Alice 3

A Java Primer

Ashok.R

Introduction

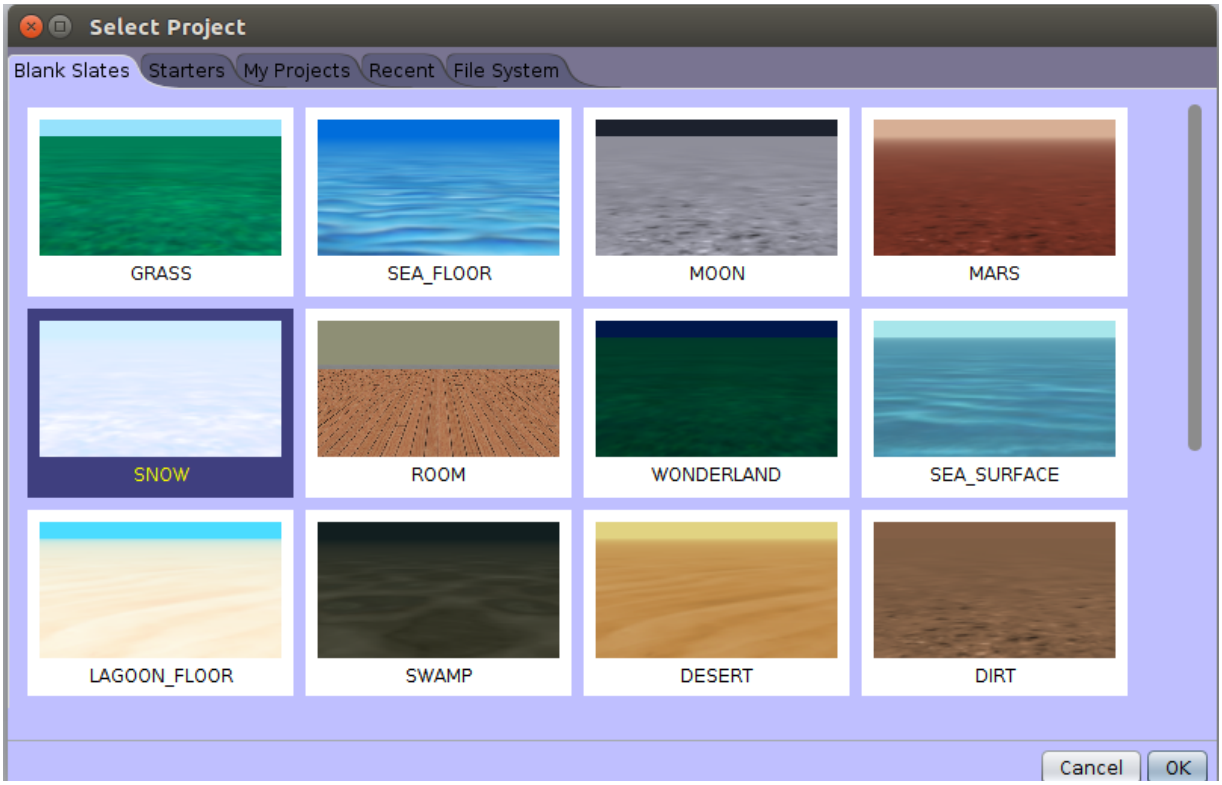
Learn OOP concepts with characters interacting in 3D environment

Java is the No.1 choice by industry to develop smart phone applications, enterprise solutions and video games. From desktops to cloud, java usage is everywhere. Java is used in more than 3 billion devices. Java is the object oriented programming language.

Alice3, a java based tool from Carnegie Mellon University, introduces students with no prior programming experience to object oriented programming concepts in a drag-and-drop 3D environment. Alice3 is encourages students to be creative. Alice3 eliminates complex program syntax and develops computational thinking. Alice3 helps to understand how an algorithm is designed and implemented in a program to solve a problem.

