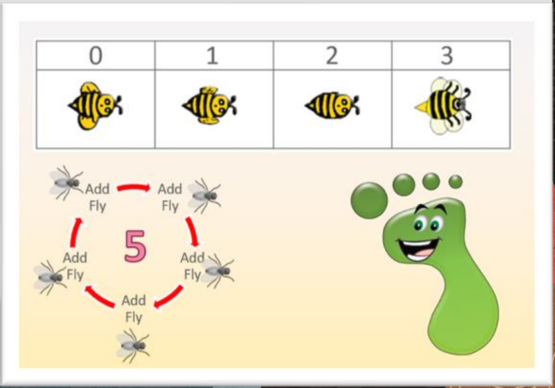
ORACLE
Academy

Java Fundamentals

3-10
Loops, Variables, and Arrays

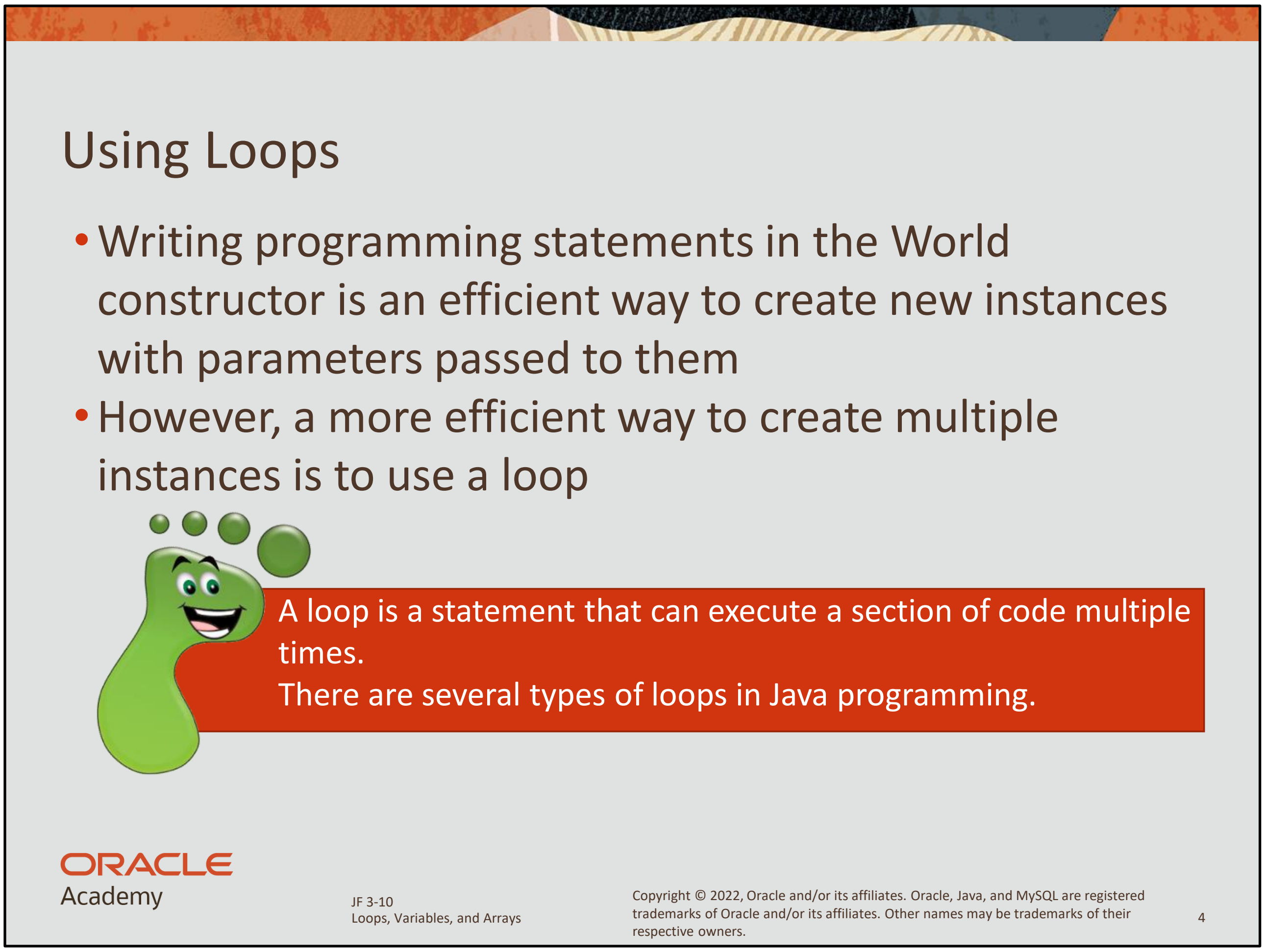


ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Create a while loop in a constructor to build a world
  - Describe an infinite loop and how to prevent one from occurring
  - Use an array to store multiple variables used to create a world
  - Create an expression using logic operators
  - Describe the scope of a local variable in a method

ORACLE Academy



Using Loops

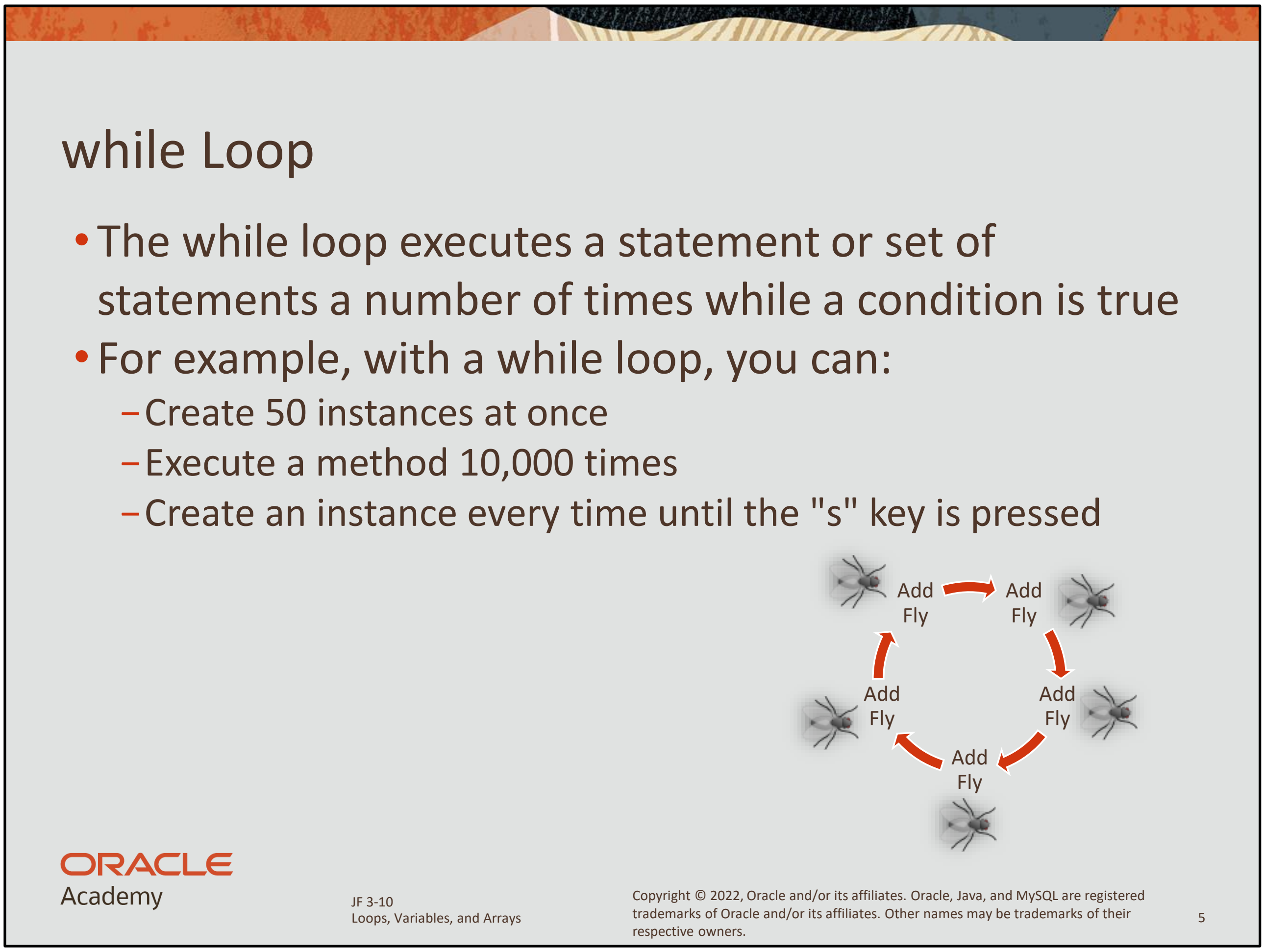
- Writing programming statements in the World constructor is an efficient way to create new instances with parameters passed to them
- However, a more efficient way to create multiple instances is to use a loop

A loop is a statement that can execute a section of code multiple times.
There are several types of loops in Java programming.

ORACLE Academy

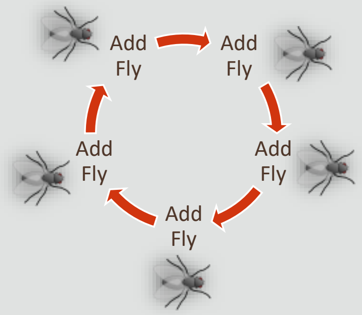
Repeating lines of codes in programming is a common task. Adding a loop around a section of code will often be seen in programming code.

Most programming languages have 3 basic loop methods. We will look at all of these during the course.



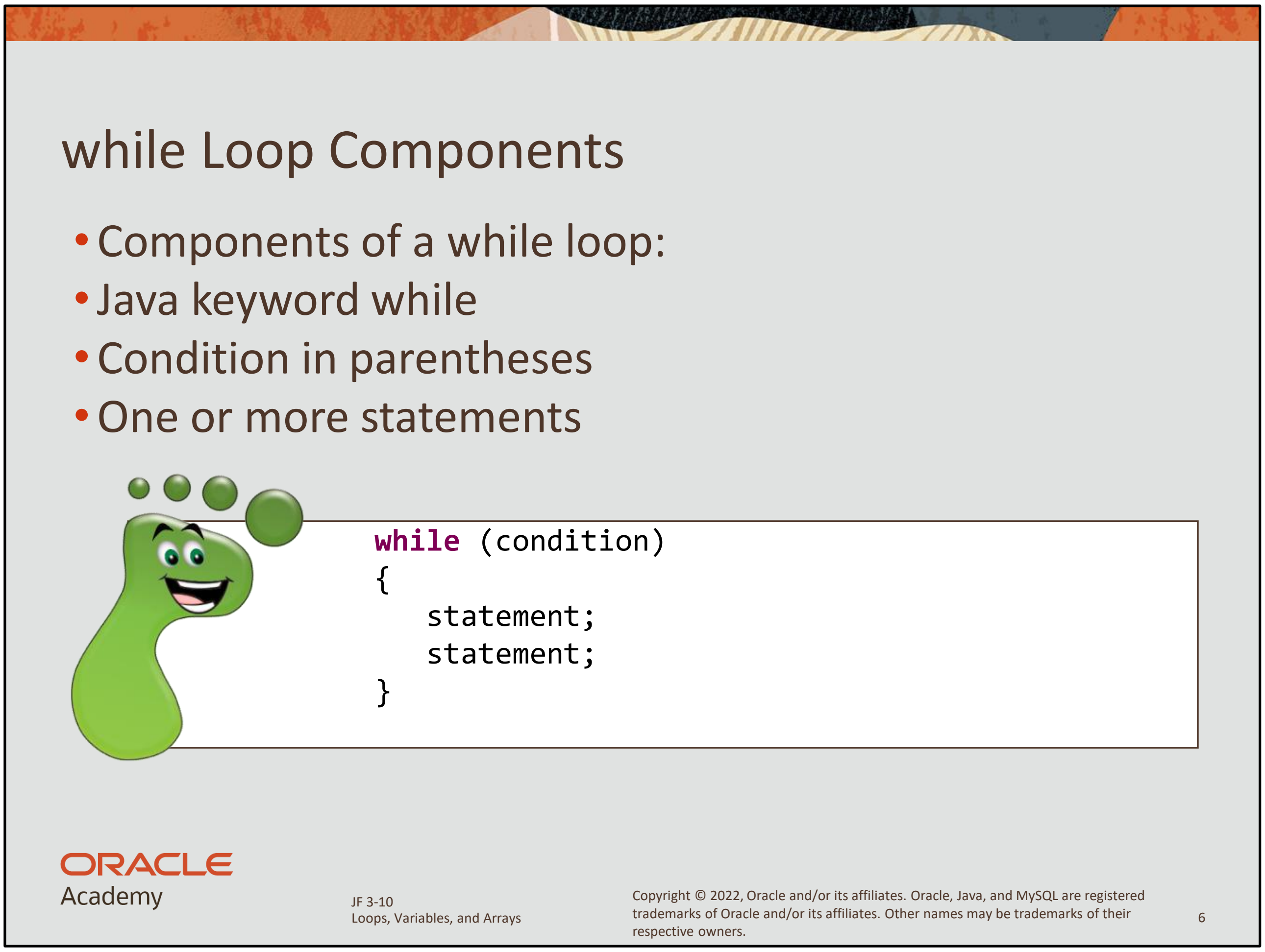
while Loop

- The while loop executes a statement or set of statements a number of times while a condition is true
- For example, with a while loop, you can:
  - Create 50 instances at once
  - Execute a method 10,000 times
  - Create an instance every time until the "s" key is pressed



ORACLE Academy

We can use any of the loops for any occasion, but some are better suited to particular scenarios.



while Loop Components

- Components of a while loop:
- Java keyword while
- Condition in parentheses
- One or more statements

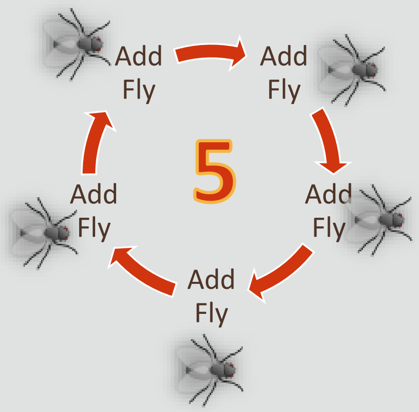
```
while (condition)
{
    statement;
    statement;
}
```

