

The logo for Oracle Academy. The word "ORACLE" is in a bold, orange, sans-serif font. Below it, the word "Academy" is in a smaller, dark gray, sans-serif font. The entire logo is centered on a light gray background, which is framed by dark gray horizontal bars at the top and bottom.

# ORACLE

## Academy

# Java Fundamentals

2-13

## Java Variables and Data Types

**ORACLE**  
Academy



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

# Objectives

- This lesson covers the following objectives:
  - Describe variables
  - Describe Java simple types
  - Define arithmetic operators
  - Describe relational and logical operators
  - Describe assignment operators



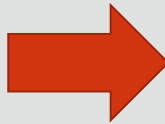
# Alice 3 Versus Java

Alice 3	Java
3D programming environment that uses visual representations for the Java programming language	Programming language; syntax can be edited using integrated development environment (IDE)
Used to create animations or interactive games while working with programming constructs	Used to create applications that run on any platform, including the web, using Java syntax
Drag and drop interface designed to reduce syntax errors and make learning programming easier	IDE helps you objects model real world objects, allow for re-use and easier maintenance

This table shows some of the features of creating programs in Alice 3 compared to how they are described in a pure Java environment.

## Variables in Alice 3

- Variables are declared (created) in your code
- A typical application uses various values which continuously change while the program is running
- For example, in Alice 3, a car is programmed to roll over a certain number of times
- The value entered by one user may be different from the value entered by another user



Variables are place holders for values that are stored in memory.

# Variables Manage Values

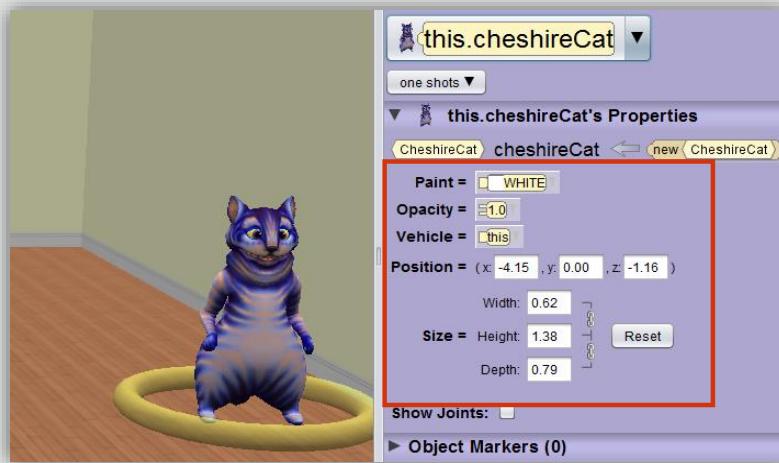
- Variables manage the values entered by different users in Alice 3

A variable is a place in memory where data of a specific type can be stored for later retrieval and use by your program. Each variable is given a unique name to make it easy to find.

When creating variables their name should be meaningful and unique. Using the same name for different variables can lead to confusion and errors within the code.

# Object Properties

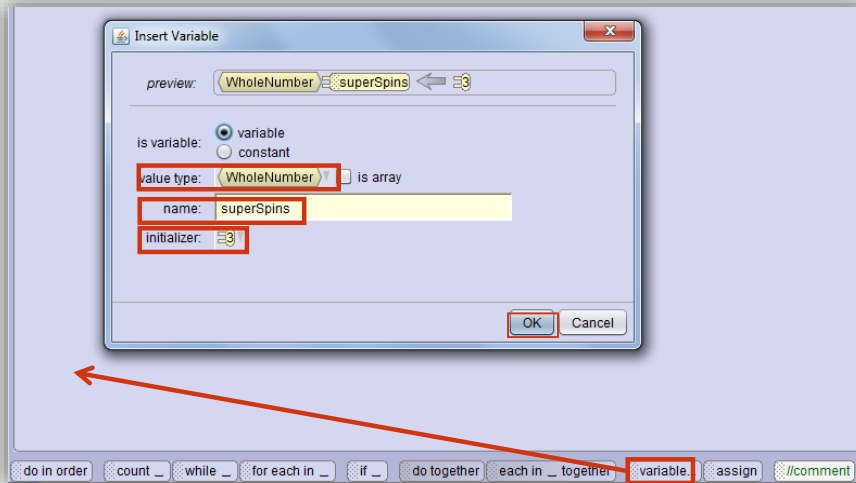
- Object properties are variables that store information about the object, such as its color, height, and depth



The properties of an object are thought of as variables because you can change the value of them in the code at any time.

