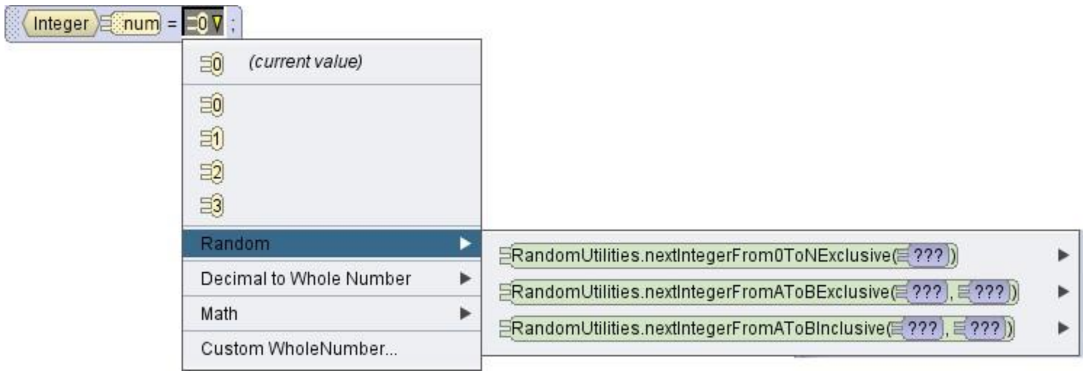
Random numbers make it possible to create games and to make the movements of the characters more natural. We can use random numbers in many ways.

One thing we can do with random users is to flip a coin or select a card from a deck. Those are things that we think of as random.

We can also use random numbers to make movement seem more natural. When a person jumps up and down, they don't jump exactly the same amount each time, so we could have them jump a random amount within a range.

Random Integer Values

If we declare an integer num with an initial value of 0, then click to change the value, you will see the choices shown below.

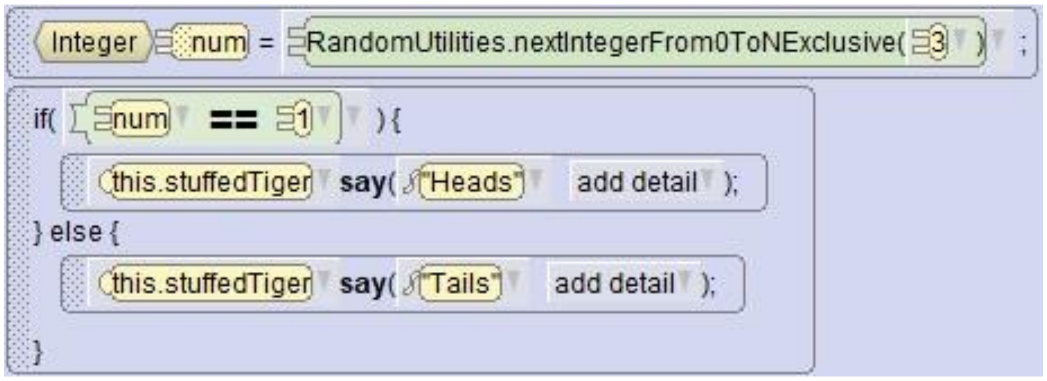

Integer num = 0 ;
0 (current value)
0
1
2
3
Random
Decimal to Whole Number
Math
Custom WholeNumber...
RandomUtilities.nextIntegerFrom0ToNExclusive(???)
RandomUtilities.nextIntegerFromAToBExclusive(???, ???)
RandomUtilities.nextIntegerFromAToBInclusive(???, ???)

The first random choice is From 0 to N **exclusive**. There is one argument, that would be N. If we select 4 for the value of the argument, the value of num will be 0, 1, 2, or 3. The value 4 is excluded.

The second random choice is From A to B Exclusive. There are two arguments that will be A and B. If we select 2 for the first argument (A) and 5 for the second argument (B), the value of num will be 3 or 4. Since it is exclusive, the values of A and B are not values that will be assigned.