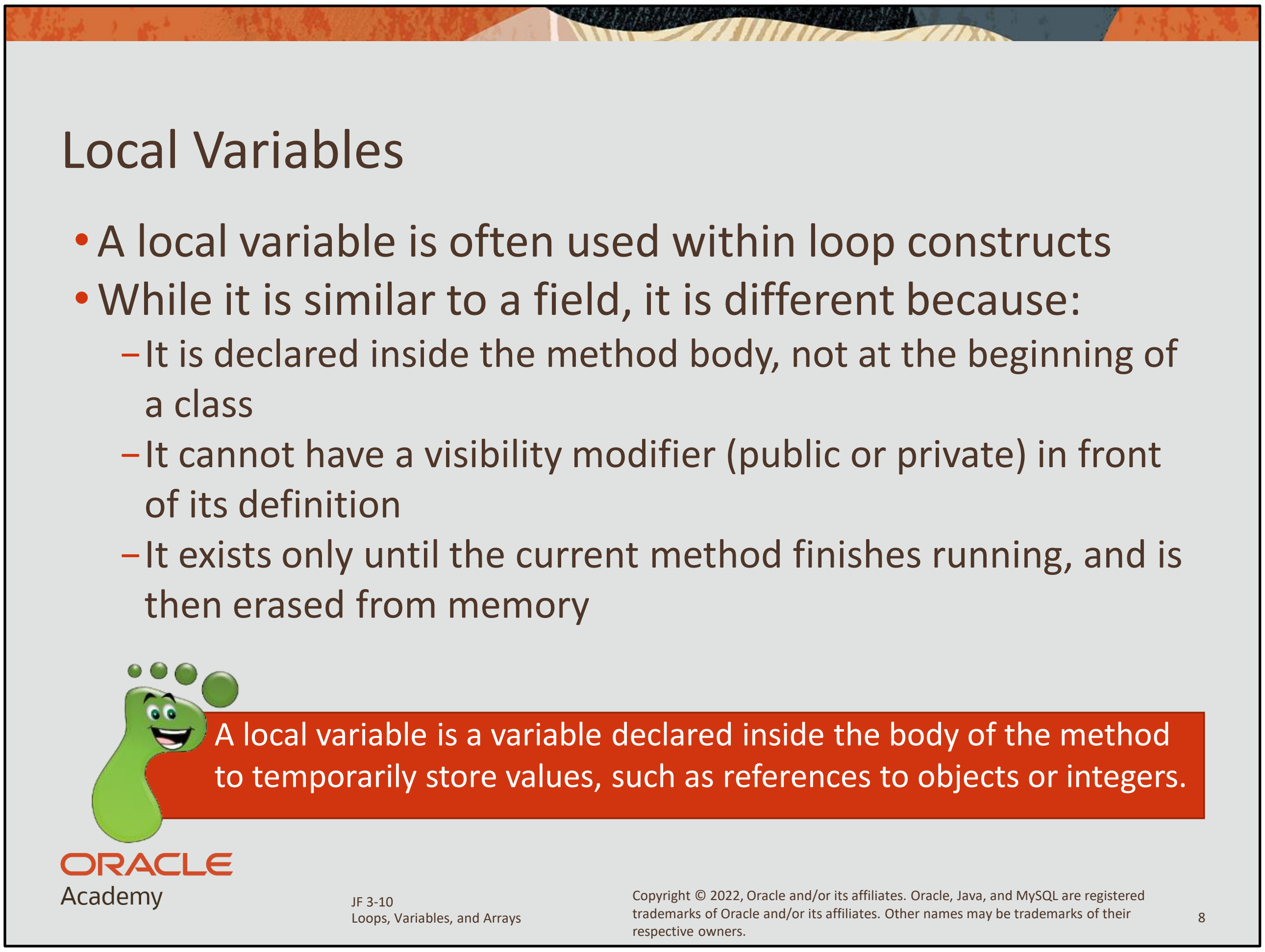
ORACLE
Academy

Remember the condition must only return true or false.

Control Execution of while Loop

- Components to control how many times the while loop is executed:
- Loop variable:
  - A counter that tells how many times to execute the loop (often named i)
- Control operators
- Local variable





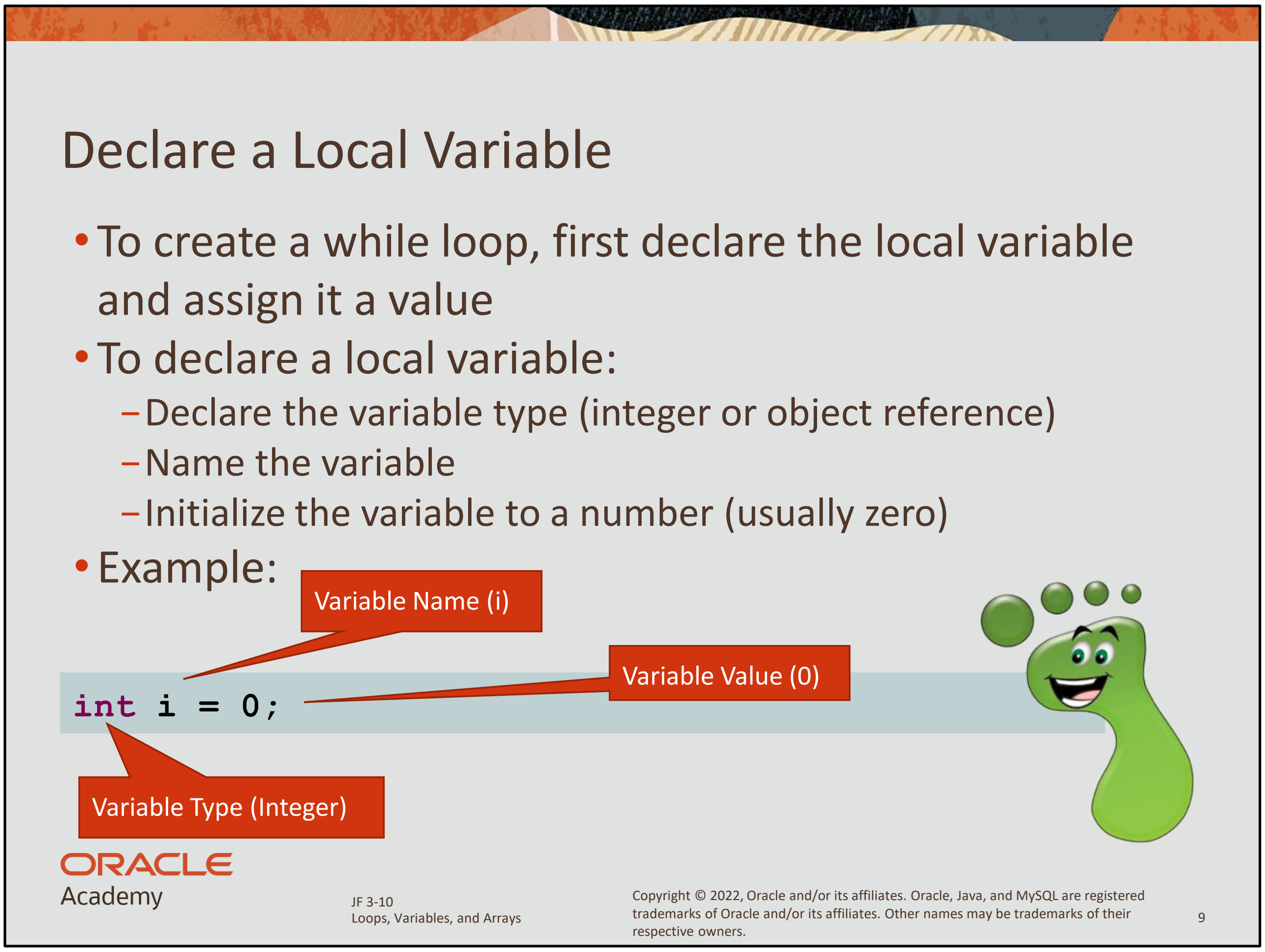
Local Variables

- A local variable is often used within loop constructs
- While it is similar to a field, it is different because:
  - It is declared inside the method body, not at the beginning of a class
  - It cannot have a visibility modifier (public or private) in front of its definition
  - It exists only until the current method finishes running, and is then erased from memory

A local variable is a variable declared inside the body of the method to temporarily store values, such as references to objects or integers.

ORACLE Academy

Where as an instance variable has visibility within the whole class (can be accessed), a local variable cannot be seen outside out of the brackets it was declared in. This is normally referred to as scope.



Declare a Local Variable

- To create a while loop, first declare the local variable and assign it a value
- To declare a local variable:
  - Declare the variable type (integer or object reference)
  - Name the variable
  - Initialize the variable to a number (usually zero)
- Example:

Variable Name (i)

Variable Value (0)

```
int i = 0;
```

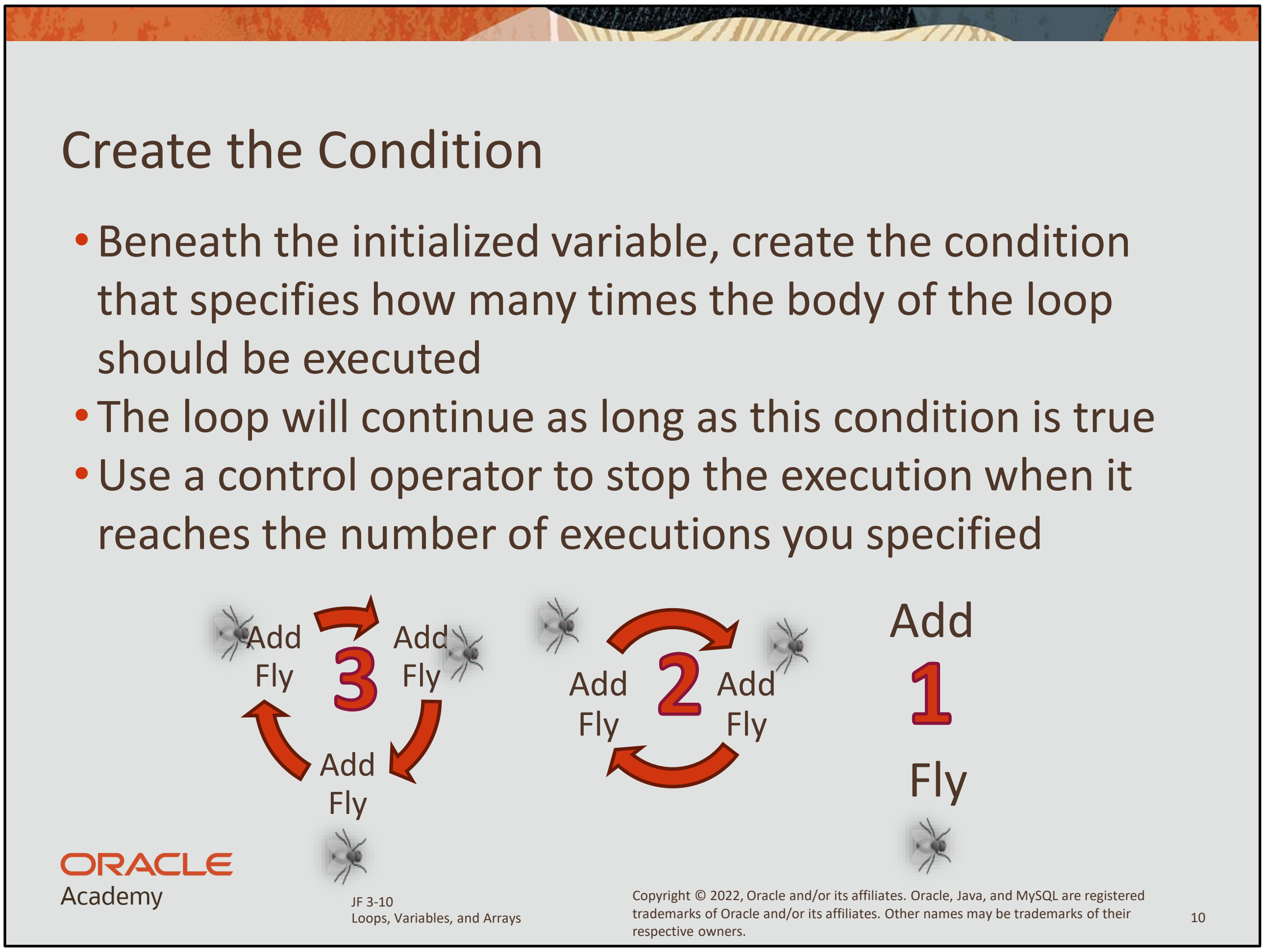
Variable Type (Integer)

ORACLE Academy

JF 3-10
Loops, Variables, and Arrays

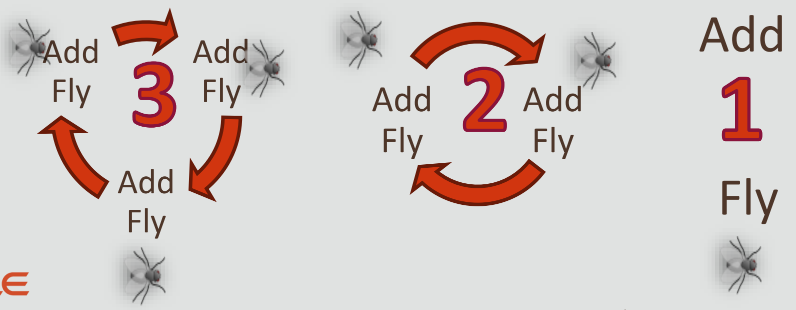
9

The letter "i" is often used as the name for the local variable. If you have a loop inside another loop then the letter "j", then "k" is often used. This letter is thought to be used in programming because of its origins in mathematics where it designated integers. This was then later picked up in the language Fortran.



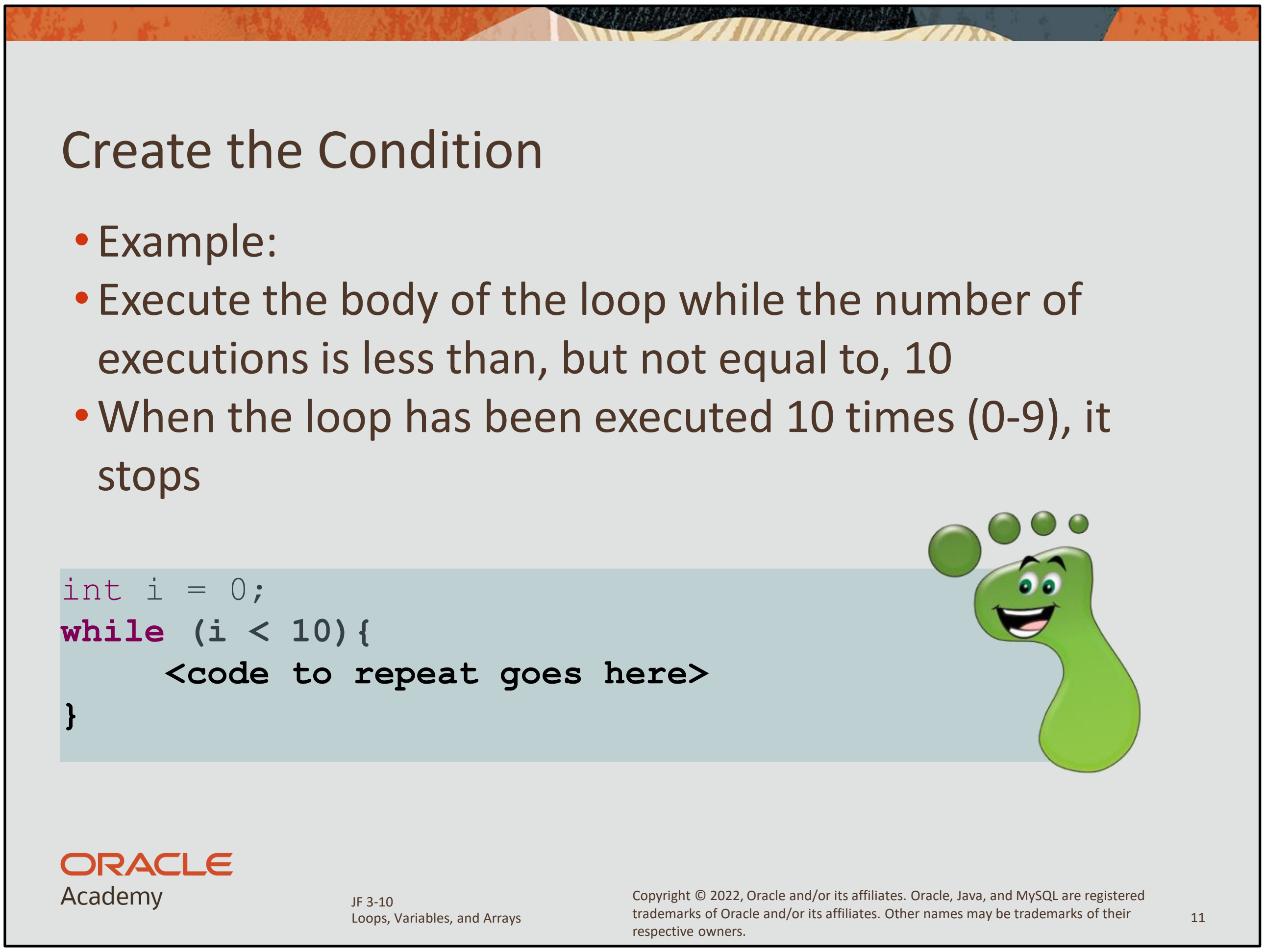
Create the Condition

- Beneath the initialized variable, create the condition that specifies how many times the body of the loop should be executed
- The loop will continue as long as this condition is true
- Use a control operator to stop the execution when it reaches the number of executions you specified



ORACLE Academy

We will see an example in the following slide.