## Declare Variables in Alice 3

- Drag the Variable tile into the Code editor to declare a new variable





# Declare Variables in Java

- In Java, you declare variables that can then be referenced in other parts of the program

```
public class Print{  
    public static void main(String[] args){  
        //variable declaration section  
        int i=1;  
        int j;  
  
        while(i<=7){  
            for(j=1; j<=i; j++)  
                System.out.println("*");  
            i=i+2;  
            System.out.println();  
        }//end while  
    }//end method main  
}//end class Print
```

i and j are variables that are declared here

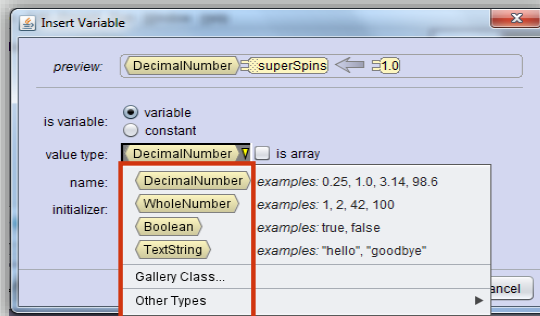
i and j are called in the program body

This is how you declare variables in Java outside of the Alice 3 environment. You will create variables like this when you use Greenfoot and Eclipse later in the course.

## Data Types in Alice 3

- When you declare a variable in Alice, you must define the data type

A variable's data type defines the kind of values that a variable can store.



The data type is what defines the type of information that you can store in the variable.

DecimalNumber – A number that has a decimal part to it.

WholeNumber – A non decimal number

Boolean – true or false

TextString – Can contain any alphanumeric characters (letters and numbers). Any number stored as a TextString is stored as a character not a numeric value.

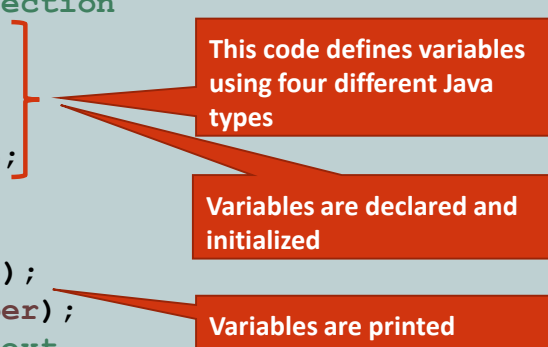
# Variable Data Types in Alice 3

Data Type	Description
Decimal Number	<ul style="list-style-type: none"><li>• Perform arithmetic and set the value of a procedure's arguments</li><li>• Examples: 0.1, 2.25, 98.6</li></ul>
Whole Number	<ul style="list-style-type: none"><li>• Perform arithmetic and set the value of a procedure's arguments</li><li>• Examples: 1, 459, 30</li></ul>
Boolean	<ul style="list-style-type: none"><li>• One of two values: true or false</li><li>• Usually is the result of tests that compare one thing to another</li></ul>
Object	<ul style="list-style-type: none"><li>• Any object in Alice 3 like a cat, dog, person, etc</li></ul>
Classes	<ul style="list-style-type: none"><li>• The classes of objects in your animation</li><li>• Examples: Biped, Scene, Quadraped</li></ul>
TextString	<ul style="list-style-type: none"><li>• A String of characters such as "hello" and "goodbye"</li></ul>
Other	<ul style="list-style-type: none"><li>• Sounds, colors, and other special values</li></ul>

# Data Types in Java Example

```
class PrintText{
    public static void main(String[] args){
        //variable declaration section
        byte aByte = -10;
        int aNumber = 10;
        char aChar = 'b';
        boolean isBoolean = true;

        //print variables alone
        System.out.println(aByte);
        System.out.println(aNumber);
        //print variables with text
        System.out.println("aChar = " + aChar);
        System.out.println("The boolean var is:" + isBoolean);
    } //end method main
} //end class PrintText
```



This code defines variables using four different Java types

Variables are declared and initialized

Variables are printed

The data type must always come before the name of the variable in the declaration. You cannot access a value through the data type only the variable name.

# Java Simple Types

- In Java, there are eight basic data types, called "simple types" or "primitives"

Data Type	Size	Example Data	Data Description
boolean	1 bit	True, false	Store true/false flags
byte	1 byte (8 bits)	12, 128	Store integers from -127 to 128
char	2 bytes	'A', '5', '#'	Store a single Unicode character
short	2 bytes	6, -14, 2345	Store integers from -32,768 to 32,767

You can see by the type of data that they store how they compare to data types in Alice 3.