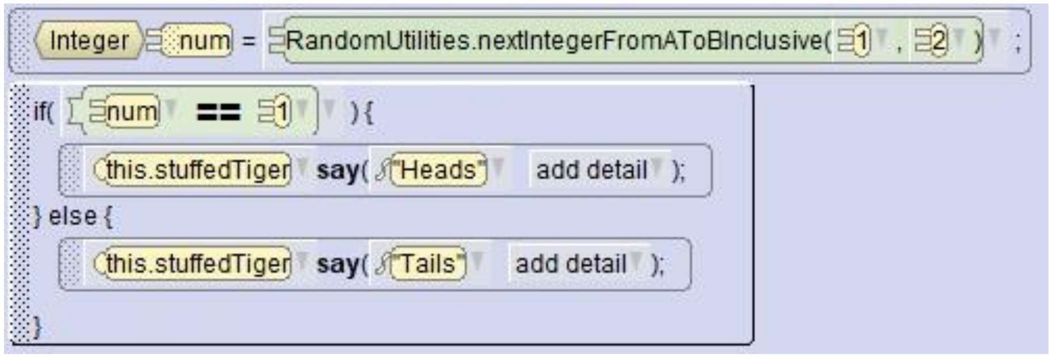
The third random choice is From A to B Inclusive. There are two arguments that will be A and B. If we select 2 for the first argument (A) and 5 for the second argument (B), the value of num will be 2, 3, 4 or 5. Since it is inclusive, the values of A and B are values that might be assigned.

If we wanted to flip a coin, we might use the code shown here: With 0 to N Exclusive, where n = 3, the value of num would be 1 or 2.



```
Integer num = RandomUtilities.nextIntegerFrom0ToNExclusive( 3 ) ;
if( num == 1 ){
    this.stuffedTiger say( "Heads" add detail );
} else {
    this.stuffedTiger say( "Tails" add detail );
}
```

We could have also used from A to B inclusive where A is 0 and B is 1. The possible values are 0 and 1.

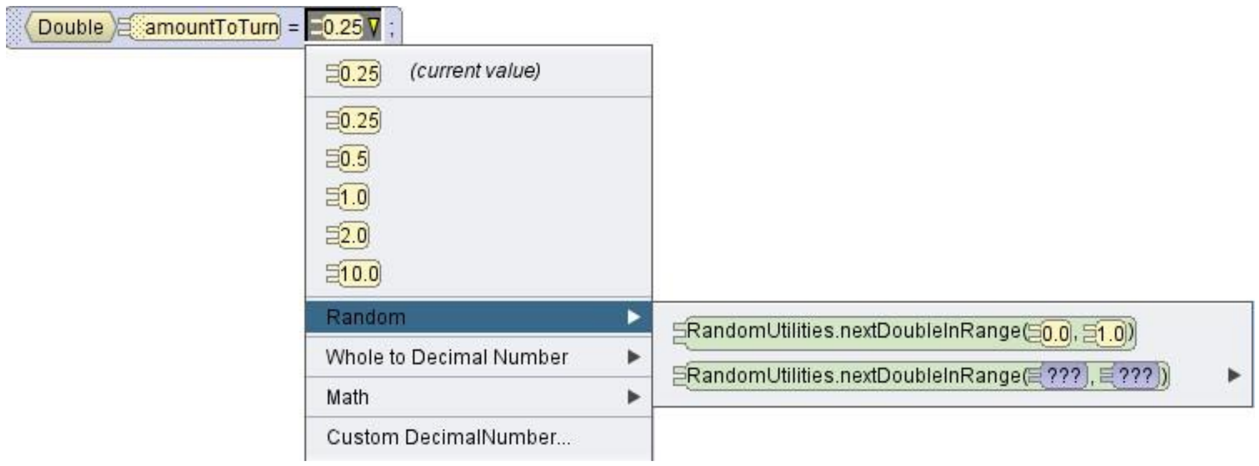


```
Integer num = RandomUtilities.nextIntegerFromAToBInclusive( 1 , 2 ) ;
if( num == 1 ){
    this.stuffedTiger say( "Heads" add detail );
} else {
    this.stuffedTiger say( "Tails" add detail );
}
```

Challenge: How would you roll dice by selecting a random value of 1, 2, 3, 4, 5, or 6?