Create the Condition

- Example:
- Execute the body of the loop while the number of executions is less than, but not equal to, 10
- When the loop has been executed 10 times (0-9), it stops

```
int i = 0;
while (i < 10){
     <code to repeat goes here>
}
```

ORACLE Academy

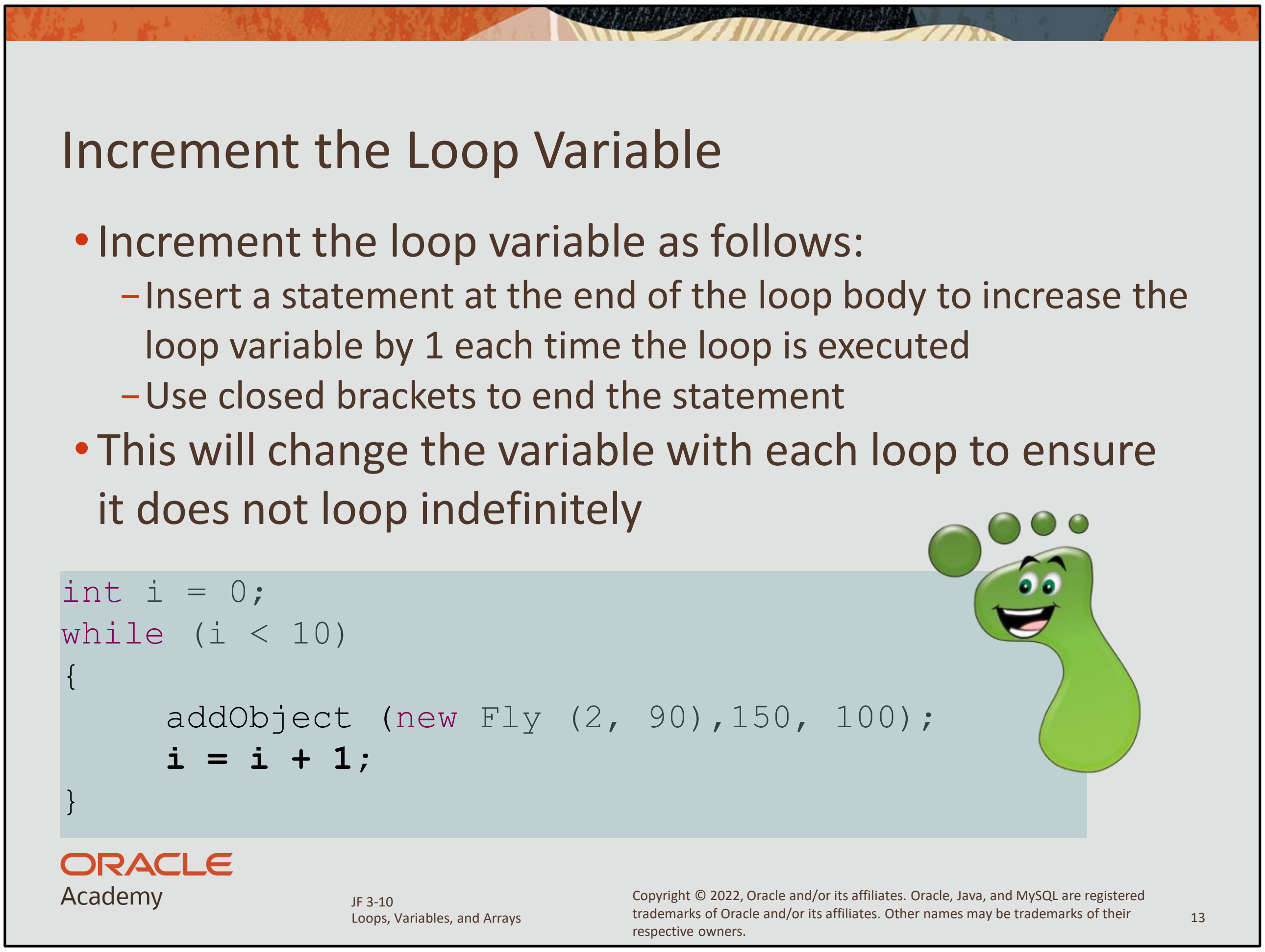
JF 3-10
Loops, Variables, and Arrays

For anyone who has done programming before you may notice a problem. We will visit this in a few slides.

Insert the Statements to Execute

- In brackets, insert the statements to execute
- Example:
  - Add 10 new Fly objects with a specific speed and direction

```
int i = 0;
while (i < 10)
{
     addObject (new Fly (2, 90),150, 100);
}
```



Increment the Loop Variable

- Increment the loop variable as follows:
  - Insert a statement at the end of the loop body to increase the loop variable by 1 each time the loop is executed
  - Use closed brackets to end the statement
- This will change the variable with each loop to ensure it does not loop indefinitely

```
int i = 0;
while (i < 10)
{
     addObject (new Fly (2, 90),150, 100);
     i = i + 1;
}
```

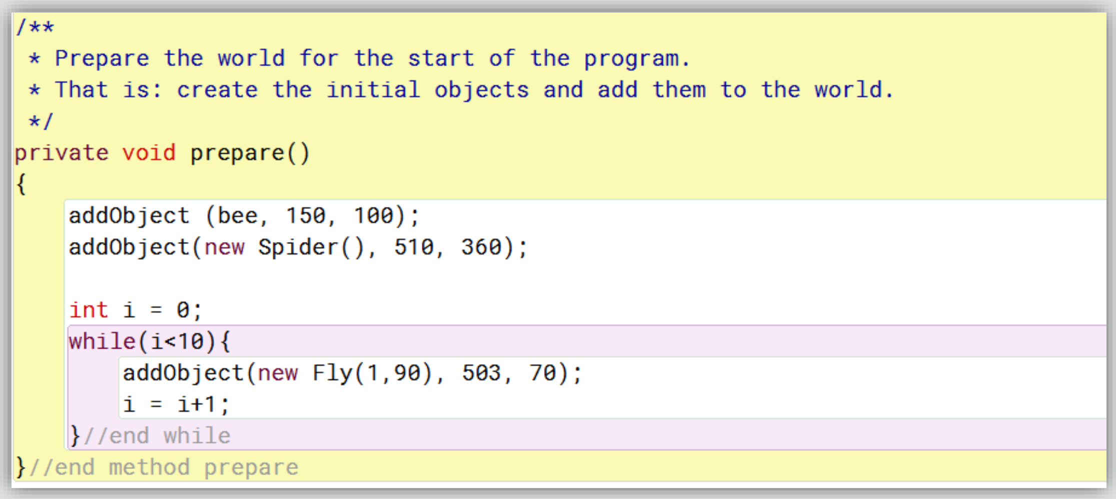
ORACLE Academy

JF 3-10
Loops, Variables, and Arrays

Remember i = i + 1 would normally be written as i++;
If we forget to add this line then the code will never stop looping. This is referred to as an infinite loop.

while Loop Example

- When inserted into the BeeWorld constructor it creates 10 Flies when the world is initialized



```
/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
{
    addObject (bee, 150, 100);
    addObject(new Spider(), 510, 360);

    int i = 0;
    while(i<10){
        addObject(new Fly(1,90), 503, 70);
        i = i+1;
    }//end while
}//end method prepare
```

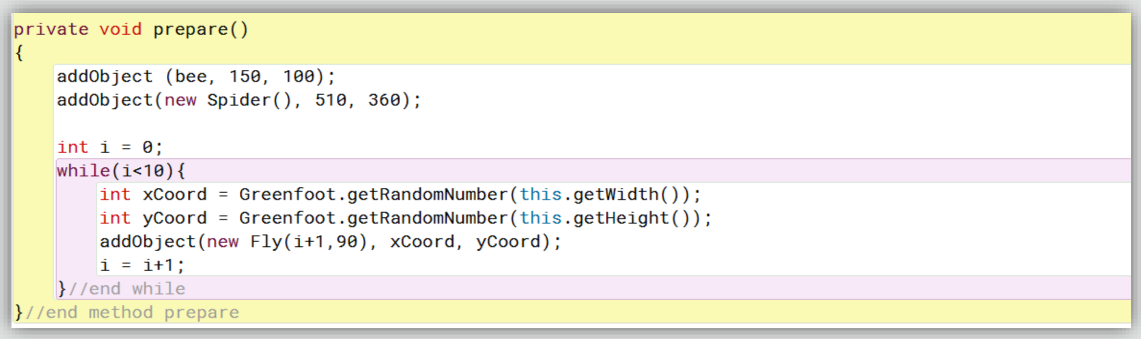
As this is in the constructor it will only ever be called once.

Object Placement and while Loops

- In the previous example, when the constructor is executed, all of the instances are placed at the same coordinates
- This causes them to sit on top of each other
- Create an expression that calculates the size of an instance, and then places subsequent instances at different random coordinates
- The max speed of the flies will be increased within the loop

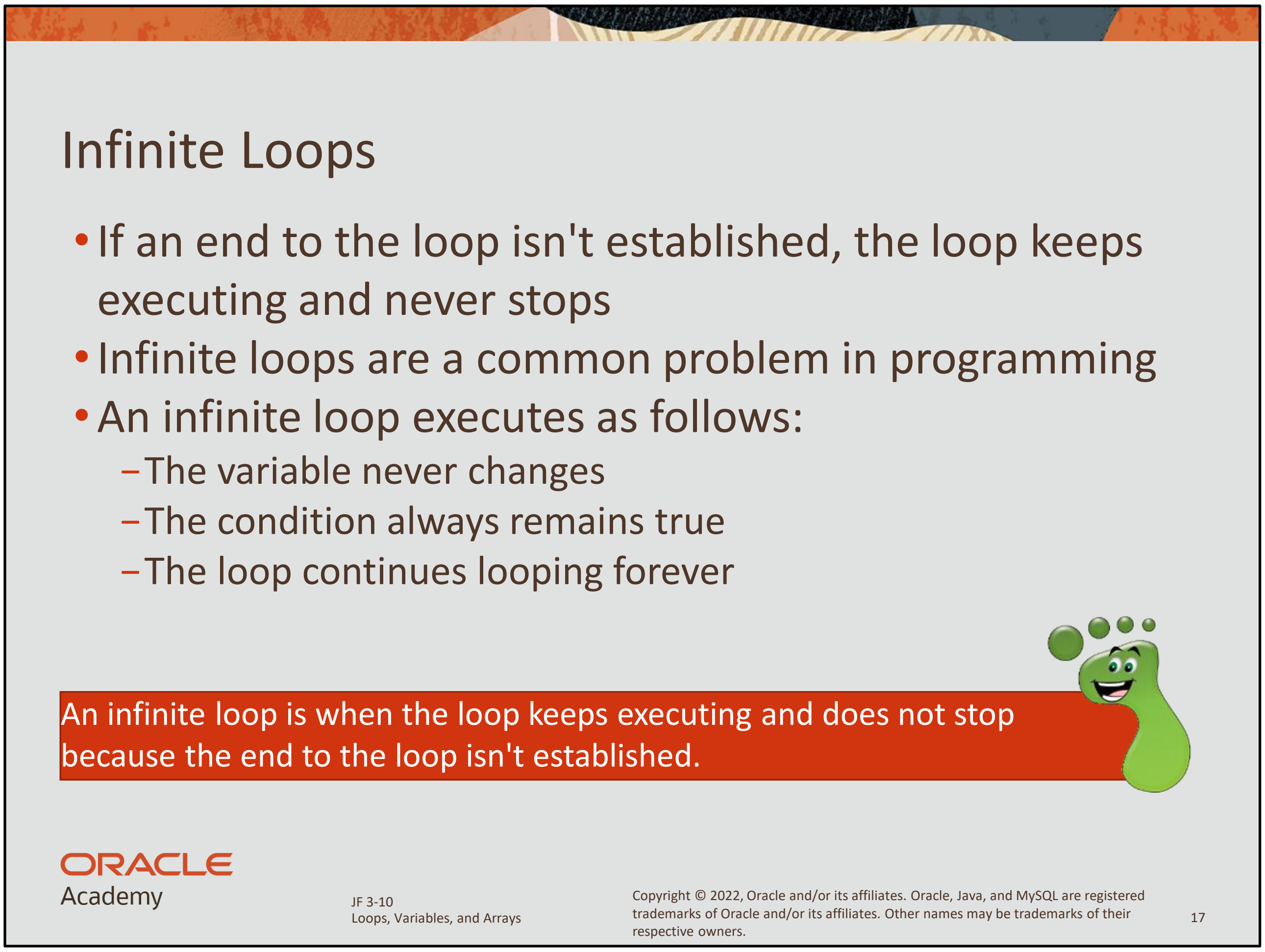
Calculate the Placement of Instances

- To program instances to land at different coordinates, replace the fixed x-coordinate for the object's width with an expression that includes:
  - Variable i
  - Random x and y coordinates



```
private void prepare()
{
    addObject (bee, 150, 100);
    addObject(new Spider(), 510, 360);

    int i = 0;
    while(i<10){
        int xCoord = Greenfoot.getRandomNumber(this.getWidth());
        int yCoord = Greenfoot.getRandomNumber(this.getHeight());
        addObject(new Fly(i+1,90), xCoord, yCoord);
        i = i+1;
    }//end while
}//end method prepare
```

The getWidth() and getHeight() methods return the current width and height of the world. So if we changed the world dimensions then we would be able to access them without changing our code.



