

Unit II: Creating Code

Chapter 11: A Little Math in Alice 3

In this chapter, you will learn how to perform simple mathematical calculations in Alice using arithmetic operators. You will also learn how to create and use a counter to keep track of how many times an action occurs and how to calculate a total by adding values as a program runs. These skills will help you use numbers and calculations to control and add functionality to your Alice programs.

Objectives

- Use assign and math to assign a value.
- Create code to do math using just one step.
- Use functions to get information about objects, such as **getHeight**.
- Find a total, percent and other values.

Scene

We have a beach scene with **seller** selling T-Shirts and coke. There is a **tourist**. Items for sale have a price, a **double**. The number of shirts or cokes you buy is an **integer**.



Addition

The tourist asks for a coke and a T-Shirt. As usual, we find the total for the two items one at a time. We start by an assign statement setting total to the priceCoke.

```
void myFirstMethod( )
do in order
  //Beach scene selling T-Shirts and coke to tourists
  //Programmer: Janet Joy
  Double priceTShirt = 10.0 ;
  Double priceCoke = 2.25 ;
  Double total = 0.0 ;
  this.tourist say ("I would like a coke and a T-Shirt." add detail);
  total = priceCoke ;
```

Next, we click the down-arrow after priceCoke, then select priceCoke+???, then select priceTshirt,

void myFirstMethod()

do in order

```
//Beach scene selling T-Shirts and coke to tourists
//Programmer: Janet Joy
```

Double priceTShirt = 10.0 ;

Double priceCoke = 2.25 ;

Double total = 0.0 ;

this.tourist say ("I would like a coke and a T-Shirt.")

total = priceCoke

priceCoke (current value)

- 0.0
- 0.25
- 0.5
- 1.0
- 2.0
- 10.0
- 2.25
- Random
- Whole to Decimal Number
- Math

priceCoke + ???

priceCoke - ???

priceCoke * ???

priceCoke / ???

??? + ???

??? - ???

??? * ???

??? / ???

min, max

absolute value, round, ceiling, floor

sqrt, pow

sin, cos, tan, asin, acos, atan, atan2, log, exp, log, t

0.0

0.25

0.5

1.0

2.0

10.0

2.25

Custom DecimalNumber..

total

priceCoke

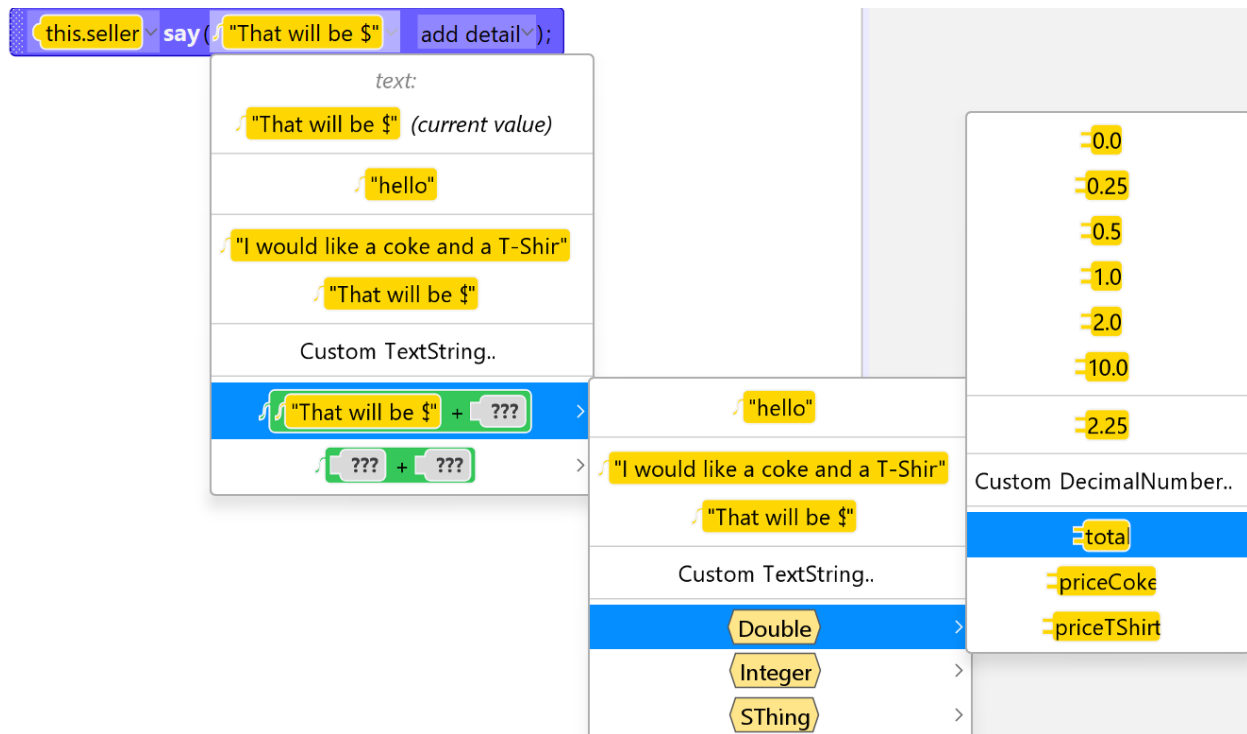
priceTshirt

Now the seller can tell the tourist the total. We also do this one step at a time, First saying "That will be \$", then clicking the down arrow to add total to the sentence.

