ORACLE Academy

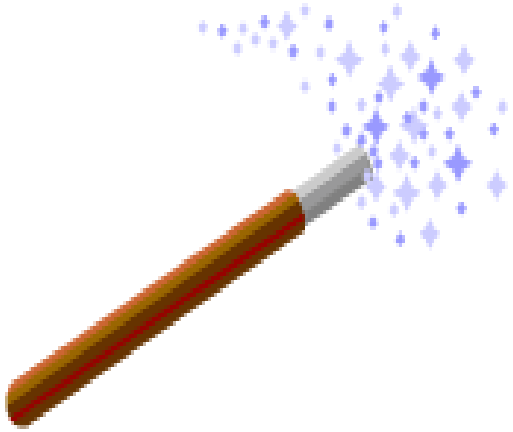
Correlating Java Methods, Classes, and Other Structures with Alice 3 Tools

What Will I Learn?

Objectives

In this lesson, you will learn how to:

- Describe a method, class, and instance
- Describe a scenario where an IF control structure would be used
- Describe a scenario where a WHILE control structure would be used
- Recognize the syntax for a method, class, function, and procedure
- Describe input and output



Why Learn It?

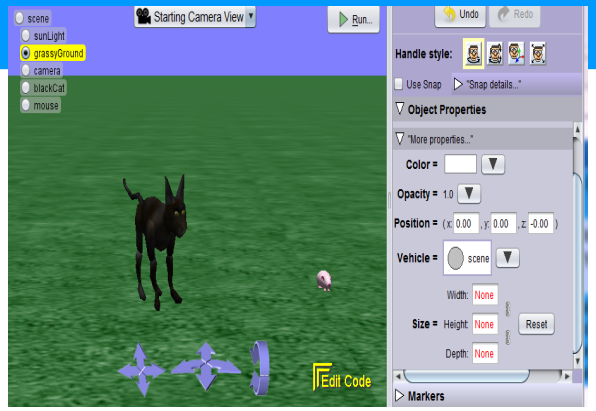
Purpose

In Alice 3, you search for classes in the gallery and add instances of classes to the scene. In the code editor, you use procedures (known in Java as “methods”) and functions to program objects to move and act. You also use control structures to refine your objects' movements.

As you develop your animation in Alice 3, you are learning the fundamentals of the Java programming language. This lesson helps you understand how the methods, classes, and other structures that you have used in Alice 3 correspond to the Java programming language.

Alice versus Java

Alice 3	Java
3D programming environment	Programming language; can be edited using integrated development environment (IDE).
Can be used to create animations, interactive games, or videos.	Can be used to create applications that run on any platform, including the web.
Drag and drop coding reduces syntax errors	Object oriented language reduces complexity because objects model real world objects, allow for re-use and easier maintenance.




```
public class HelloWorld
{
    public static void main()
    {
        System.out.println( "Hello World!" );
    }

}
```