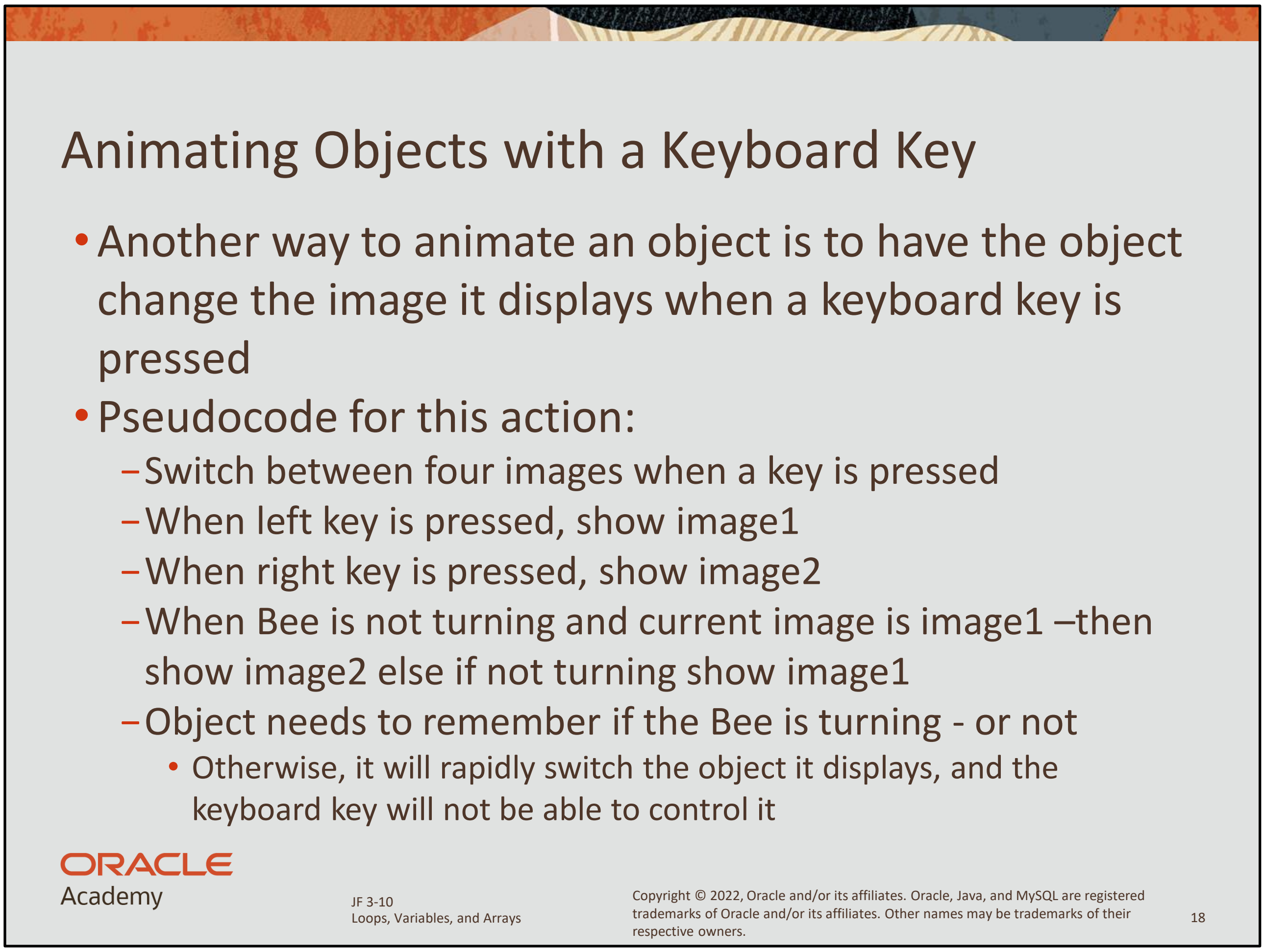
Infinite Loops

- If an end to the loop isn't established, the loop keeps executing and never stops
- Infinite loops are a common problem in programming
- An infinite loop executes as follows:
  - The variable never changes
  - The condition always remains true
  - The loop continues looping forever

An infinite loop is when the loop keeps executing and does not stop because the end to the loop isn't established.

ORACLE Academy

You should always double check your logic and code to make sure that the loop will only run the numbers of times that you would expect it to.

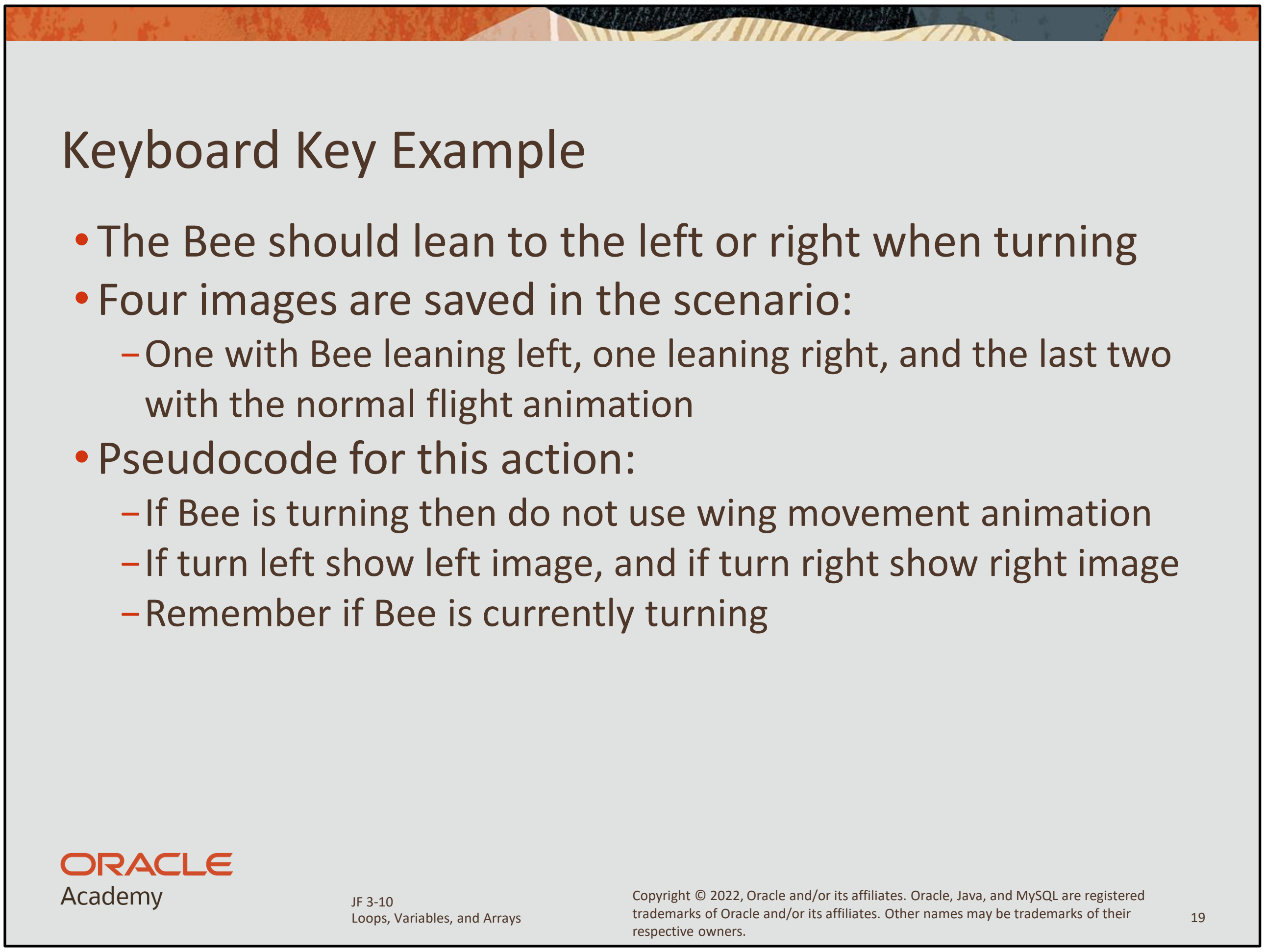


Animating Objects with a Keyboard Key

- Another way to animate an object is to have the object change the image it displays when a keyboard key is pressed
- Pseudocode for this action:
  - Switch between four images when a key is pressed
  - When left key is pressed, show image1
  - When right key is pressed, show image2
  - When Bee is not turning and current image is image1 –then show image2 else if not turning show image1
  - Object needs to remember if the Bee is turning - or not
    - Otherwise, it will rapidly switch the object it displays, and the keyboard key will not be able to control it

ORACLE Academy

Basically we should switch between 1 of 2 images unless we are turning. If we are turning then display either the left or right turning image.



Keyboard Key Example

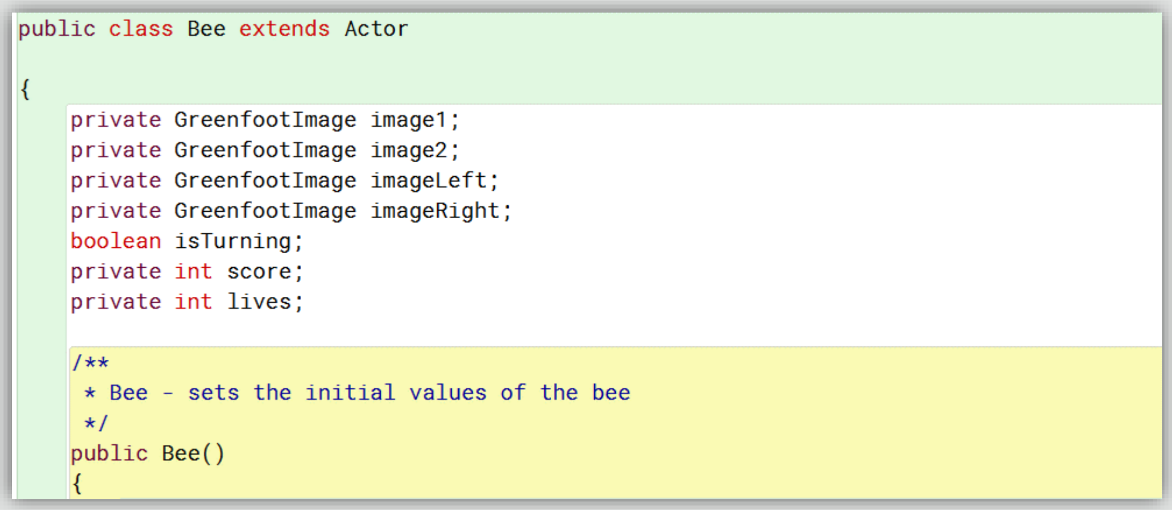
- The Bee should lean to the left or right when turning
- Four images are saved in the scenario:
  - One with Bee leaning left, one leaning right, and the last two with the normal flight animation
- Pseudocode for this action:
  - If Bee is turning then do not use wing movement animation
  - If turn left show left image, and if turn right show right image
  - Remember if Bee is currently turning

ORACLE
Academy

When storing the state of a current action we often use a class variable.

Declare Class Variables

- First, add new fields in the Bee class to store the left and right turns, and a field to store the current turn

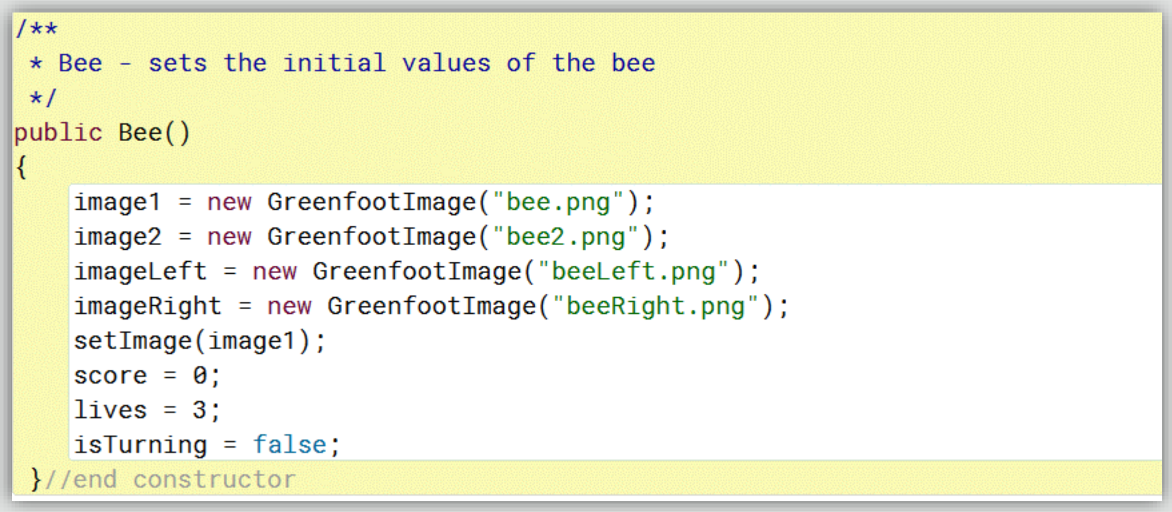


```
public class Bee extends Actor

{
    private GreenfootImage image1;
    private GreenfootImage image2;
    private GreenfootImage imageLeft;
    private GreenfootImage imageRight;
    boolean isTurning;
    private int score;
    private int lives;

    /**
     * Bee - sets the initial values of the bee
     */
    public Bee()
    {
```

It is good practice to use good class variable names, especially when setting boolean values. We could have called the variable notTurning, but it takes more thought to think what notTurning actually means rather than isTurning.

Initialize Variables

- Then initialize our class variables in the constructor



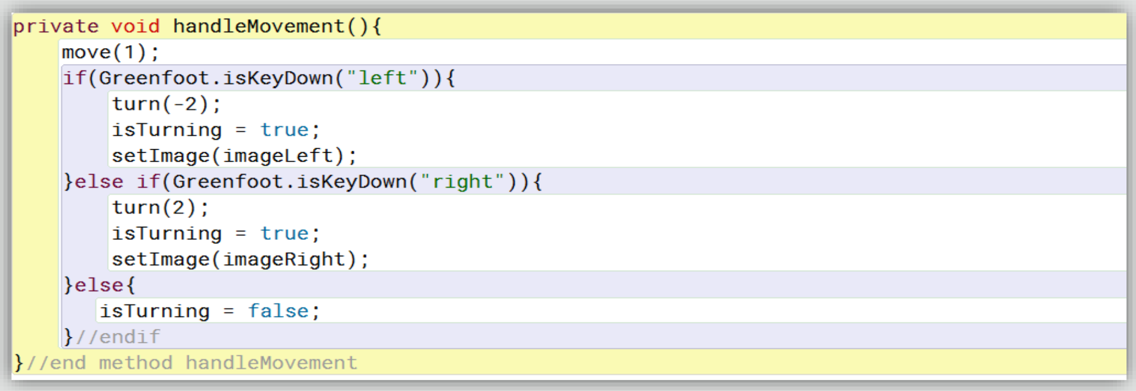
```
/**
 * Bee - sets the initial values of the bee
 */
public Bee()
{
    image1 = new GreenfootImage("bee.png");
    image2 = new GreenfootImage("bee2.png");
    imageLeft = new GreenfootImage("beeLeft.png");
    imageRight = new GreenfootImage("beeRight.png");
    setImage(image1);
    score = 0;
    lives = 3;
    isTurning = false;
 }//end constructor
```

ORACLE Academy

Remember that Java is case sensitive so "BeeLeft.png" is not the same as "beeLeft.png". Check that you are using the correct case for your sound and image files.