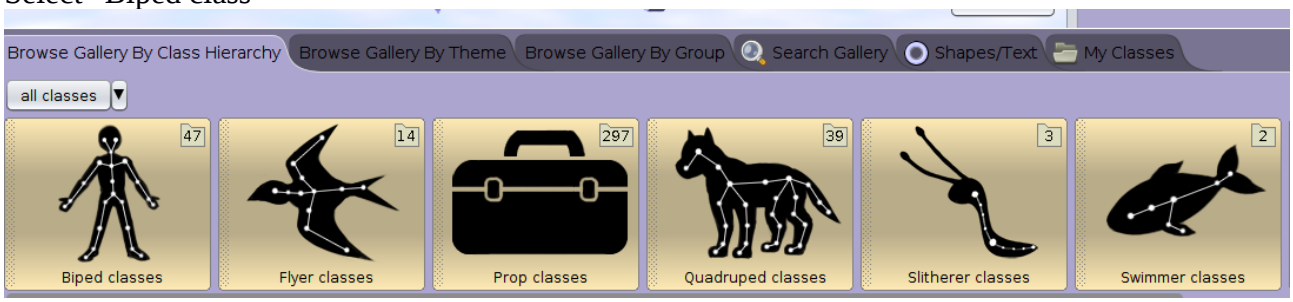
Hello World

Open Alice 3. Select “Snow” ground as the blank slate.



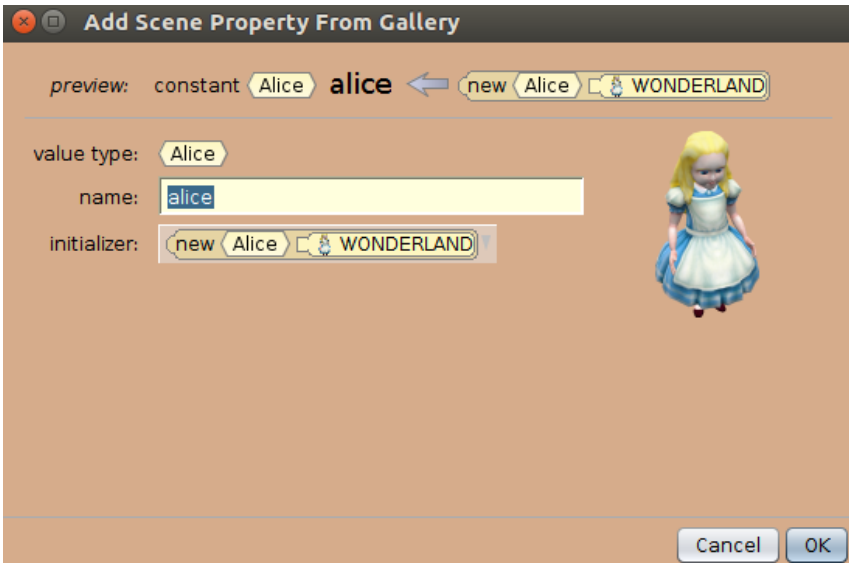
Goto “Setup Scene”

Select “Biped class”



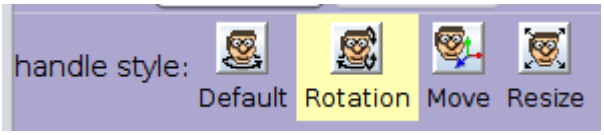
Pull “Alice” and place her in the scene.

Keep the object “alice” default settings as is.

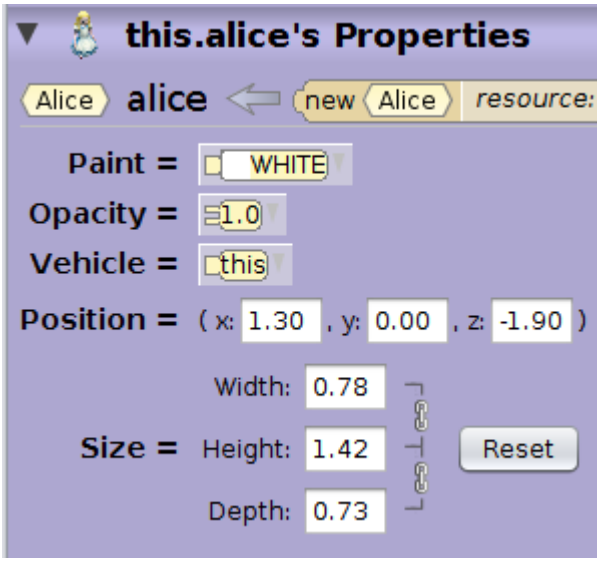


Press on the circle around ‘alice’ and move the cursor to rotate her.

