()** message to the ground
 - New Methods are called in **my_first_method()**



Methods for Shots

- Scenes can be divided into shots
- Shots can be further divided into pieces
- Reasons for using scenes, shots, and pieces
 - To create a program that reflects the user story
 - To create a program that has a modular design
- Example of a scheme using scenes and shots
 - Level 1: **my_first_method()**
 - Level 2: three methods for three scenes
 - Level 3: four methods for four shots in Scene 2



Methods for Shots (continued)

- Implementing the scheme
 - Test each shot in Scene 2 using a **say()** method
 - Call the four shot methods from **playScene2()**
 - Call three scene methods from **my_first_method()**
- Structure diagram reflects organization of user story
- All objects added to **world** become part of **world**
- Scene and shot messages are stored in the **world**
- World method: affects behavior of multiple objects

Methods for Shots (continued)

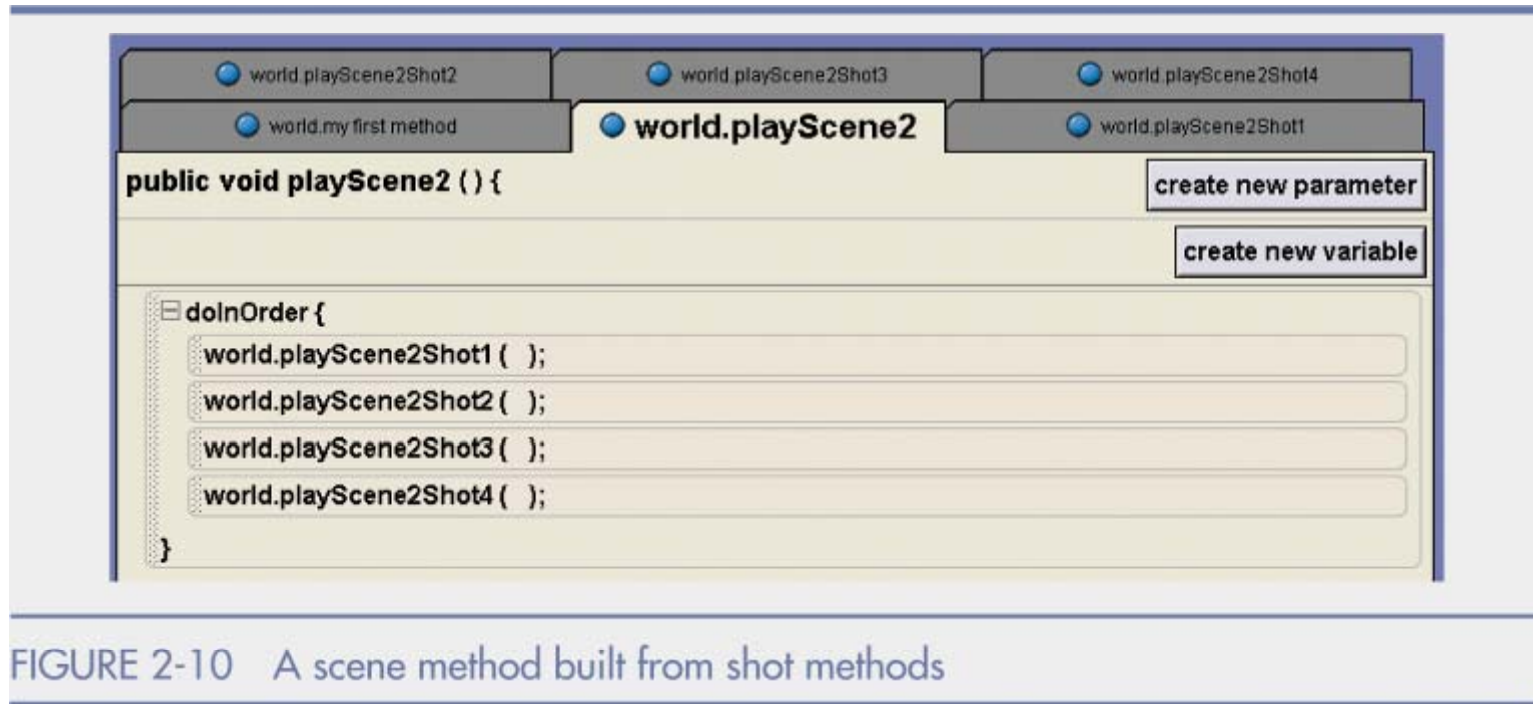


FIGURE 2-10 A scene method built from shot methods

Methods for Shots (continued)