this.tourist say ("I would like a coke and a T-Shirt.")

total = priceCoke + priceTshirt

this.seller say ("That will be \$")



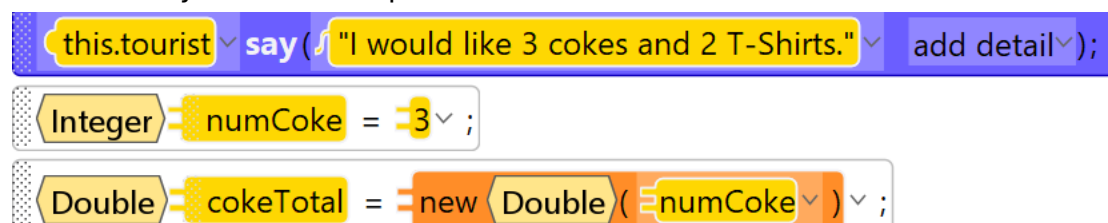
The complete code is:

```
public void myFirstMethod() {
    // Beach scene selling T-Sshirts and coke to tourists
    // Programmer : Janet Joy
    Double priceTShirt = 10.0;
    Double priceCoke = 2.25;
    Double total = 0.0;
    this.tourist.say( "I would like a coke and a T-Shirt." );
    total = priceCoke+priceTShirt;
    this.seller.say( "That will be $" + total );
}
```

When we run it, the seller says **"That will be \$12.25"**

Multiplication

This time , the tourist asks for 3 cokes and 2 T-Shirts. This time, we will find the total for the cokes and another total for the T-Shirts, then add them together. We create a new variable **numCoke** and give it a value of 3. Then we create a variable **cokeTotal** and convert the interger to double so that we can multiply 2 double values. When you build the statement, the * is the symbol for multiplication.



We build the statement to calculate the total for the T-Shirts the same way. Our final code is:

```
public void myFirstMethod() {
    // Beach scene selling T-Sshirts and coke to tourists
    // Programmer : Janet Joy
    Double priceTShirt = 10.0;
    Double priceCoke = 2.25;
    Double total = 0.0;
    this.tourist.say( "I would like 3 cokes and 2 T-Shirts." );
    Integer numCoke = 3;
    Double cokeTotal = new Double( numCoke ) * priceCoke;
    Integer numTShirt = 2;
    Double tshirtTotal = new Double( numTShirt ) * priceTShirt;
    total = cokeTotal + tshirtTotal;
    this.seller.say( "That will be $" + total );
}
```

Testing

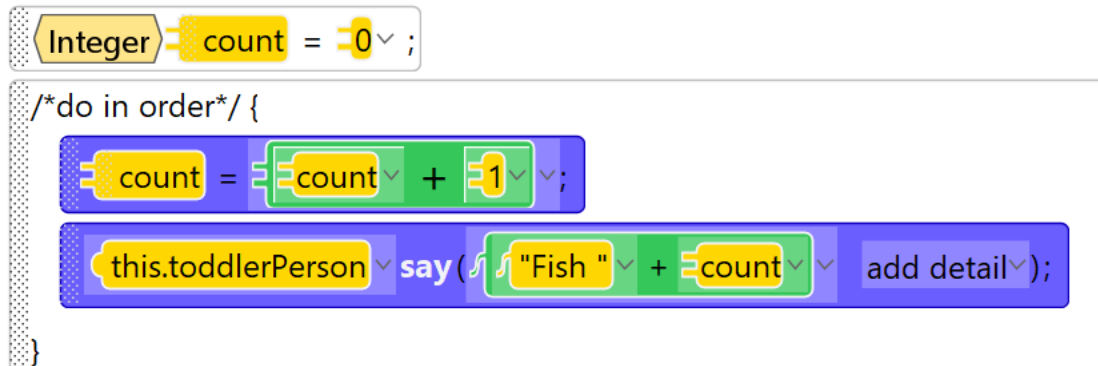
When you test a program like this, you should calculate the result, then run it to see if the code is right. If you just run it, you may not notice if the calculation is correct. When we run this program the seller says “That will be \$26.75” You may want to add some statements to have the seller say “The total for the 3 cokes is \$6.75” then say “The total for the 2 T-Shirts is \$20. Showing the value at each step is especially important if there are numerous steps.