# Java Simple Types

Data Type	Size	Example Data	Data Description
int	4 bytes	6, -14, 2345	Store integers from -2,147,483,648 to 2,147,483,647
long	8 bytes	3459111, 2	Store integers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	3.145, .077	Store a positive or negative decimal number from $1.4023 \times 10^{-45}$ to $3.4028 \times 10^{+38}$
double	8 bytes	.0000456, 3.7	Store a positive or negative decimal number from $4.9406 \times 10^{-324}$ to $1.7977 \times 10^{308}$

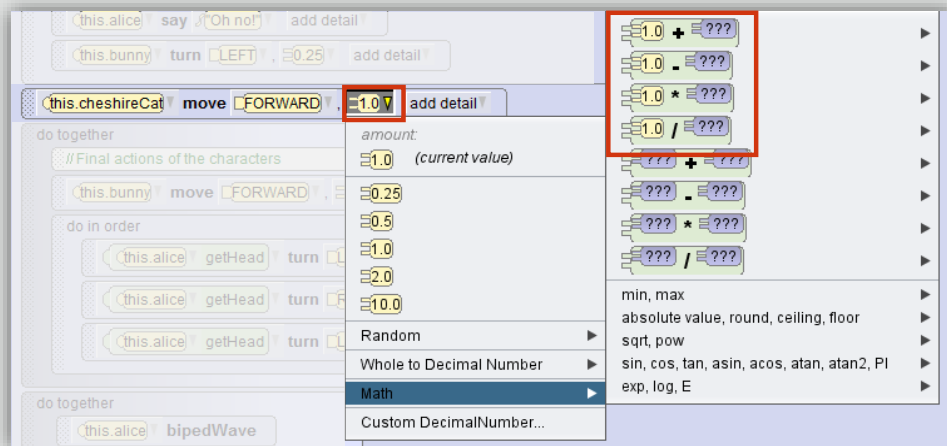
## Arithmetic Operators in Alice 3

- Use add (+), subtract (-), multiply (\*), and divide (/) to create expressions in:
  - Amount and Duration arguments
  - getDistance functions
- The value of an arithmetic operator can be stored in a variable

Arithmetic operators perform basic mathematical operations. They take two operands and return the result of the mathematical calculation.

# Arithmetic Operators in Distance Argument

- Using the drop-down arrow beside the argument value gives you access to the math function



Alice 3 is capable of both simple and complex mathematical expressions.



# Arithmetic Operators in Java

- Arithmetic operators work the same in Alice and Java, but they are accessed differently
- In Java, you include arithmetic operators directly in your code

```
class BasicOperators {  
    //using arithmetic operators  
    public static void main(String[] args) {  
        //variable declaration section  
        int a = 1 + 1;  
        int b = 3 * 3;  
        int c = 1 + 8 / 4;  
        int d = -2;  
        System.out.println("a = " + a);  
        System.out.println("b = " + b);  
        System.out.println("c = " + c);  
        System.out.println("d = " + d);  
    } //end method main  
} //end class BasicOperators
```

The way mathematical equations are used in Java is the same as in most programming languages so understanding these gives you a set of transferable programming skills.

# Relational Operators

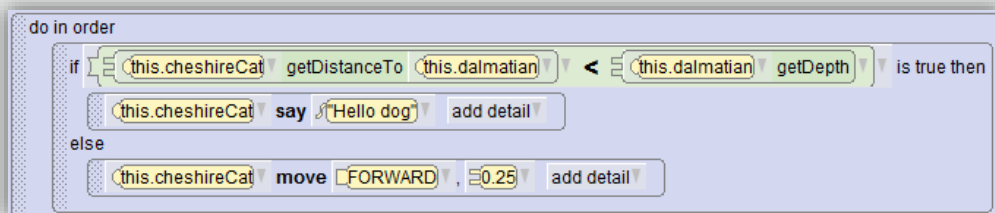
- Relational operators include:  
    `=`    `≠`    `>`    `<`    `≥`    `≤`
- Expressions with relational operators produce true or false values

A relational operator is a lexical unit used to express a relation, such as equality or greater than, between two expressions. Two suitable expressions combined with a relational operator often form a relational expression or condition in a programming language.

Relational operators are used within control statements to evaluate the condition in order to control the program flow.