You can change the handle style to move, rotate or resize ‘alice’.



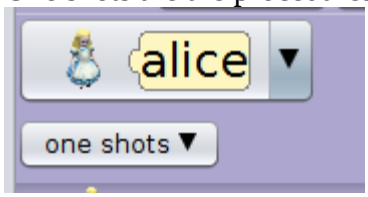
You can also adjust the size and position of ‘alice’ using her object properties:

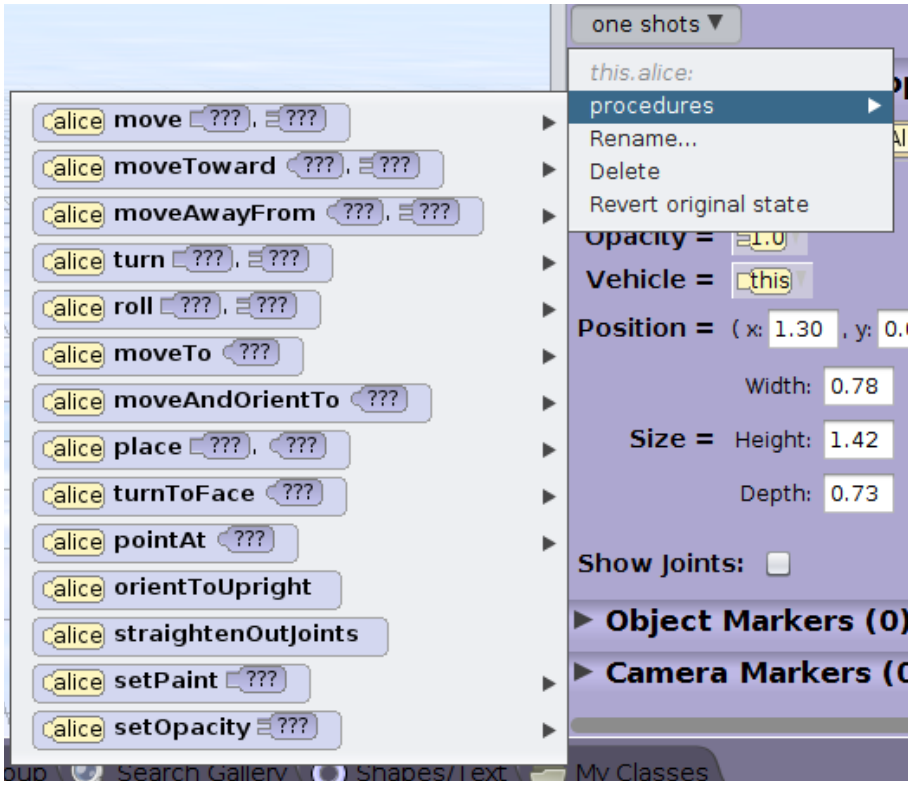


Note:
1. Paint holds the value of her color
2. Opacity decides her visibility. (0 – Completely invisible ; 1 – Fully visible; 0.5- Half transperant)
3. Vehicle attaches alice with any object. By default, all the objects including her are attached to the scene object (denoted by ‘this’).
4. Increasing ‘x’ will shift alice to the left. Decreasing shifts to the right.
5. Increasing ‘z’ shifts alice backward. Decreasing shifts frontward.
6. Increasing ‘y’ shifts alice up.

One shots

One shots are the procedures associated with the selected object.





You can choose ‘Turn’ -> left-> ‘0.25’ to turn alice 1/4th of the full rotation.