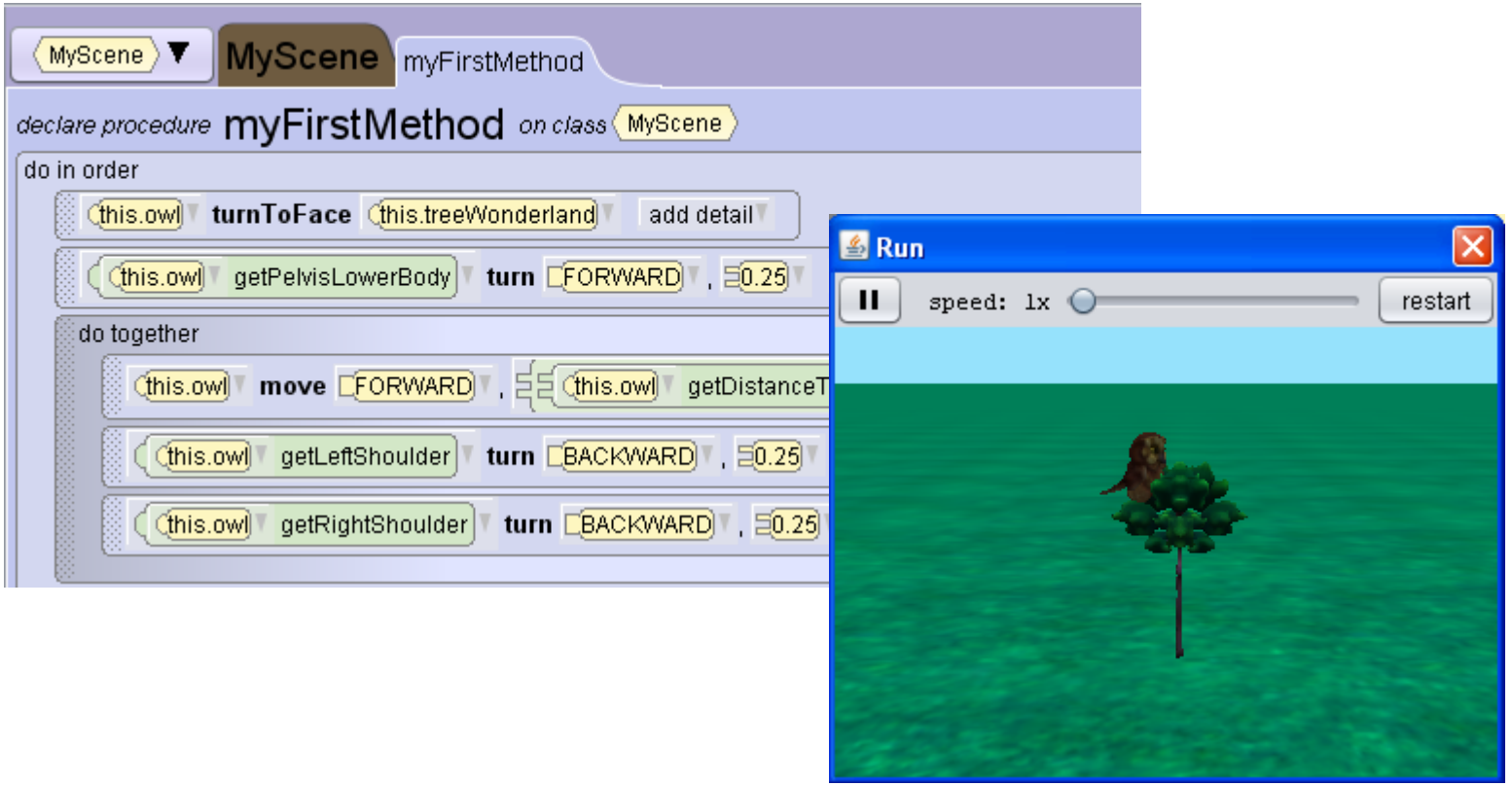
Methods (aka Procedures) in Alice

In Alice 3, a procedure is a piece of code that sends a message to an object asking it to perform an action. A procedure does not return a value. Alice 3 has a set of procedures for each class.

TheMethods panel lists all procedures, functions and properties for a selected object. This area contains all the built-in and user-defined methods and settings you can access for each object in your scene.

Methods (aka Procedures) in Alice (cont.)

You can also create ("declare") your own procedures in Alice.