## Relational Operators in Alice 3

- The following example shows a relational operator used to test the distance between the Cheshire cat and the Dalmatian
- If the distance from the cat to the dog is less than the depth of the dog, the cat says, "**Hello dog!**"
- If the distance is not less, then the cat moves forward .25 meters



This example shows that if the distance between the Cheshire Cat and the Dalmatian is less than the depth of the Dalmatian then the cat will say hello. If the distance is not less than the depth then the cat will move forward.

# Relational Operators in Java

Symbol	Operator Name	Example
==	Equal To	(A == B) is false
!=	Not Equal To	(A != B) is true
>	Greater Than	(A > B) is false
>=	Greater Than or Equal To	(A >= B) is false
<	Less Than	(A < B) is true
<=	Less Than or Equal To	(A <= B) is true

These are traditional mathematical symbols that again are used across most programming languages.

# Relational Operators in Java Example

- Relational operators in Java and Alice work the same way
  - Below is sample Java code containing relational operators

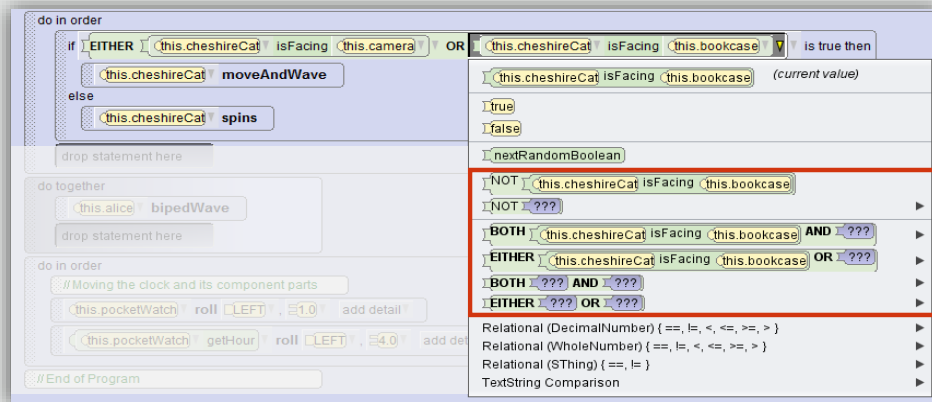
```
class Test {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 20;  
  
        System.out.println("a == b = " + (a == b) );  
        System.out.println("a != b = " + (a != b) );  
        System.out.println("a > b = " + (a > b) );  
        System.out.println("a < b = " + (a < b) );  
        System.out.println("b >= a = " + (b >= a) );  
        System.out.println("b <= a = " + (b <= a) );  
    } //end method main  
} //end class Test
```

This slide uses an output statement (System.out.println()) to display the value of the result of an expression.

# Logical Operators in Alice 3

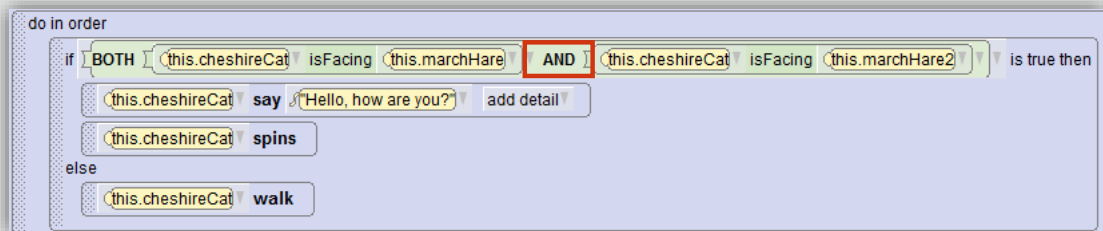
- There are variations of logical operators in Alice 3

Logical operators are the boolean operators AND, OR and NOT. Expressions written with logical operators result in true or false.



## Logical Operators in Alice 3 Example

- The example uses both a and b logical operators to combine two conditions into a single condition
- Both of these conditions must be true for the overall condition to be true
- Otherwise, it will be false



# Logical Operators in Java

- Logical operators work the same in Alice 3 and Java, they are just accessed differently
- In Java, the logical operators are typed using symbols

Symbol	Operator Name
<b>&amp;&amp;</b>	Conditional AND or "Short-circuited Operators"
<b>  </b>	Conditional OR or "Short-circuited Operators"
<b>!</b>	NOT



# Logical Operators in Java Example

- This example demonstrates the NOT operator

```
class BoolNotDemo {  
    public static void main(String[] args){  
        //variable declaration section  
        int x = 2;  
        int y = 1;  
        boolean bl;  
  
        bl = !(x > y); // bl is false  
        System.out.println("x is not greater than y:"+bl);  
  
        bl = !(y > x); // bl is true  
        System.out.println("y is not greater than x:"+bl);  
    } //end method main  
} //end class BoolNotDemo
```

The NOT operator can be confusing to read. The line `bl = !(x > y);` x is greater than y which is true, because of the NOT operator you are looking for the opposite of that. When the true value is passed into the NOT operator it sets it to its opposite value which is false.

