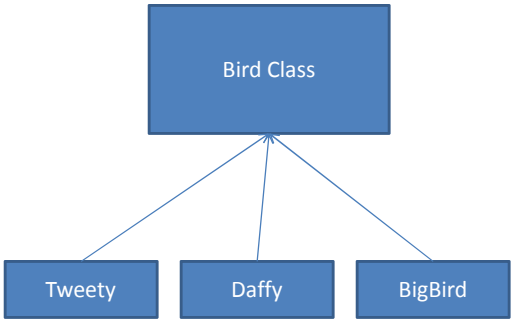
Alice Methods and Classes

Methods and Classes

- **Methods**
  - Coordinated sequence of instructions carried out when requested (e.g. move, turn to, etc.)
- **Class**
  - A class defines an object, in our cases the 3D models
  - Classes contain
    - Data: e.g. position, color
    - Methods that pertain to the specific object
  - A class is a blueprint that tells Alice how to create, display, and manipulate the object
  - When an object is created we call this **instantiation** and we create an **instance** of the class

Classes vs. Objects



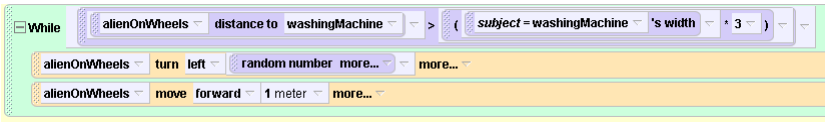
To do: Add three birds to an Alice world

Methods

- Methods must be associated with some object
  - Might be the world object
  - Should only deal with things specific to the object for which it is defined
- Methods give us a way to divide a program up into small, manageable pieces that work together
  - Process called **abstraction**
  - Makes programs easier to write and debug than one giant code block
  - The process of breaking a problem into methods is called **stepwise refinement**
- Methods may take **parameters**
  - Input data that the method should operate on, like a function

World Methods

- For example, consider our world where we had the alien on wheels randomly try to get close to the washing machine



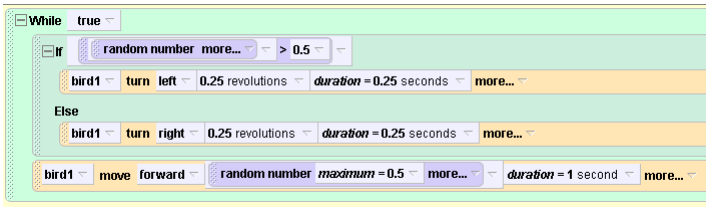
- This is not too complex on its own, but when part of a larger script it can get messy

World Methods

- We can create a method named “AlienSeek” that runs this code
- Under “World”, click “Methods” and “Create New Method” and name it “AlienSeek”
  - Put code into the new method
- To invoke it, we just drag out “AlienSeek” from the World methods where we want to use it
  - Greatly simplifies the invoking code and also gives it a name to make it more meaningful
- To do: Create method in Alice

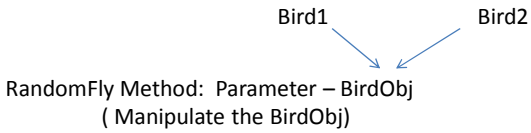
Another Method Example

- Create a new world with three Birds
- Create a World method called “RandomFly”
- While true
  - With 50% chance
    - Make Bird1 turn to the left randomly up to 0.25
  - Otherwise
    - Make Bird1 turn to the right randomly up to 0.25
  - Go forward some random distance from 0 to 0.5



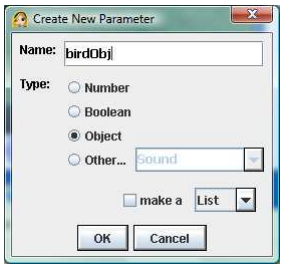
Problem: How about for all birds?

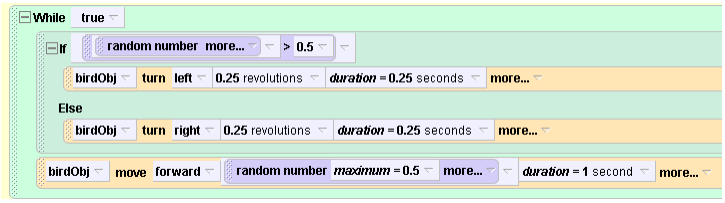
- What if we want all birds to fly? The new method only moves Bird1 and it would be nice to avoid having to copy the method for Bird2, Bird3 when it should be the same
- Solution
  - Make the method take a **parameter** that specifies which bird object to move



Creating a Parameter

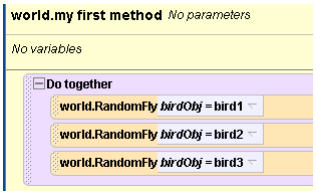
- Click “Create new Parameter” button in the method
- Give it a name like birdObj
- Drag birdObj in place of Bird1





Invoking the Method

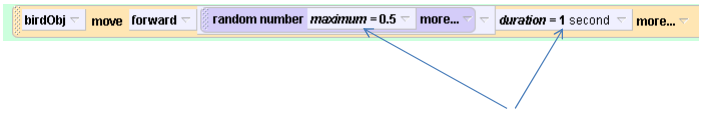
- We need to specify the parameter when we invoke the method



- All the birds now fly together!