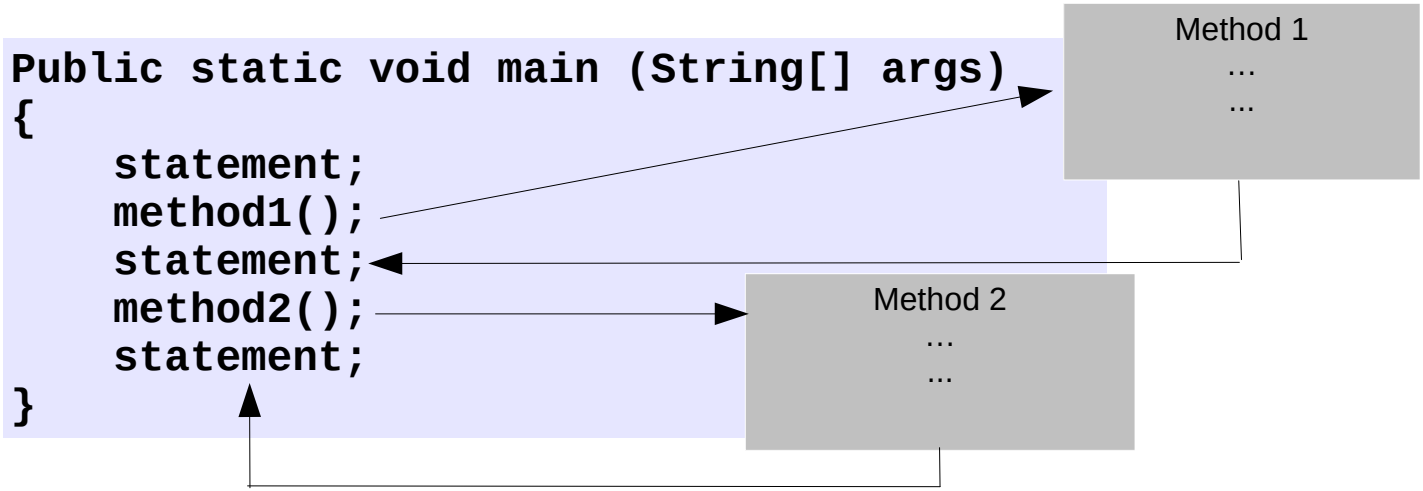
Methods in Java

A method in Java is the same as a procedure in Alice. A method is a piece of code that sends a message to an object asking it to perform an action. A method belongs to a class.

A method is referred to by name and can be called at any point in a program using the method's name. When a method name is encountered in a program, that method is executed.

Methods in Java (cont.)

When the method is finished, execution returns to the area of the program code from which it was called, and the program continues on to the next line of code.



Methods in Java (cont.)

There are three things you need to decide for any method: what it does, what inputs it needs, and what answer it gives back. Here is the Java syntax for a method.

```
[modifiers] dataType methodName(parameterList) {
 methodBody
 return result;
}
```

Methods in Java (cont.)

- Modifiers are optional and can be public, private, protected, or left blank; `dataType` is a type of data, like `int`.
- `methodName` is the name of your method.
- `parameterList` is a comma-separated list of parameter names with their data types; each parameter is listed with its data type first, then its name.
- `methodBody` is the set of statements that perform the task.
- `return` is a keyword that sends the result value back to the code that called the method.

```
[modifiers] dataType methodName(parameterList) {
 methodBody
 return result;
}
```

Methods in Java (cont.)

Below is a method called findMax.

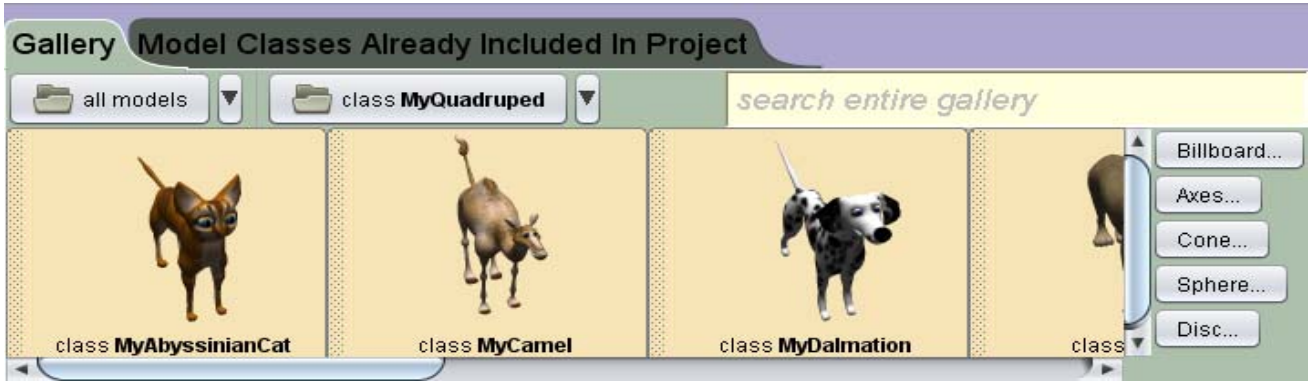
```
public class TestFindMax {
   /** Main method */
   public static void main(String[] args) {
      int i = 5;
      int j = 2;
      int k = findMax(i, j);
      System.out.println("The maximum between " + i +
                         " and " + j + " is " + k);
}
/** Return the max between two numbers */
public static int findMax(int num1, int num2) {
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

Classes in Alice

A class is a specification, such as a blueprint or pattern and a set of instructions, of how to construct something.

Example

A Dalmatian is a dog. When you add a dalmatian to a scene, it will have the basic properties of the Dalmatian class: four legs, two ears, a white and black spotted coat, the ability to walk, etc.



Classes in Java