Counting

In this scene there is a toddler in a boat and 4 fish in the water. The toddler will count them. We start with an integer variable count with an initial value of zero.



We want to have 2 statements that we repeat four times. If we put the 2 statements in a do in order block, we can build it once and then copy the block to the clipboard (right click on the do in order block) and paste it by dragging it to the code.



Run to make sure the toddler says one, then copy and paste.

We don't have another person in the scene so when the toddler stops at three, the boat says "There's one more." Then we past the block from the clipboard again to get to 4.

```

public void myFirstMethod() {
    // Toddler counts fish
    // Programmer : Janet Joy
    Integer count = 0;
    {
        // do in order
        count = count+1;
        this.toddlerPerson.say( "Fish " + count );
    }
    {
        // do in order
        count = count+1;
        this.toddlerPerson.say( "Fish " + count );
    }
    {
        // do in order
        count = count+1;
        this.toddlerPerson.say( "Fish " + count );
    }
    this.fishingBoat.say( "There's one more." );
    {
        // do in order
        count = count+1;
        this.toddlerPerson.say( "Fish " + count );
    }
}

```

Increment

Starting with a value of 0, then adding one to it is called **incrementing**. It is very common in programming and very useful. For instance, we might want to count how many questions a student get right on a quiz, or how many customer come into the store. The opposite, subtracting 1 is called **decrementing**. It is not as common as incrementing.

Running Total

Loki and Thor are taking inventory; counting the number of books on the bookshelf. We have both **count** and **total**. The counter is used to count the shelves, total is the total number of books. The code will be the same for each shelf, so we put the statements in a **do in order** block so that we can use the clipboard to copy and paste. We use a variable statement to initialize the variables, then use assignment to give the variables a new value. The variable number is created in each **do together** block, because its **scope** is only the block where it is created. If we had declared it along with count and total, we could use **assign** to give it a new value each time. The code shows only the first two shelves.



```
public void myFirstMethod() {
    // Counting Books
    // Programmer : janet Joy
    Integer count = 0;
    Integer total = 0;
    {
        // do in order
        count = count+1;
        this.thor.say( ( "How many books on shelf " + count ) + "?" );
        Integer number = 9;
        this.loki.say( "There are " + number );
        total = total+number;
        this.thor.say( ( "That's " + total ) + " so far." );
    }
    {
        // do in order
        count = count+1;
        this.thor.say( ( "How many books on shelf " + count ) + "?" );
        Integer number = 12;
        this.loki.say( "There are " + number );
        total = total+number;
        this.thor.say( ( "That's " + total ) + " so far." );
    }
}
```

Division

Three monkeys find a bunch of bananas and agree to share them equally. In Alice ((and most programming language) there are two kinds of division. Integer/integer and double/double. With integers $20/3$ is 6. The remainder can be found with %: $20\%3$ is 2. With double values, $20.0/3.0$ is 6.666666.

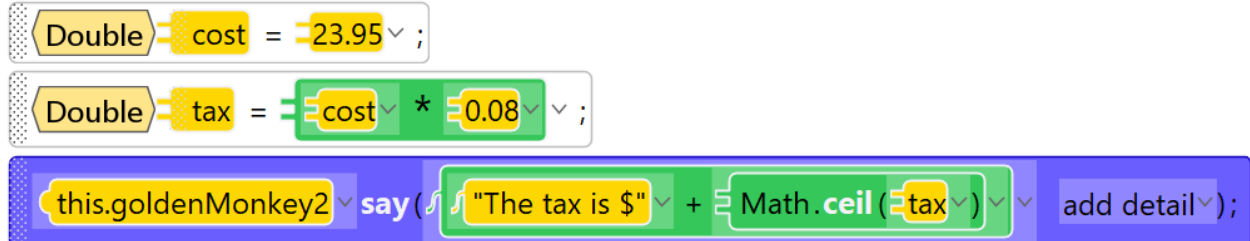


Here is the code for sharing the bananas, the first part using integers, the last part using double.

```
void myFirstMethod()
do in order
    //Division: Monkeys share bananas
    //Programmer: Janet Joy
    Integer numBananas = 20;
    Integer bananasEach = numBananas / 3;
    Integer extraBananas = numBananas % 3;
    this.goldenMonkey2 say ("We each get " + bananasEach + " bananas");
    this.goldenMonkey3 say ("And there will be " + extraBananas + " bananas left over.");
    //Using double values:
    Double eachBananas = new Double(numBananas) / 3.0;
    this.goldenMonkey2 say ("We each get " + eachBananas + " bananas");
```

Percent

In the code below, we find a tax of 8% by multiplying the cost by 0.08. That comes out to \$1.916 we can select from various functions to display the value we want.



Rounding and Other Functions

In the code above, if you click the down-arrow after **eachBanana** you will see several **functions** at the bottom of the pop-up window. Here you will find min, max, floor, ceiling, round, pow (raise to a power) , and some trigonometry functions.