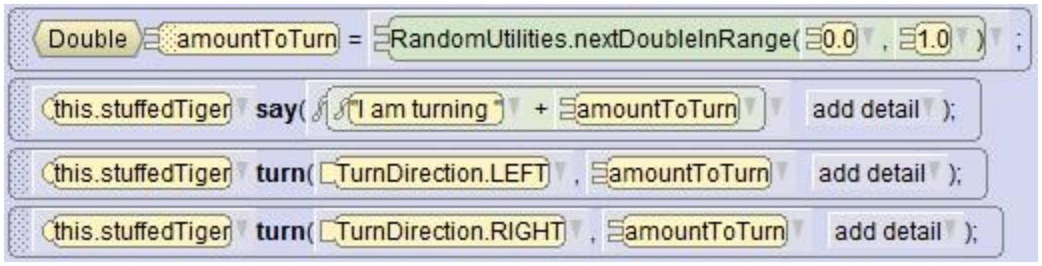
Random Decimal Values

The amount an object moves, turns or rolls is always a double value. If we have the stuffed tiger turn some amount to the left and then turn the same amount to the right, he will be back in the same position he started. We will use a random amount. When we declare a Double variable amountToTurn and then click to change the value, there are two random choices: next Double in range 0.0 to 1.0 and next Double in range with two arguments.

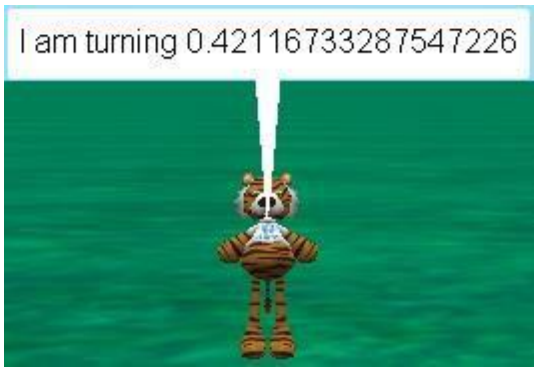

Double amountToTurn = 0.25 ;

These two functions include the first value in the possible values, but not the second. If we select the first random option the value of amountToTurn could be 0.0, but not 1.0. It will be >= 0.0 and < 1.0.

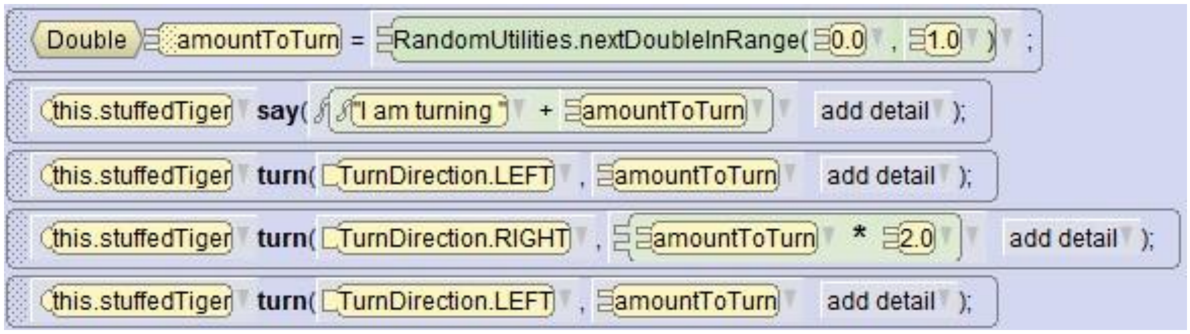
Here is the code to make the stuffed tiger turn.


```
Double amountToTurn = RandomUtilities.nextDoubleInRange( 0.0 , 1.0 ) ;
this.stuffedTiger say( "I am turning " + amountToTurn add detail );
this.stuffedTiger turn( TurnDirection.LEFT , amountToTurn add detail );
this.stuffedTiger turn( TurnDirection.RIGHT , amountToTurn add detail );
```