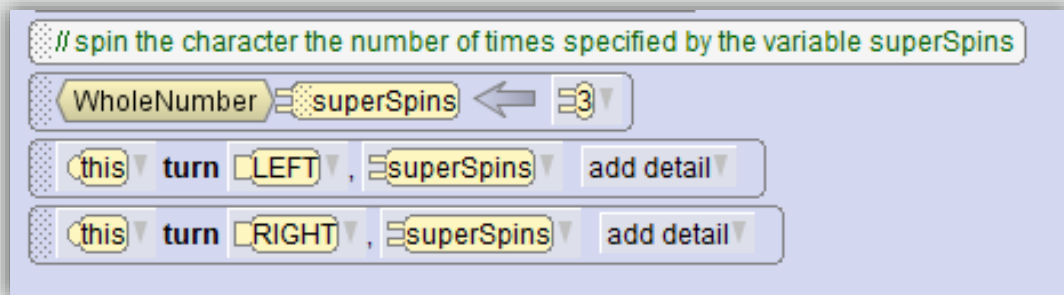
## Assignment Operators in Alice 3

- Assignment operators change the value of a variable
- In the Alice 3 interface, you assign values to properties and variables
- The color, opacity, position, vehicle and size properties are assigned values in the Object Properties



## Assignment Operators in Alice 3 Example

- In this example, the local variable superSpins is assigned an initial value of three
- The number of times the character spins is assigned to superSpins



# Assignment Operators in Java

- In Java, use the equal sign ("=") to assign one value to another
  - In this example, the value of y is 5 and the value of z is 25

```
class AssignmentDemo{
    public static void main(String[] args) {
        //variable declaration section
        int x=5;
        int y=10;
        int z=20;
        y = x;
        z = y + z;

        System.out.println("The value of y is:"+ y);
        System.out.println("The value of z is:"+ z);
    } //end method main
} //end class AssignmentDemo
```

In the code above the statement `y = x;` means that the value of x will be stored in y. The variable on the left of the `=` sign always holds the value of the expression on the right. In this case although y was given an initial value of 10 it takes on the value of x when this line is executed.

z is given the value of the result of the expression `y + z;` this equates to `5 + 20` so z will have a value of 25 when this line executes.

## Handling Standard Operations in Java

- Java provides a special syntax for handling standard operations such as `z = y + z;`
- Syntax is `z += y;`
- This code saves a few keystrokes, but still assigns `z` the value of `y` plus `z`

The statement `z += y;` can be read as `z` is equal to the value currently held by `z` plus the value of `y`.

## Assignment Syntax for Other Operations

Symbol	Example	Equivalent To
<code>+=</code>	<code>x += y</code>	<code>x = x + y;</code>
<code>- =</code>	<code>x -= y</code>	<code>x = x - y;</code>
<code>* =</code>	<code>x *= y</code>	<code>x = x * y;</code>
<code>/ =</code>	<code>x /= y</code>	<code>x = x / y;</code>

Either method will result in the same answer but the assignment syntax means you have less to type.

# Assignment Syntax Code Example

```
class AssignmentDemo2{
    public static void main(String[] args) {
        //variable declaration section
        int x=5;
        int y=10;
        x += y;

        System.out.println("The += result is:"+ x);

        x -= y;
        System.out.println("The -= result is:"+ x);

        x *= y;
        System.out.println("The *= result is:"+ x);

        x /= y;
        System.out.println("The /= result is:"+ x);
    } //end method main
} //end class AssignmentDemo2
```

# Terminology

- Key terms used in this lesson included:
  - Arithmetic operators
  - Assignment operators
  - Data type
  - Logical operators
  - Relational operators
  - Variable



# Summary

- In this lesson, you should have learned how to:
  - Describe variables
  - Describe Java simple types
  - Define arithmetic operators
  - Describe relational and logical operators
  - Describe assignment operators