```
["public”] ["abstract"|"final"]"class" Class_name
  ["extends" object_name] ["implements" interface_name]
"{"
// properties declarations
// behavior declarations
"}"
```

- The first group is optional and refers to the visibility from other objects. Public means visible everywhere. The default is package or visible within the current package only.

Classes in Java (cont.)

- The second group is optional and defines whether the class can be inherited or extended by other classes. Abstract classes must be extended and final classes can never be extended by inheritance. The default indicates that the class may or may not be extended at the programmers discretion.
- Class_name is the name of the class.
- The third option of extends is related to inheritance.
- The fourth option of implements is related to interfaces.

Classes in Java (cont.)

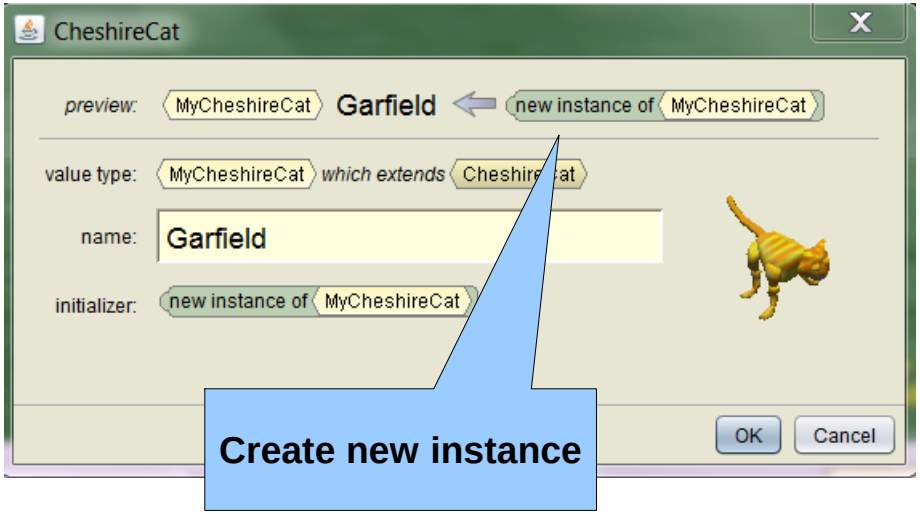
Below is code for creating a class called Cat in Java.

```
class Cat{

   int catAge;
   public Cat(String name){
      System.out.println("Name is :" + name );
   }
   public void setAge(int age){
       catAge = age;
   }
   public int getAge( ){
       System.out.println("Age is :" + catAge );
       return catAge;
   }
   public static void main(String []args){
      Cat myCat = new Cat( "Garfield" );
      myCat.setAge( 6 );
      myCat.getAge( );
      System.out.println("Variable Value :" +
      myCat.catAge );
   }
}
```

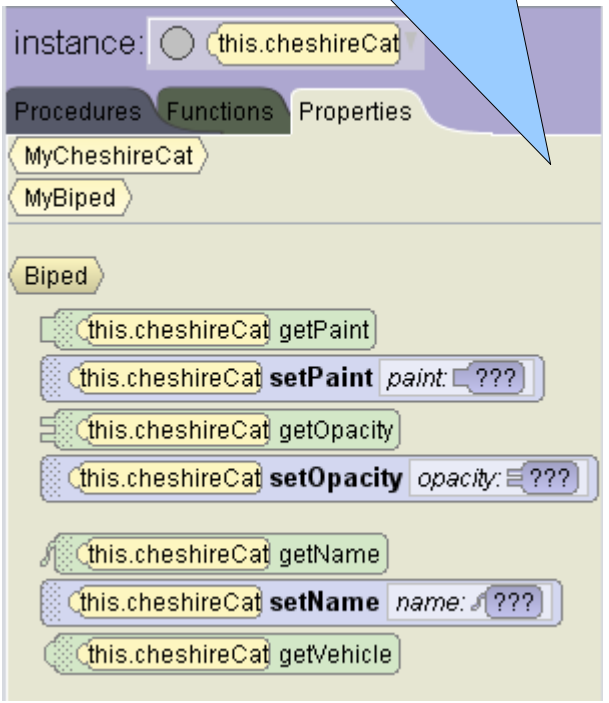
Instances in Alice

An object is an instance of a class.

For example, when you add a cheshire cat to a scene, you are creating an instance of the CheshireCat class.



Properties of cheshirecat instance that can be changed



Instances in Java

Recall the cat class. Note how in the main method, the myCat instance of the cat class is created. An instance of a class is created using the new operator followed by the class name.