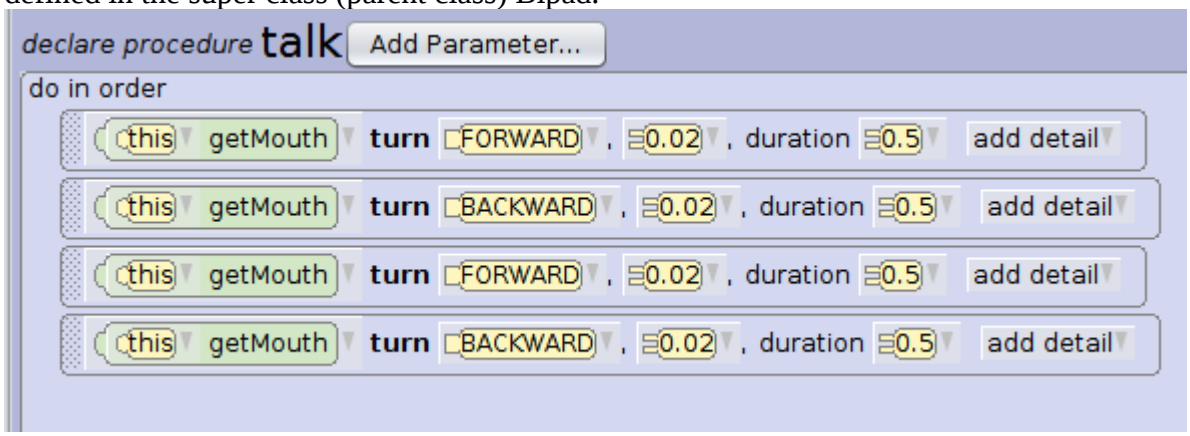
Click ‘Edit code’ to start coding.

'Run' the program.

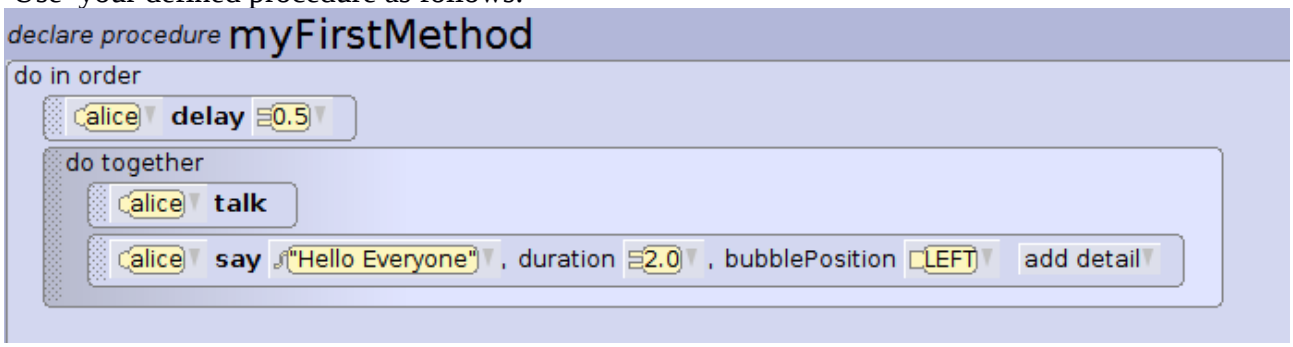
Output:
'alice' turns and says 'Hello World'.

User defined procedure

To make the mouth movement during talking, use the following user defined procedure 'talk' defined in the super class (parent class) Bipad.



Use your defined procedure as follows.