Set isTurning Variable

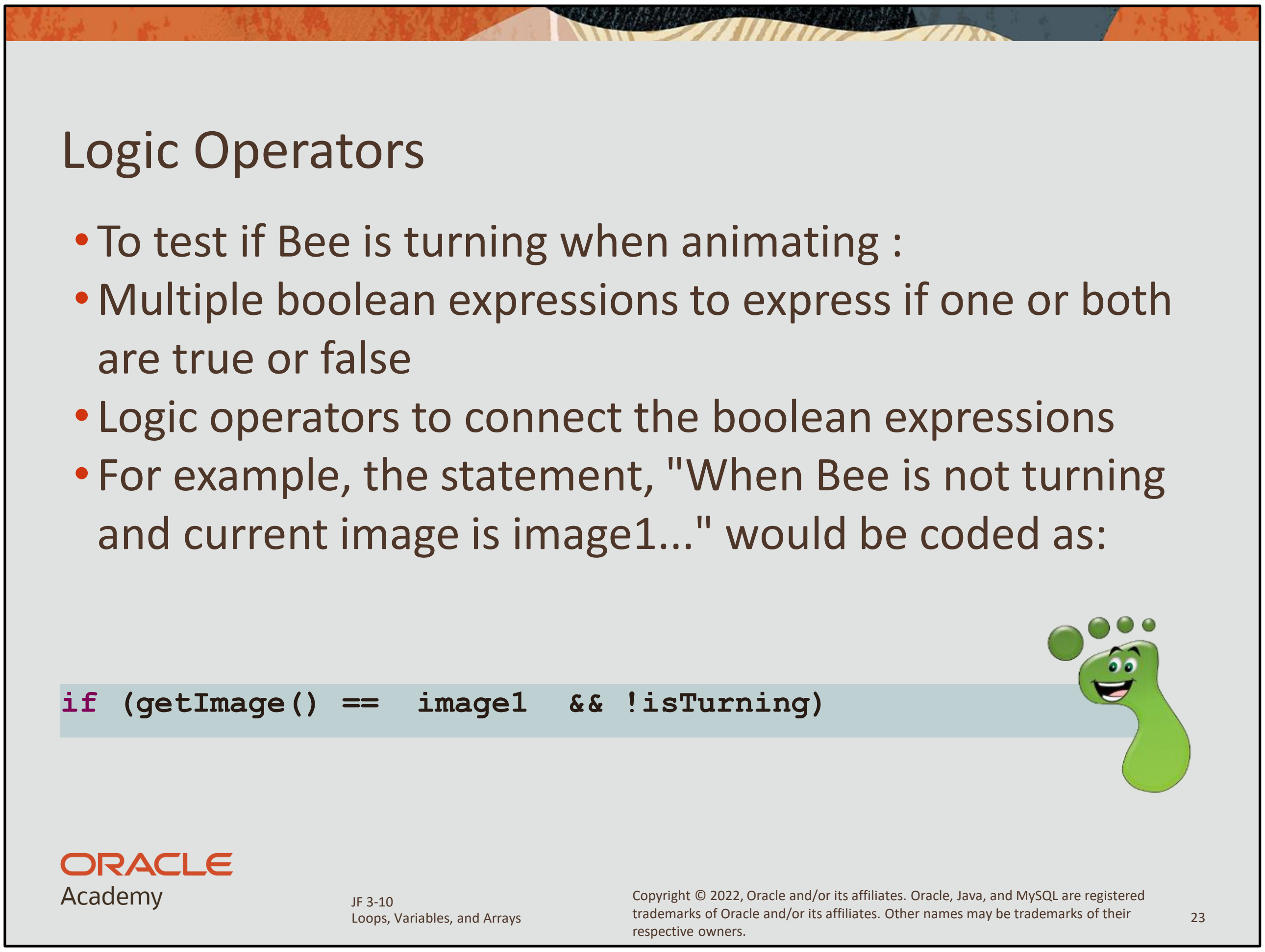
- Next, in our handleMovement() method we wrote earlier we will set the state of our isTurning variable to true if we are turning, else we will set it to false
- We will also set the appropriate turn image



```
private void handleMovement(){
    move(1);
    if(Greenfoot.isKeyDown("left")){
        turn(-2);
        isTurning = true;
        setImage(imageLeft);
    }else if(Greenfoot.isKeyDown("right")){
        turn(2);
        isTurning = true;
        setImage(imageRight);
    }else{
       isTurning = false;
    }//endif
}//end method handleMovement
```

ORACLE
Academy

At this point, the bee would still try to animate during the turn.



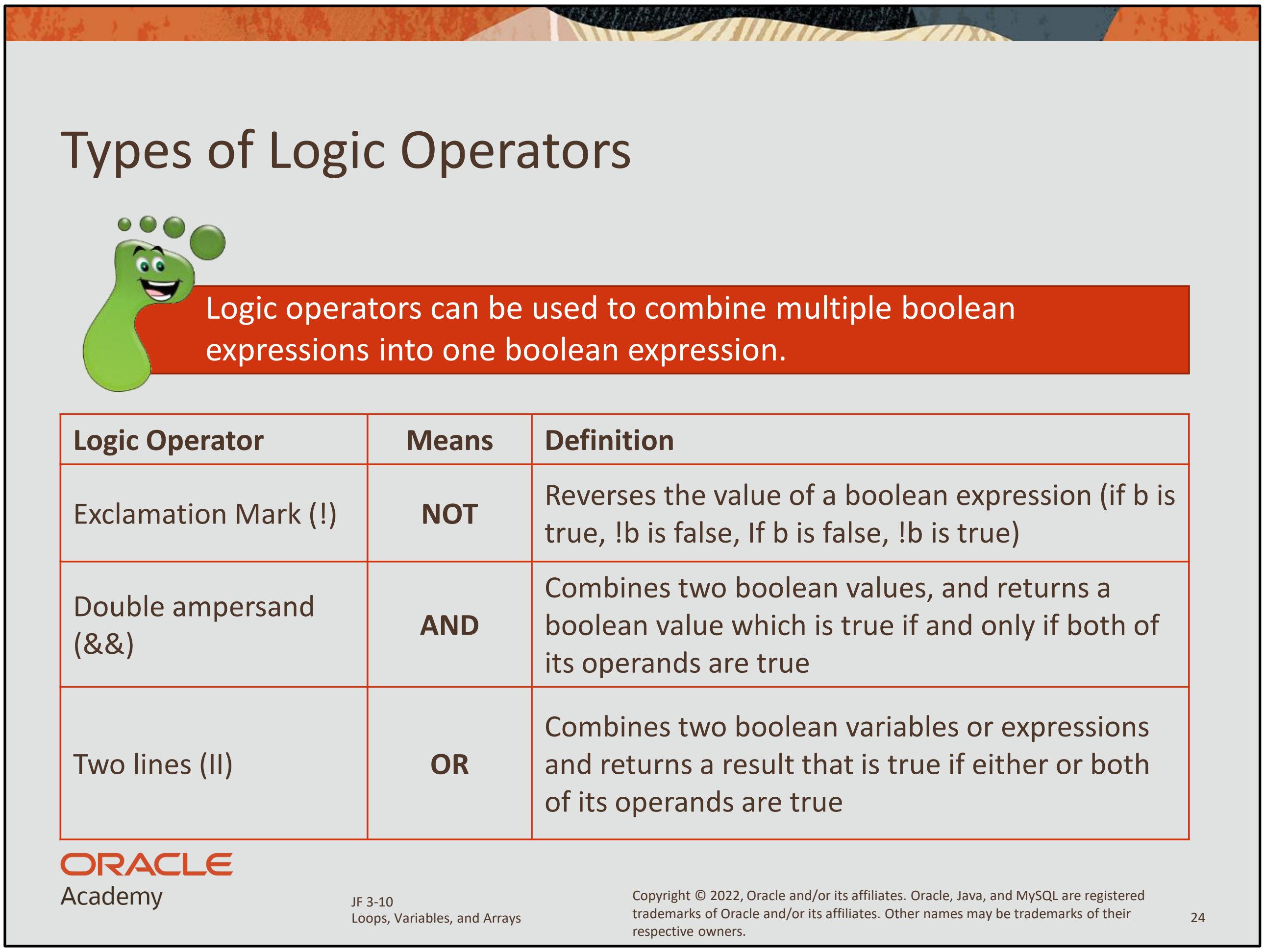
Logic Operators

- To test if Bee is turning when animating :
- Multiple boolean expressions to express if one or both are true or false
- Logic operators to connect the boolean expressions
- For example, the statement, "When Bee is not turning and current image is image1..." would be coded as:

```
if (getImage() ==  image1  && !isTurning)
```

ORACLE
Academy

&& (means "and") only returns true if the statements on both sides of the && are true.



Types of Logic Operators

Logic operators can be used to combine multiple boolean expressions into one boolean expression.

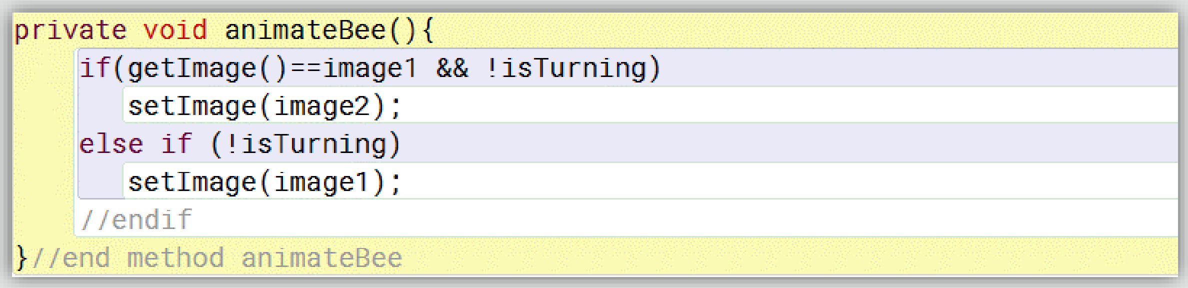
Logic Operator	Means	Definition
Exclamation Mark (!)	**NOT**	Reverses the value of a boolean expression (if b is true, !b is false, If b is false, !b is true)
Double ampersand (&&)	**AND**	Combines two boolean values, and returns a boolean value which is true if and only if both of its operands are true
Two lines (II)	**OR**	Combines two boolean variables or expressions and returns a result that is true if either or both of its operands are true

ORACLE
Academy

You can create truth tables for the AND /NOT/OR. To do this following the guide below.
For the AND/OR you can use 2 inputs of true and false and show the result.
SO
TRUE TRUE
TRUE FALSE
FALSE TRUE
FALSE FALSE
Are the 4 possible inputs. For the AND/OR show the resulting boolean value.

Logic Operators Example

- Logic operators set the image that appears if the Bee is turning or not



```
private void animateBee(){
    if(getImage()==image1 && !isTurning)
        setImage(image2);
    else if (!isTurning)
        setImage(image1);
    //endif
}//end method animateBee
```

Remember that && represents "and" and will only return true if both sides are true.

Arrays

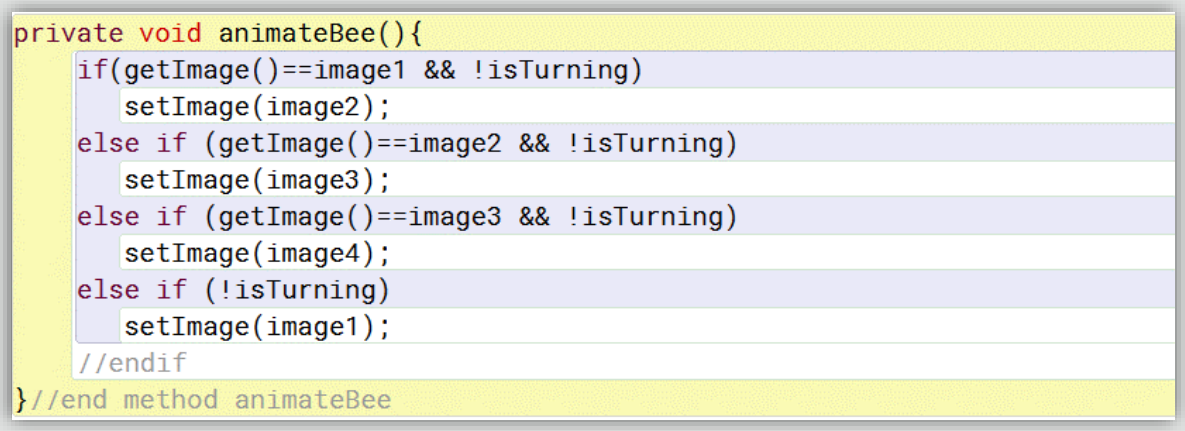
- When we created our basic animation we used two images to simulate wings flapping

- Most animations will have multiple images, but this would increase the number of conditions to test in our code

The more images, the smoother the animation.

Arrays

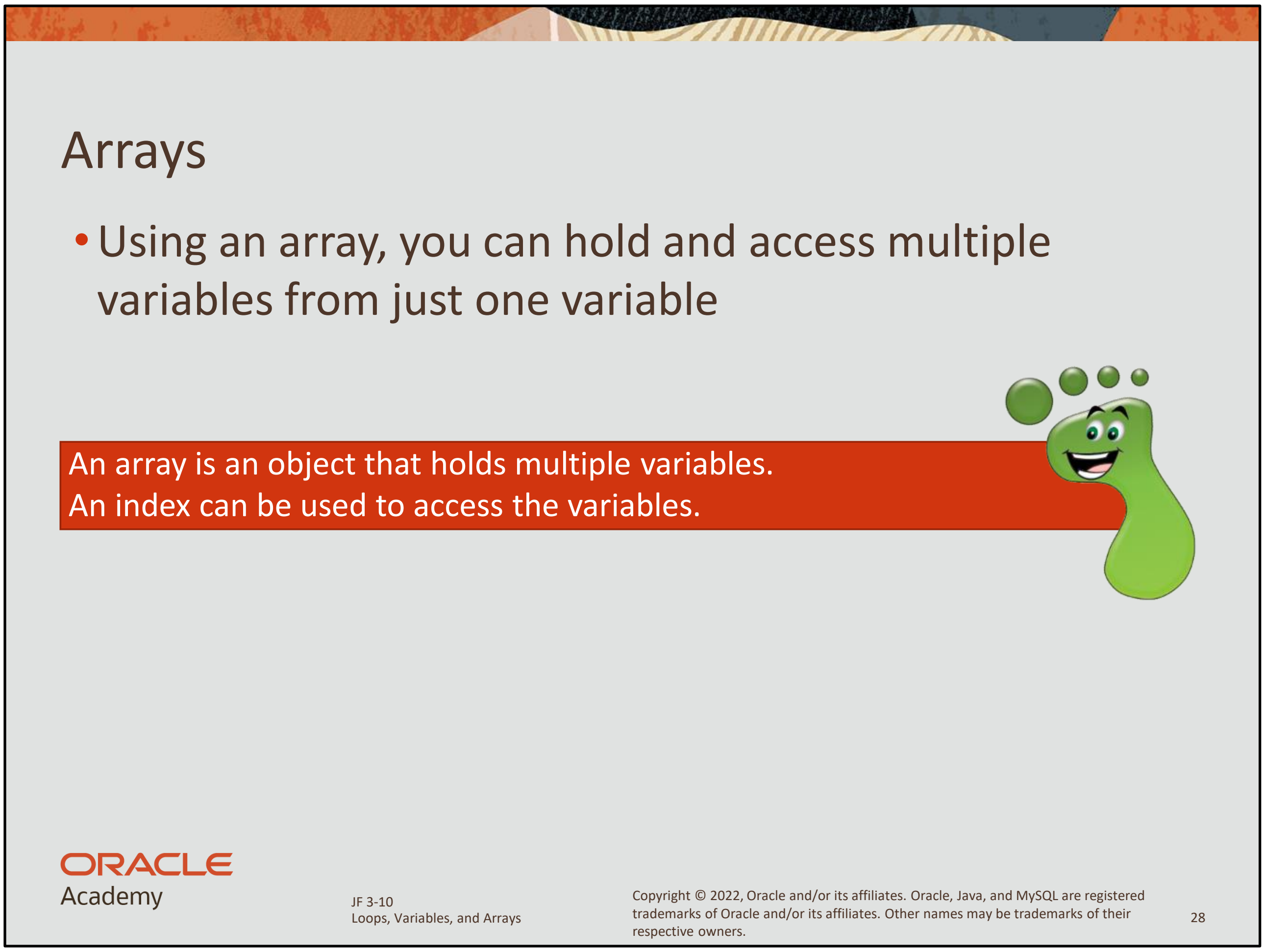
- With only 4 images our code would look like this:



```
private void animateBee(){
    if(getImage()==image1 && !isTurning)
       setImage(image2);
    else if (getImage()==image2 && !isTurning)
       setImage(image3);
    else if (getImage()==image3 && !isTurning)
       setImage(image4);
    else if (!isTurning)
       setImage(image1);
    //endif
}//end method animateBee
```

- We could easily have 8 or more images!