```
class Cat{

   int catAge;
   public Cat(String name){
      System.out.println("Name is :" + name );
   }
   public setAge( int age ){
       catAge = age;
   }
   public getAge( ){
       System.out.println("Age is :" + catAge );
       return catAge;
   }
...continued on next slide
```

Instances in Java (cont.)

Recall the cat class. Note how in the main method, the myCat instance of the cat class is created. an instance of a class is created using the new operator followed by the class name.

```
...Continued from previous slide

   public static void main(String []args){
      Cat myCat = new Cat( "Garfield" );
      myCat.setAge( 6 );
      myCat.getAge( );
      System.out.println("Variable Value :" + myCat.catAge );
   }
}
```

Create new instance

Properties of instance that can be changed

Control Structures

Control structures allow you to change the order of how the statements in your programs are executed.

There are two types of control structures:

Type	Description	Example
Decision control structures	Allow you to select specific sections of code to be executed.	If .. Then .. Else
Repetition control structures	Allow you to execute specific sections of the code a number of times.	WHILE loop

Both Alice and Java allow for these types of control structures.

IF Control Structures

IF control structures are statements that allow you to select and execute specific blocks of code while skipping other sections.

An IF control structure has the form:

```
if (boolean_expression) {
    doSomething();
} else {
    doSomethingElse();
}
```

IF Control Structures (cont.)

Example:

- If a student receives a score of 90% or greater on their test, then give them an “A”.
- If a student receives a score that is greater than or equal to 80% and less than 90%, then give them a “B”.
- If a student receives a score that is greater than or equal to 65% and less than 80%, then give them a “C”.
- If a student receives less than 65%, then give them an “F”.

IF Control Structures (cont.)

Below is the code implemented in Java.

```
public class Grade {
public static void main( String[] args )
{
  double grade = 89.0;
  if( grade >= 90 ){
  System.out.println( "A" );
  }
  else if( (grade < 90) && (grade >= 80)){
  System.out.println("B" );
  }
  else if( (grade < 80) && (grade >= 65)){
  System.out.println("C" );
  }
  else{
  System.out.println("F");
  }
  }
}
```

WHILE Control Structures

A While control statement or While loop is a Java statement that allows you to execute specific blocks of code a number of times.

A While loop is a statement or block of statements that is repeated as long as some condition is satisfied. The condition is tested before each loop.

A While loop has the form:

```
while( boolean_expression ){
statement1;
statement2;
. . .
}
```

WHILE Control Structures (cont.)

Below is an example of the while control structure:

```
class WhileDemo {
    public static void main(String[] args){
        int count = 1;
        while (count < 11) {
            System.out.println("Count is: "
                               + count);
            count++;
        }
    }
}
```

Input and Output

Java programs work on many platforms. They can be simple programs that run from the command line, or they can have complex graphical user interfaces.

When learning to program, you will create programs that use the command line for its input and output exclusively.

After you have gained more experience, you can build graphical user interfaces and learn about the libraries required to build them.

Input and Output (cont.)

You can print to the screen via the following code:

```
System.out.println("Hello World!”);
```

`System.out.println()` is a piece of code that someone else wrote.

Input and Output (cont.)

You can get user input using the java.io package.

```
import java.io.*;
public class ReadString {
   public static void main (String[] args) {
      System.out.print("Enter your name: ");
      BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

      String userName = null;
      try {
         userName = br.readLine();
      } catch (IOException ioe) {
         System.out.println("IO error trying to read your
         name!");
         System.exit(1);
      }
      System.out.println("Thanks for the name, " + userName);
   }
}
```

Terminology

Key terms used in this lesson included:

Class

Control structure

IF control structure

Instance

Method

WHILE control structure

Summary

In this lesson, you learned how to:

- Describe a method, class, and instance
- Describe a scenario where an IF control structure would be used
- Describe a scenario where a WHILE control structure would be used
- Recognize the syntax for a method, class, function, and procedure
- Describe input and output

Practice

The exercises for this lesson cover the following topics:

- Evaluate the Java syntax of a simple animation and define the methods, classes, and instances
- Create an animation to demonstrate the use of a control structure
- Create an animation to demonstrate the use of a looping structure
- Correlate the Java syntax with animation movements
- Create a Java syntax cheat sheet
- Present animation examples for IF and WHILE control structures