Including a statement for the tiger to say the amount he is turning makes it easier to understand what is happening. That statement can be deleted or disabled later.


I am turning 0.42116733287547226

Can you guess what this code does?



```
Double amountToTurn = RandomUtilities.nextDoubleInRange( 0.0 , 1.0 ) ;
this.stuffedTiger say( "I am turning " + amountToTurn add detail );
this.stuffedTiger turn( TurnDirection.LEFT , amountToTurn add detail );
this.stuffedTiger turn( TurnDirection.RIGHT , amountToTurn * 2.0 add detail );
this.stuffedTiger turn( TurnDirection.LEFT , amountToTurn add detail );
```

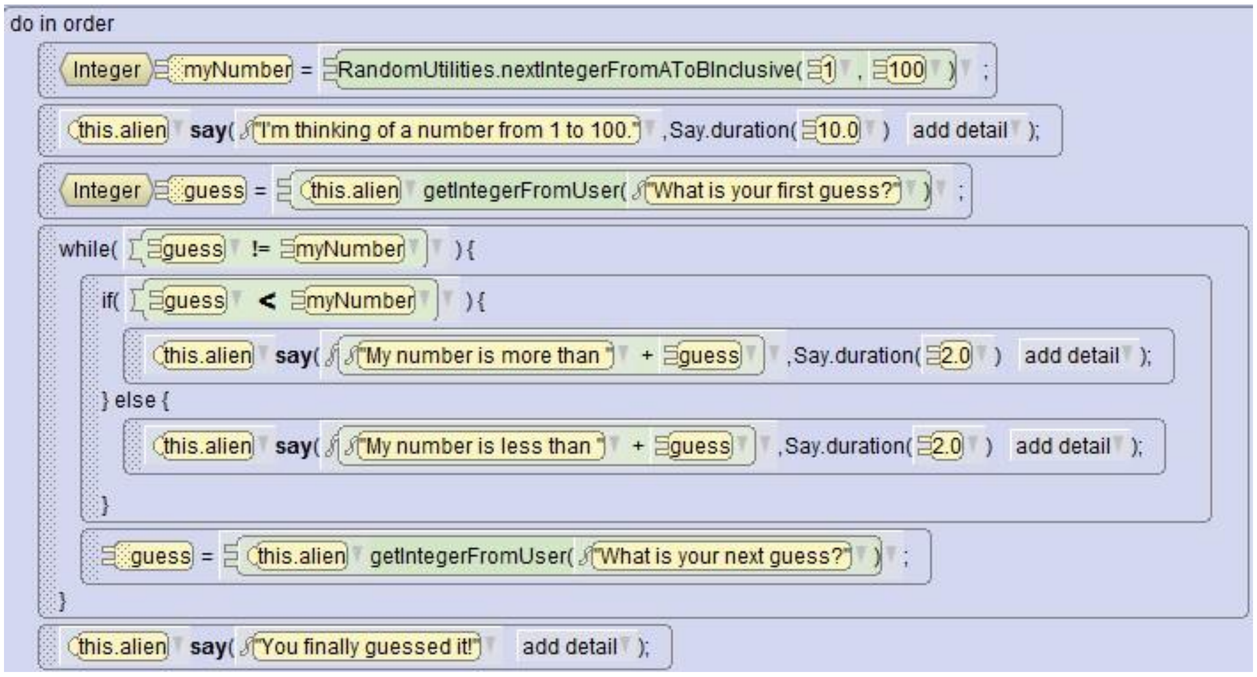
Let's suppose that the value of amount to turn is 0.5. He turns to the left 0.5. Then he turns to the right 1.0, then he turns to the left 0.5. He turns to face left, then he turns to face the right, then he ends by facing forward again.