Remember that in an if-else statement only one of the code sections can ever run.



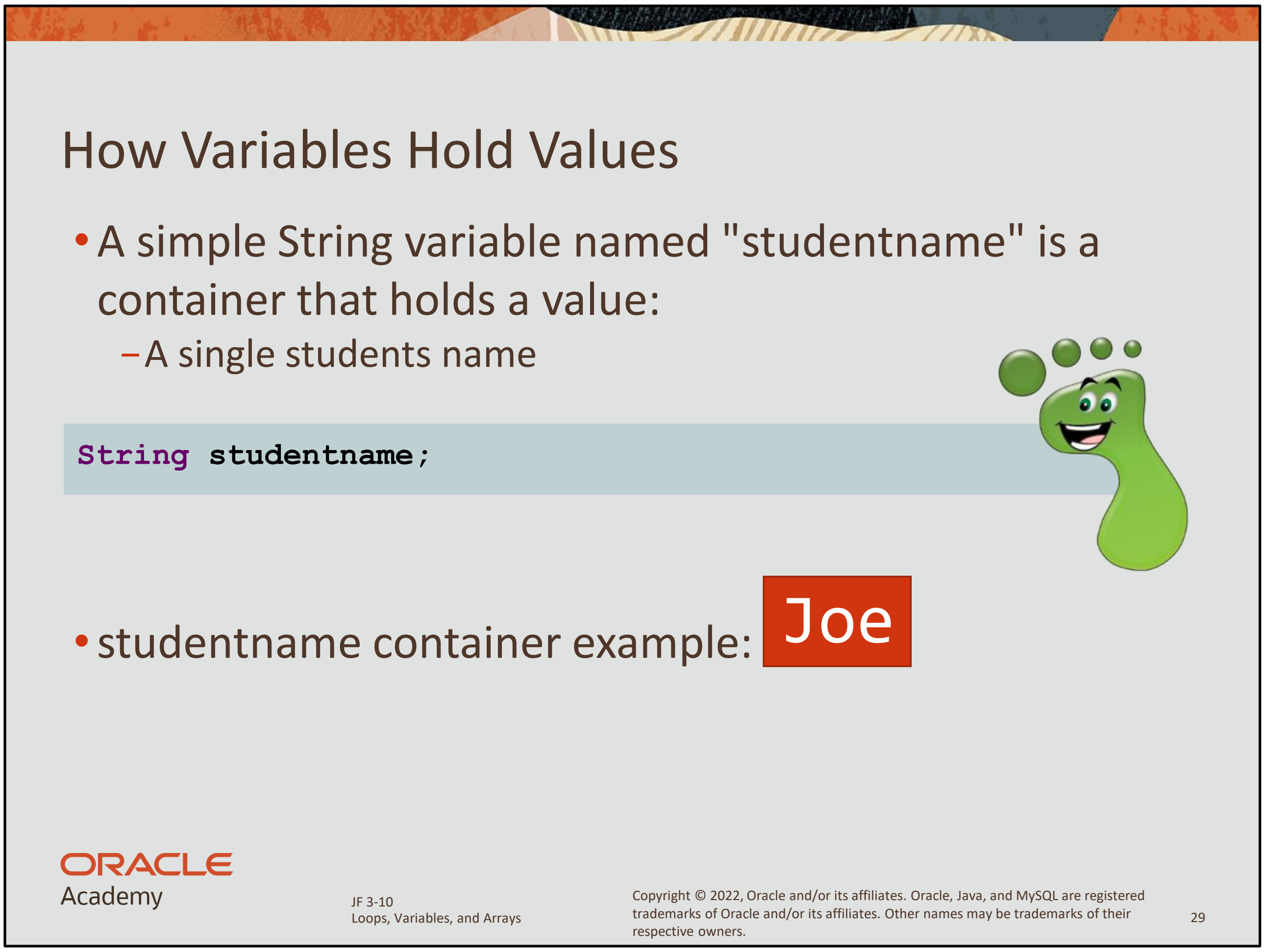
Arrays

- Using an array, you can hold and access multiple variables from just one variable

An array is an object that holds multiple variables.
An index can be used to access the variables.

ORACLE Academy

As you develop your skills and knowledge in java you will see that there are more advanced structures for storing multiple variables in a single variable.



How Variables Hold Values

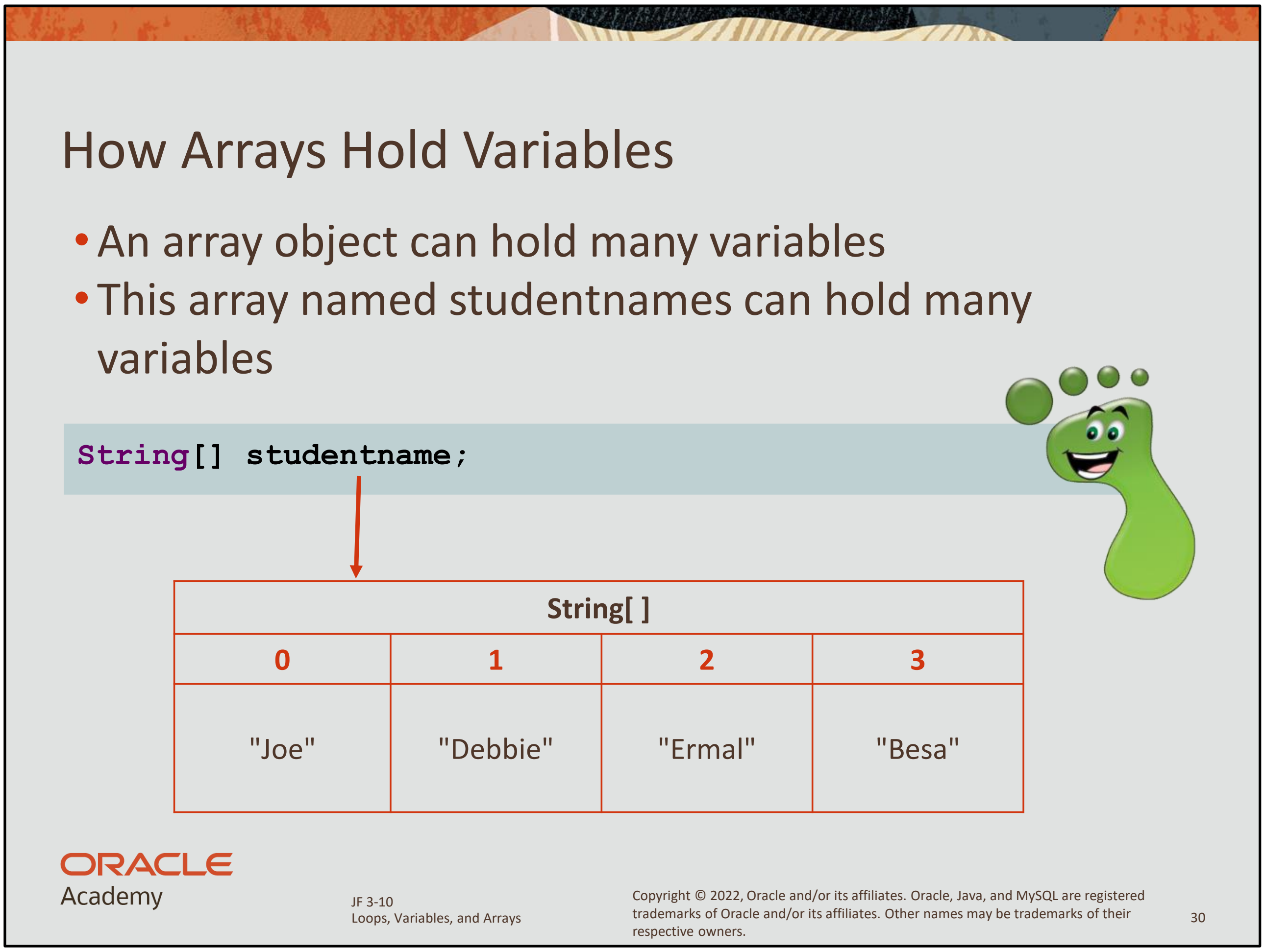
- A simple String variable named "studentname" is a container that holds a value:
  - A single students name

```
String studentname;
```

- studentname container example: Joe

ORACLE Academy

Strings hold any alphanumeric data. So we could also store "Joe!1"



How Arrays Hold Variables

- An array object can hold many variables
- This array named studentnames can hold many variables

```
String[] studentname;
```

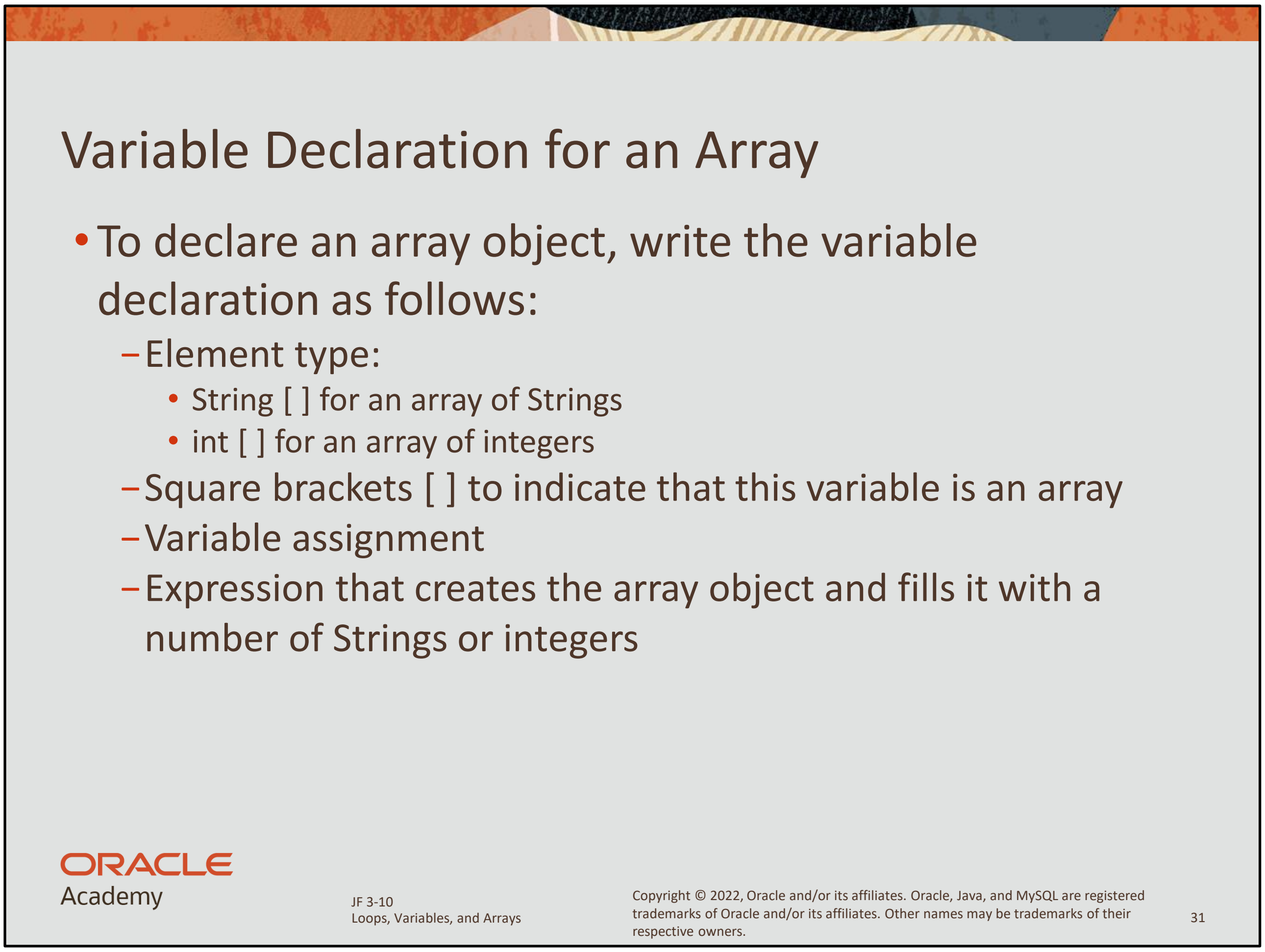
String[ ]			
0	1	2	3
"Joe"	"Debbie"	"Ermal"	"Besa"

Academy

JF 3-10
Loops, Variables, and Arrays

30

To define an array we use the square brackets. The type before the square brackets is the type that each cell in the array can store. In the case above String[] sets up an array that stores Strings.

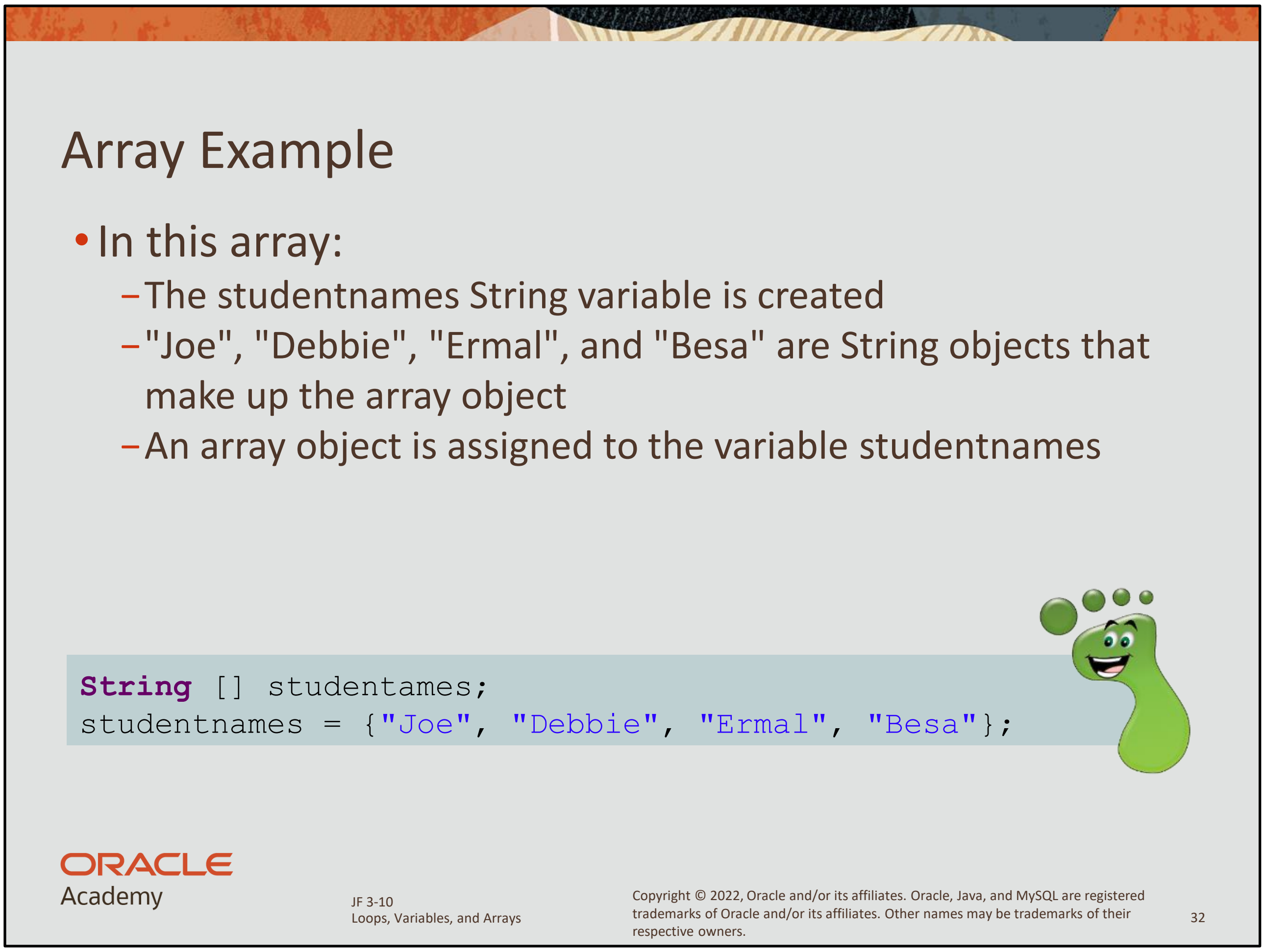


Variable Declaration for an Array

- To declare an array object, write the variable declaration as follows:
  - Element type:
    - String [ ] for an array of Strings
    - int [ ] for an array of integers
  - Square brackets [ ] to indicate that this variable is an array
  - Variable assignment
  - Expression that creates the array object and fills it with a number of Strings or integers

ORACLE
Academy

Before we use an array we will have to setup the maximum number of cells that will be available.