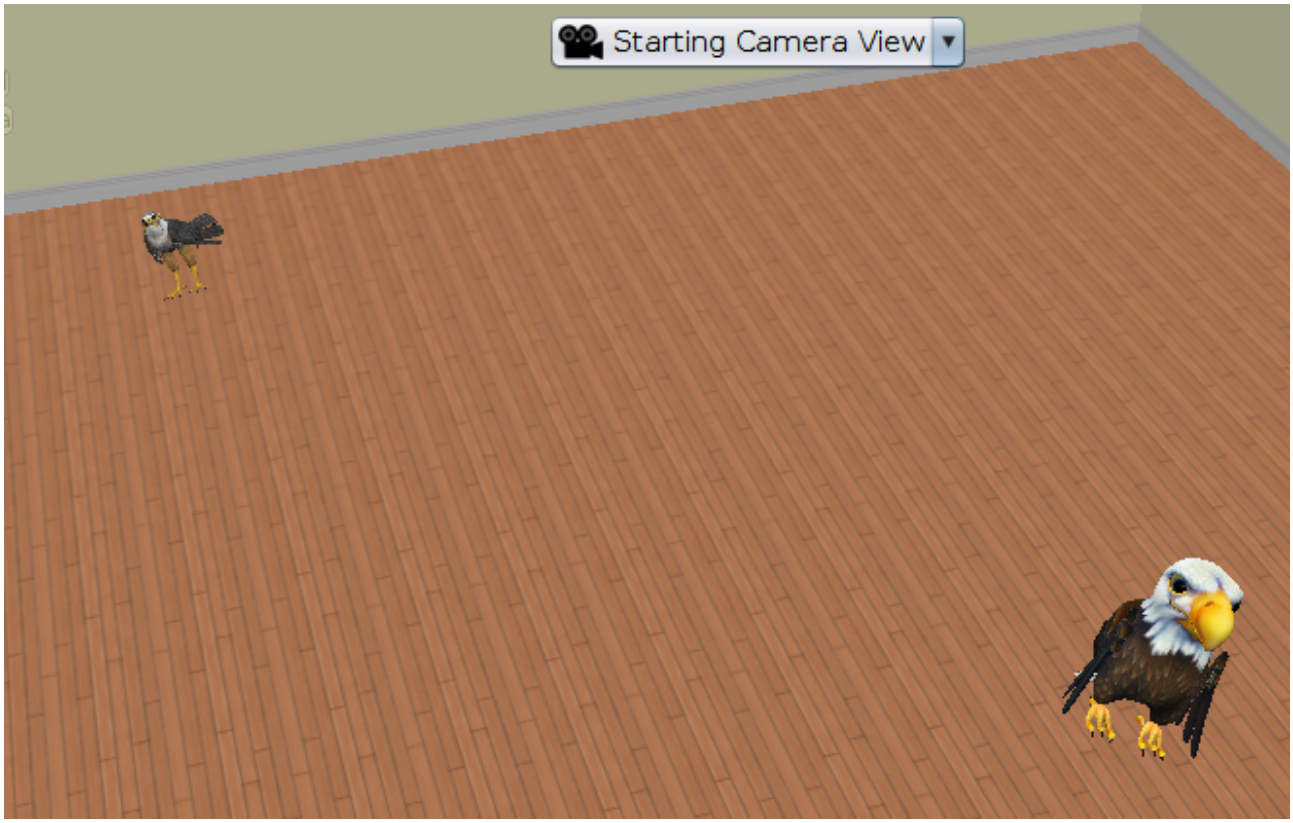
The bigger message can be synched with talk using duration and count loop as shown.