A Guessing Game

In the Alien guessing game, myNumber is a random integer from 1 to 100. The user must make a guess. The alien will tell if you have guessed it or if his number is higher or lower. Download the program from http://zebra0.com/alice/projects/alien-number.a3p Run the program, then make the following modifications:

Run the program, then modify as below.

- Count how many guesses the user takes to guess the number.
- Keep track of what the user should know is the min and max.
- Tell the user if he makes a mistake: i.e. If he guesses 50 and you say its higher, then later he guesses 40, that is a mistake.
- Write a new game where the user thinks of a number and the alien tries to guess it.



```
do in order
  Integer myNumber = RandomUtilities.nextIntegerFromAToBInclusive( 1 , 100 ) ;
  this.alien say( "I'm thinking of a number from 1 to 100." ,Say.duration( 10.0 ) add detail );
  Integer guess = this.alien getIntegerFromUser( "What is your first guess?" ) ;
  while( guess != myNumber ){
    if( guess < myNumber ){
      this.alien say( "My number is more than " + guess ,Say.duration( 2.0 ) add detail );
    } else {
      this.alien say( "My number is less than " + guess ,Say.duration( 2.0 ) add detail );
    }
    guess = this.alien getIntegerFromUser( "What is your next guess?" ) ;
  }
  this.alien say( "You finally guessed it!" add detail );
```

Experiment! Try different ranges of values. Try making the duration a facto9r of the distance with a little bit of randomness. Add a third marker and use a random value to decide wich marker to swim to.

A Penguin Jumping

Download the Penguin Jumping movie from http://zebra0.com/alice/projects/penguinjumping.a3p, then experiment! Try different ranges of values. Try making **amountToFlap** a random amount between a little less than **amountToJump** and a little more. Try to make a natural looking jump.

```
public void
myFirstMethod() {
  Double amountToJump = 0.5;
Double amountToFlap = 0.25;   while
( true ) {
    amountToJump = RandomUtilities.nextDoubleInRange( 0.1, 0.7 );
amountToFlap = amountToJump/2.0;     doTogether( ()-> {
      this.penguin.getLeftWingShoulder().turn( TurnDirection.BACKWARD, amountToFlap );
        }, ()-> {
      this.penguin.getRightWingShoulder().turn( TurnDirection.BACKWARD, amountToFlap );
        }, ()-> {
      this.penguin.move( MoveDirection.UP, amountToJump );
        } );
doTogether( ()-> {
      this.penguin.getLeftWingShoulder().turn( TurnDirection.FORWARD, amountToFlap );
        }, ()-> {
      this.penguin.getRightWingShoulder().turn( TurnDirection.FORWARD, amountToFlap );
        }, ()-> {
      this.penguin.move( MoveDirection.DOWN, amountToJump );
        } );
    }
}
```

A Fish Swims Back and Forth

Download the movie from http://zebra0.com/alice/projects/fish-back-forth.a3p, then experiment until you understand what each part of the code does and how it works.

```
public void myFirstMethod() {
  while ( true ) {
    Double distance = RandomUtilities.nextDoubleInRange( 0.25, this.clownFish.getDistanceTo(
this.onLeft ) );
    this.clownFish.turnToFace( this.onLeft );
this.clownFish.move( MoveDirection.FORWARD, distance );
this.clownFish.turnToFace( this.onRight );
    distance = RandomUtilities.nextDoubleInRange( 0.25, this.clownFish.getDistanceTo(
this.onRight ) );
    this.clownFish.move( MoveDirection.FORWARD, distance );
    }
}
```