Array Example

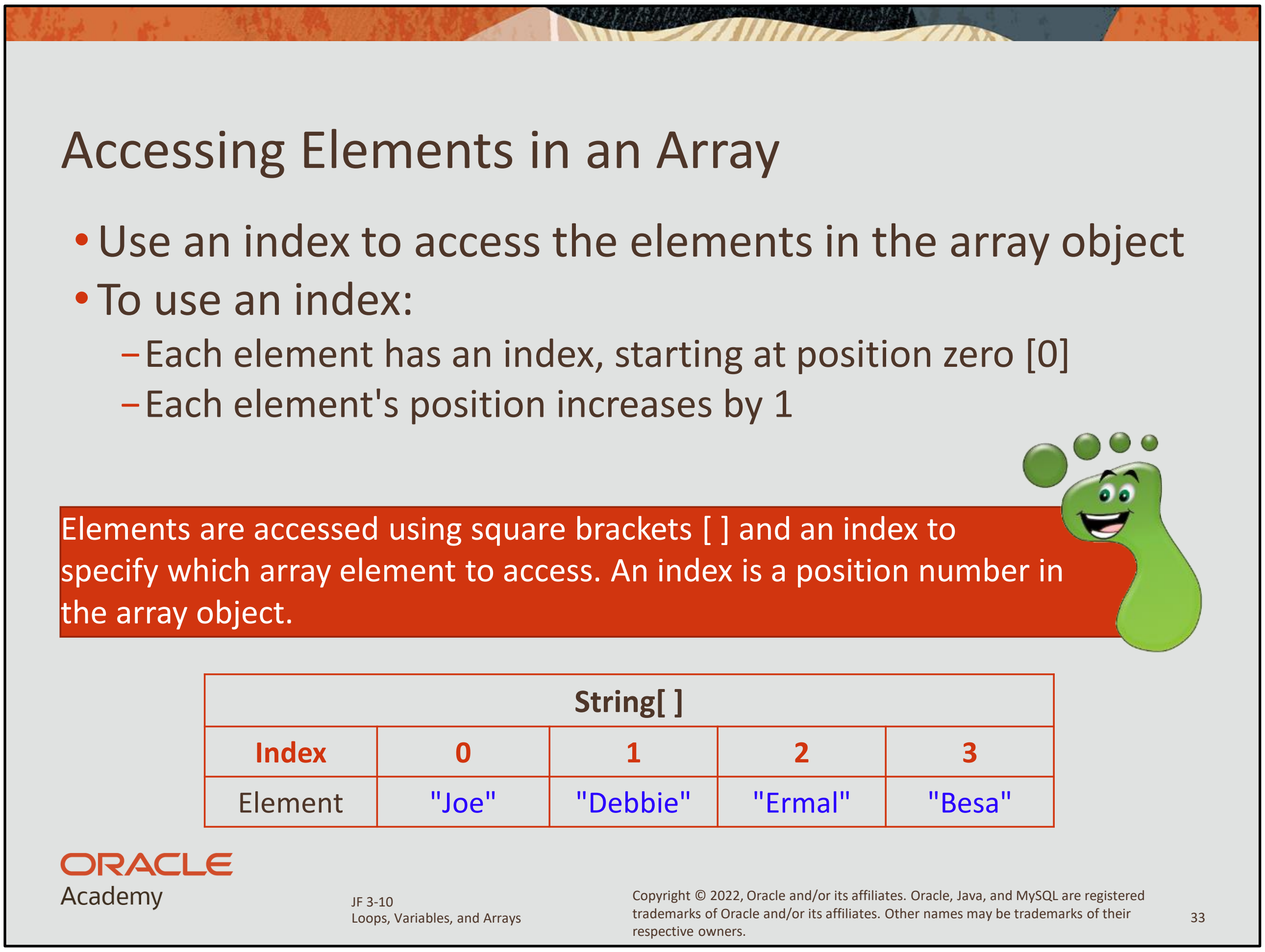
- In this array:
  - The studentnames String variable is created
  - "Joe", "Debbie", "Ermal", and "Besa" are String objects that make up the array object
  - An array object is assigned to the variable studentnames

```
String [] studentnames;
studentnames = {"Joe", "Debbie", "Ermal", "Besa"};
```

This is one example of initializing an array variable. We could have written

```
String[] studentnames = new String[4];
studentnames[0] = "Joe";
studentnames[1] = "Debbie";
studentnames[2] = "Ermal";
studentnames[3] = "Besa";
```

Notice that the first element of an array starts at the index 0.



Accessing Elements in an Array

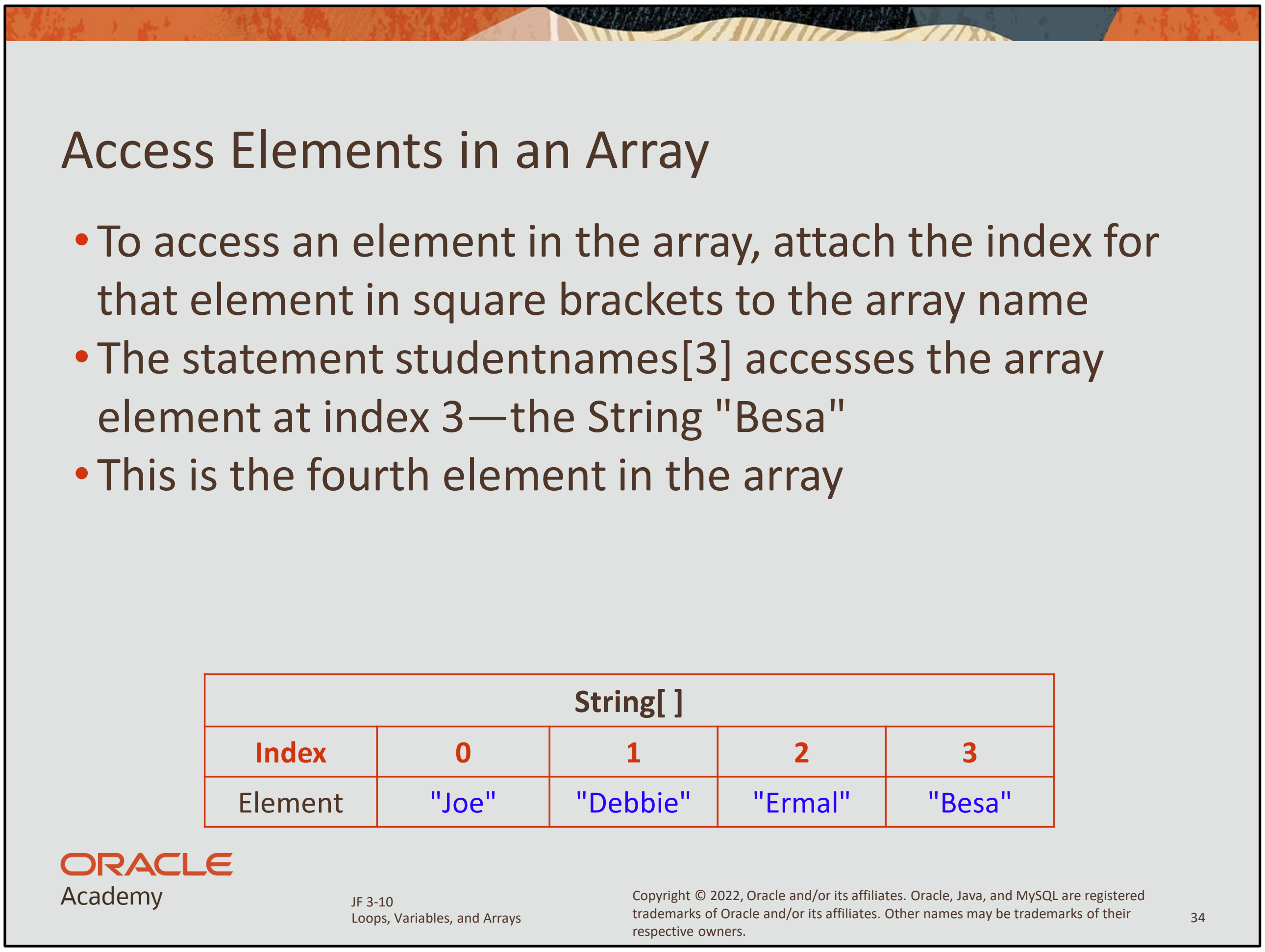
- Use an index to access the elements in the array object
- To use an index:
  - Each element has an index, starting at position zero [0]
  - Each element's position increases by 1

Elements are accessed using square brackets [ ] and an index to specify which array element to access. An index is a position number in the array object.

String[ ]				
Index	**0**	**1**	**2**	**3**
Element	"Joe"	"Debbie"	"Ermal"	"Besa"

ORACLE
Academy

You will generate an error if you try to access an array element out of the range. In this example the element range is from 0 to 3.



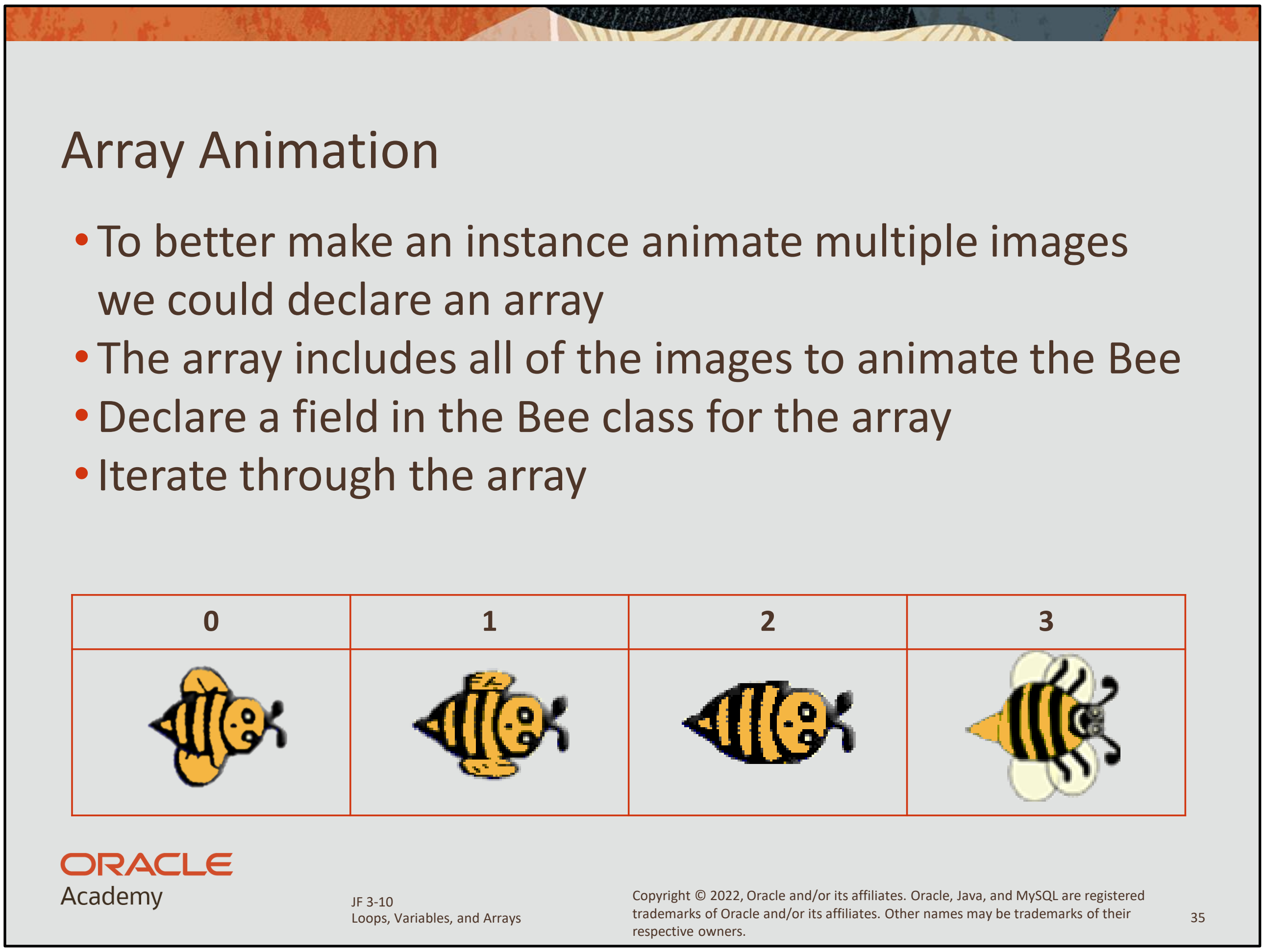
Access Elements in an Array

- To access an element in the array, attach the index for that element in square brackets to the array name
- The statement studentnames[3] accesses the array element at index 3—the String "Besa"
- This is the fourth element in the array

String[ ]				
Index	**0**	**1**	**2**	**3**
Element	"Joe"	"Debbie"	"Ermal"	"Besa"

ORACLE
Academy

Remember that this array is declared as a String array. If you return one element from the array, this value will be a String



Array Animation

- To better make an instance animate multiple images we could declare an array
- The array includes all of the images to animate the Bee
- Declare a field in the Bee class for the array
- Iterate through the array

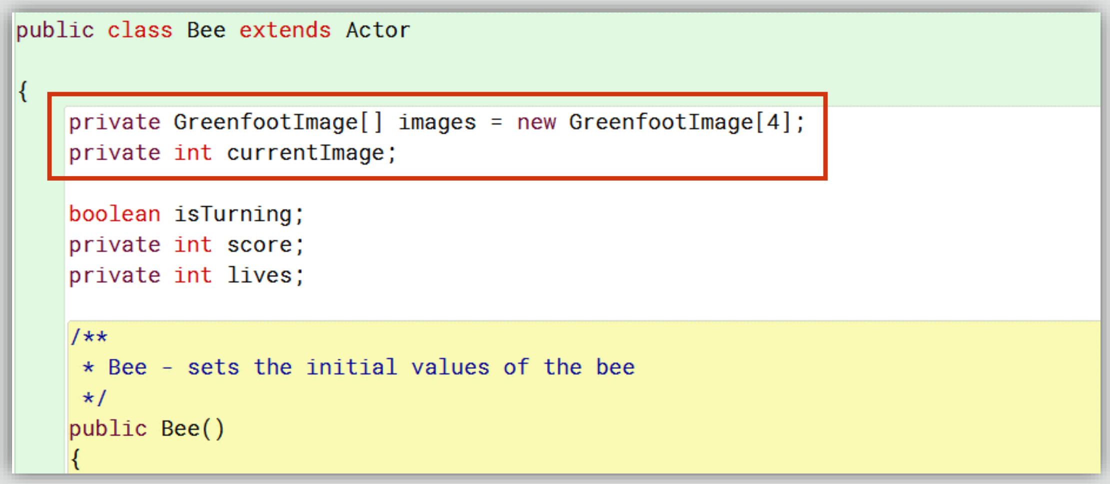
0	1	2	3

ORACLE Academy

This will mean that we can store all of our images in one variable.