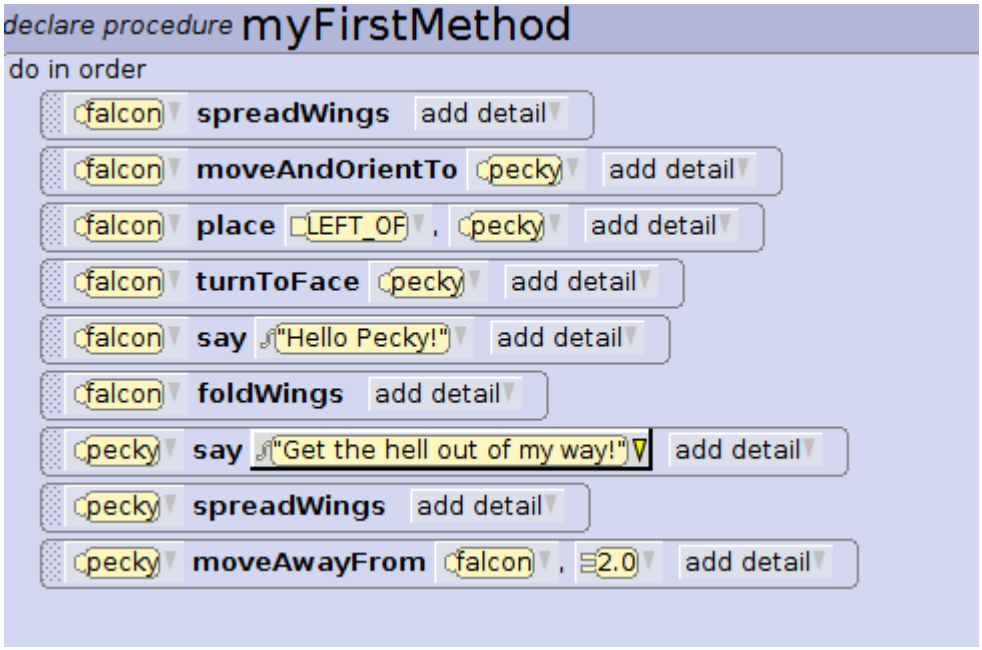
Note: A variable ‘msg’ is used to hold your message.

Save and close the project.

Falcon and Eagle

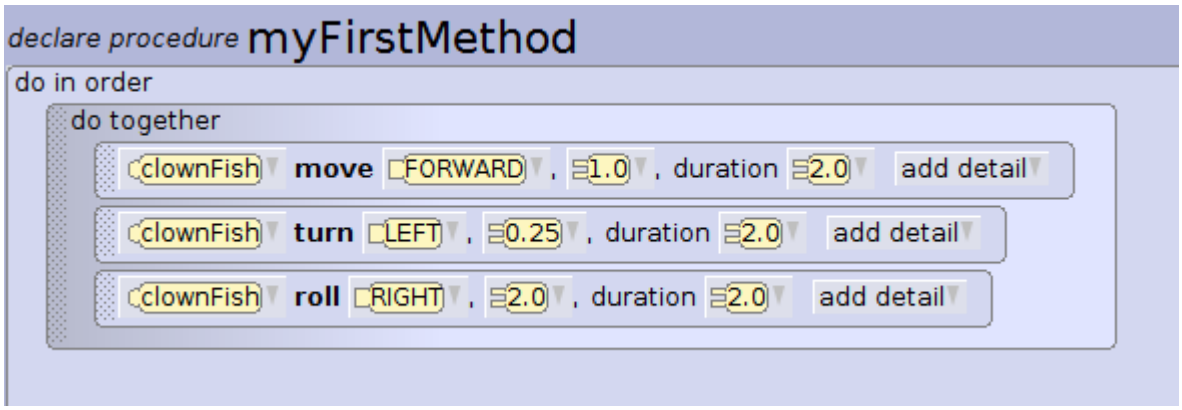
Create new one with the following scene setup with eagle and falcon on the room ground as shown.