Refinement

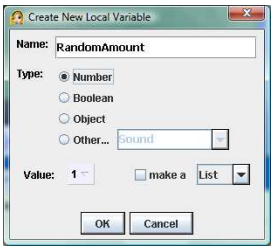
- Maybe we want the birds to fly random lengths from 0 to 1 but at the same speed
- Problem: Duration should then match the distance



We want these the same value
But if we pick another random number it may be different

Solution: Variable

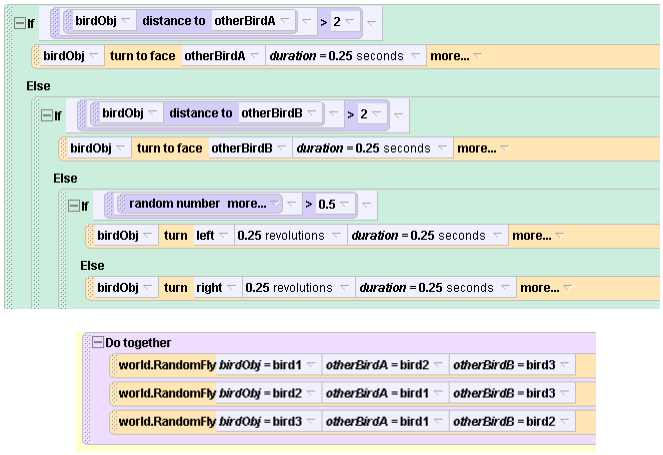
- Create a variable that holds the random number, then we can reference that same variable in both places
- Click “Create New Variable”, give it a name, and assign its value



Further Refinement: Flock

- Let's keep the birds from flying too far away from each other
- Not very elegant but workable solution:
  - If distance to otherBirdA > 3 then
    - Turn to face otherBirdA
  - Else
    - If distance to otherBirdB > 3 then
      - Turn to face OtherBirdB
    - Else
      - Do the random direction selection
  - Fly forward random amount
- Can do this by adding two more parameters for the other birds

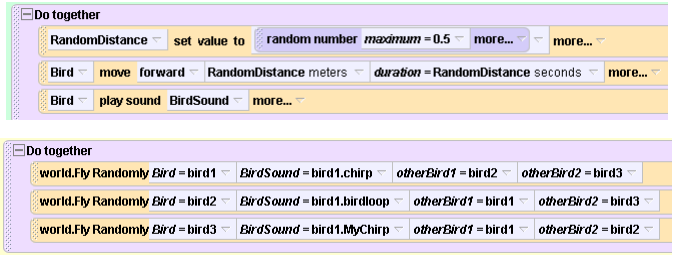
Bird Flock



More elegant solutions exist, but we will skip them here

Bird Sounds

- What if we would like each bird to play a different sound as it flies? We can send that in as a separate parameter too.
  - Can record a new sound for Bird1
- Create a new parameter, select Other, and pick "Sound" as the type of parameter

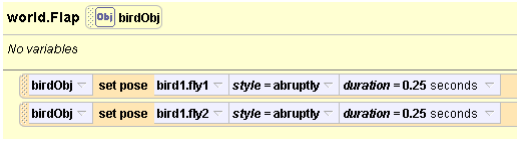


Class Methods

- We can also define methods for specific classes
- Such methods should ONLY apply to behaviors of the selected class / object
  - E.g. Select Bird1
  - Shouldn't add anything relating to other birds/objects inside this method
  - Can add a "FlapWings" method
    - Set the Pose to Fly1
    - Set the Pose to Fly2
- Notice that the new method only applies to Bird1
  - If you want the method for all birds, you can save Bird1 as a new object, then load the new object in and you will be creating multiple instances of the Bird1 class

All birds flapping

- To make all the birds flap we can make a World method that takes a Bird object as input and flaps its wings



Exercise : Magic Act

- A magician is performing a levitation illusion in which objects rise magically into the air. The magician points at a rabbit and it floats; then he points at his assistant and she floats.
- Use a single method with parameters to communicate which object is to float and the distance the object is to move upward and downward.