Create the Arrays

- Create an array in the Bee class to store 4 images
- Create an integer variable to store current image index



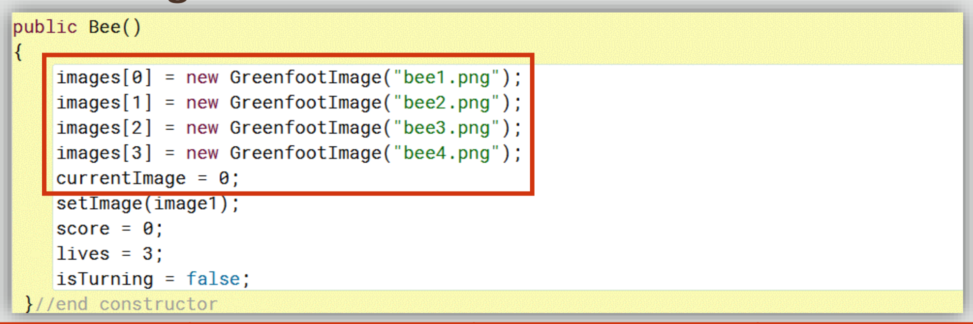
```
public class Bee extends Actor

{
    private GreenfootImage[] images = new GreenfootImage[4];
    private int currentImage;

    boolean isTurning;
    private int score;
    private int lives;

    /**
     * Bee - sets the initial values of the bee
     */
    public Bee()
    {
```

ORACLE Academy

We could declare a constant for the max images rather than use the value 4. This would make the code more readable and be far easier to change in the future.
If we had the constant MAXIMAGES = 4;
We could change the code to
….. = new GreenfootImage[MAXIMAGES];

Create the Arrays

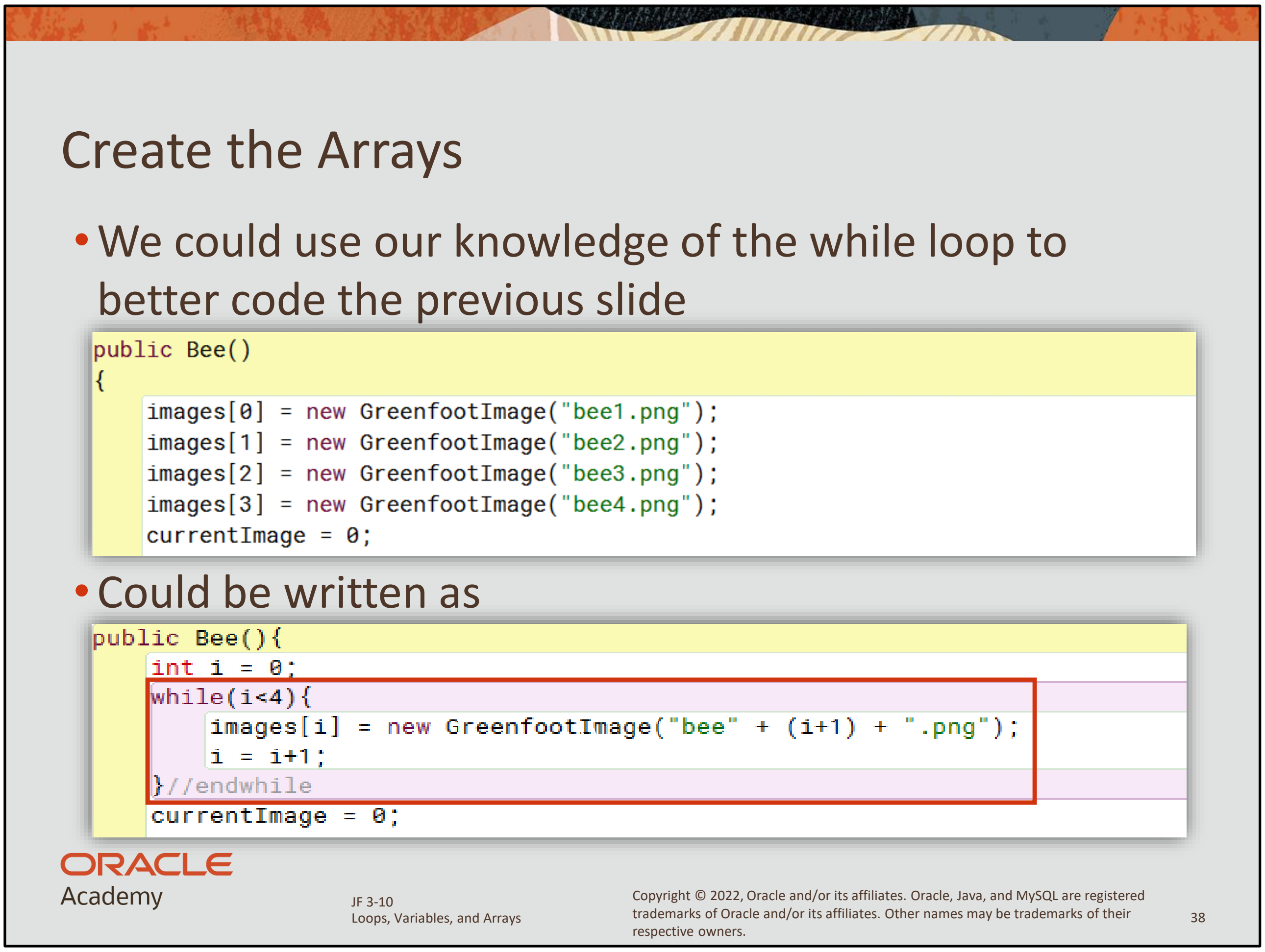
- Initialize the new image, the images array and the current image variables in the constructor



```
public Bee()
{
    images[0] = new GreenfootImage("bee1.png");
    images[1] = new GreenfootImage("bee2.png");
    images[2] = new GreenfootImage("bee3.png");
    images[3] = new GreenfootImage("bee4.png");
    currentImage = 0;
    setImage(image1);
    score = 0;
    lives = 3;
    isTurning = false;
 }//end constructor
```

0	1	2	3

Image 2 is used twice as we want to show images – bee -> bee1 -> bee2 -> bee1 then repeat.



Create the Arrays

- We could use our knowledge of the while loop to better code the previous slide

```
public Bee()
{
    images[0] = new GreenfootImage("bee1.png");
    images[1] = new GreenfootImage("bee2.png");
    images[2] = new GreenfootImage("bee3.png");
    images[3] = new GreenfootImage("bee4.png");
    currentImage = 0;
```

- Could be written as

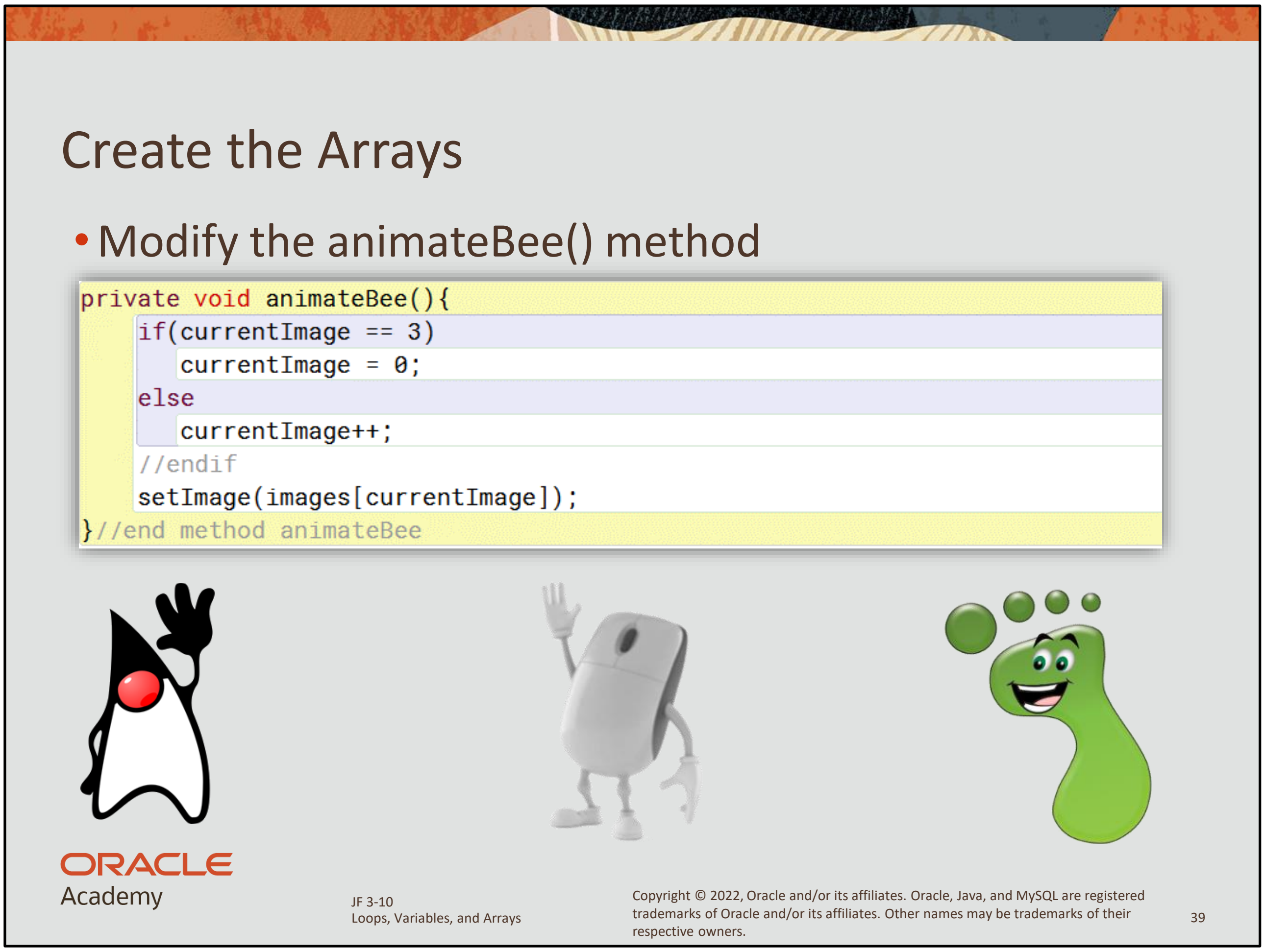
```
public Bee(){
    int i = 0;
    while(i<4){
        images[i] = new GreenfootImage("bee" + (i+1) + ".png");
        i = i+1;
    }//endwhile
    currentImage = 0;
```

ORACLE Academy

JF 3-10
Loops, Variables, and Arrays

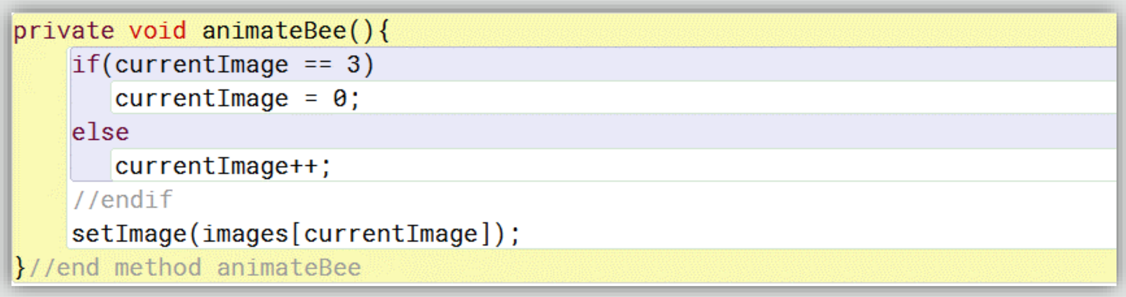
38

The second solution would be easier to add more images. The ("bee" + i + ".png"); is known as string concatenation. It joins the strings together. So if i=1, we would get "bee1.png". On the while loop example, we would copy the bee1.png and call it bee3.png.
currentimage would be a class field.

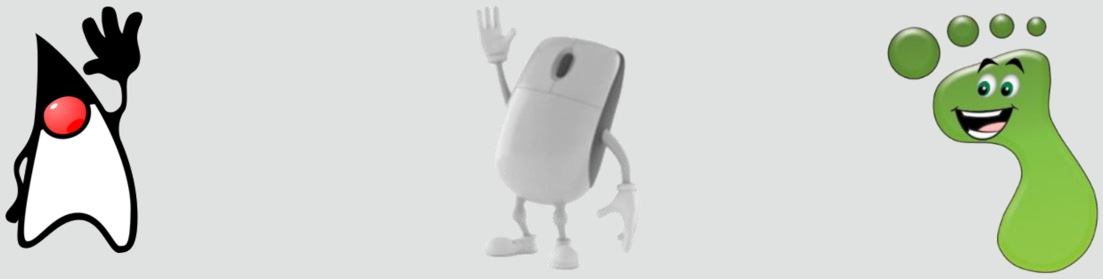


Create the Arrays

- Modify the animateBee() method



```
private void animateBee(){
    if(currentImage == 3)
        currentImage = 0;
    else
        currentImage++;
    //endif
    setImage(images[currentImage]);
}//end method animateBee
```



ORACLE Academy

Notice that the animateBee() method would now handle any amount of images. We would only require to change the index value 3 to a new value. In this case the 3 would be better substituted for a constant.
if (currentimage == MAX_IMAGES-1)…

MAX_IMAGES would be declared as a Class Field
private final int MAX_IMAGES = 4;

We would then update our code to use this constant rather than 3. This makes our code more readable and maintainable.

Terminology

- Key terms used in this lesson included:
  - Array
  - Elements
  - Index
  - Infinite loop
  - Local variables
  - Logic operators
  - Loop

Summary

- In this lesson, you should have learned how to:
  - Create a while loop in a constructor to build a world
  - Describe an infinite loop and how to prevent one from occurring
  - Use an array to store multiple variables
  - Create an expression using logic operators
  - Describe the scope of a local variable in a method

ORACLE
Academy