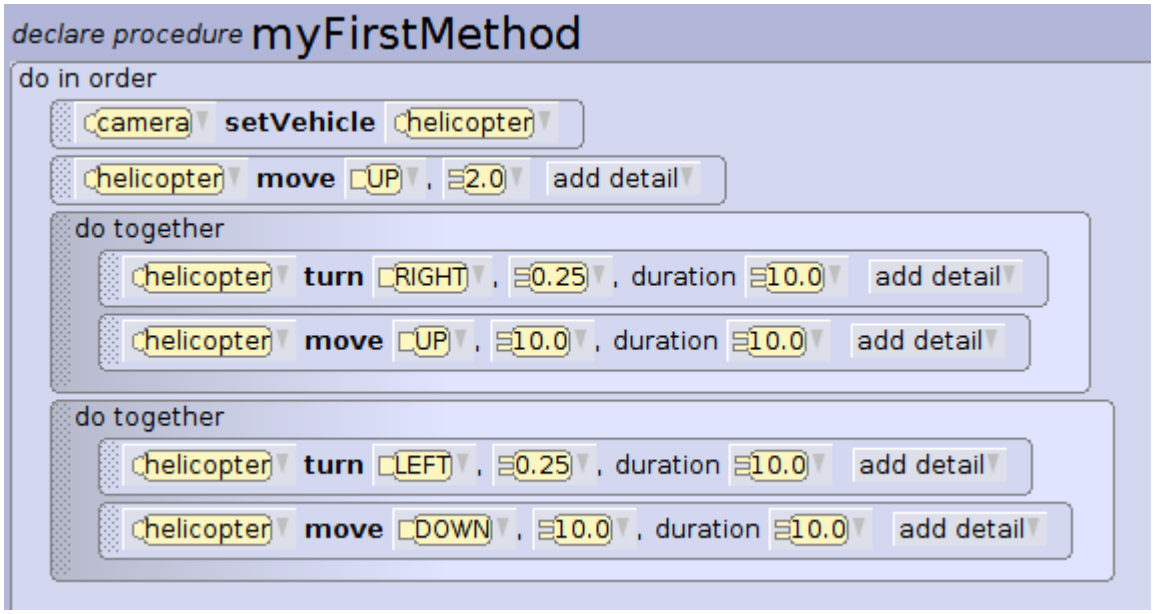
Fish Turns

Create new project. Select “Sea surface at night” as ground and place a clownFish as shown.



Helicopter flies

Place a helicopter in a ground.



SetVehicle procedure attaches Camera with helicopter. Camera then moves, whenever helicopter moves.

Program

A program is the collection of instructions/statements, when executed, solves a particular problem.

Example:
A simple program to serve dinner:

1) Serve salad
2) Serve rice
   A) If ‘non veg’, serve chicken curry
   B) Else serve veg curry
3) Serve curd
4) If guest wants more, repeat steps 2 and 3

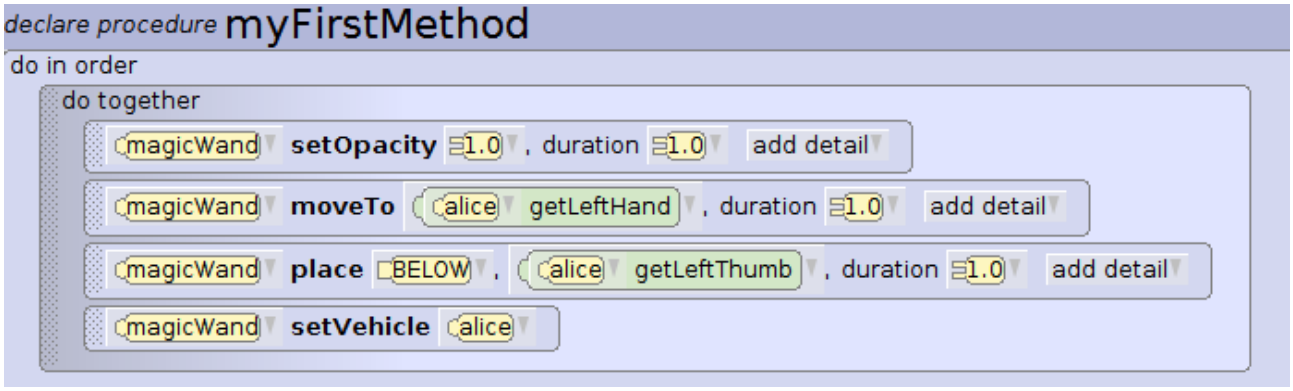
A program generally follows the sequence of instructions (Sequencing). When a decision has to be made, selection statements (IF...ELSE) are used. When things need to be repeated till a condition is met, iteration statements (WHILE or COUNT LOOP).

Basic building blocks of any program are:

1. Sequencing
2. Selection
3. Iteration

Sequencing: Alice gets magic wand

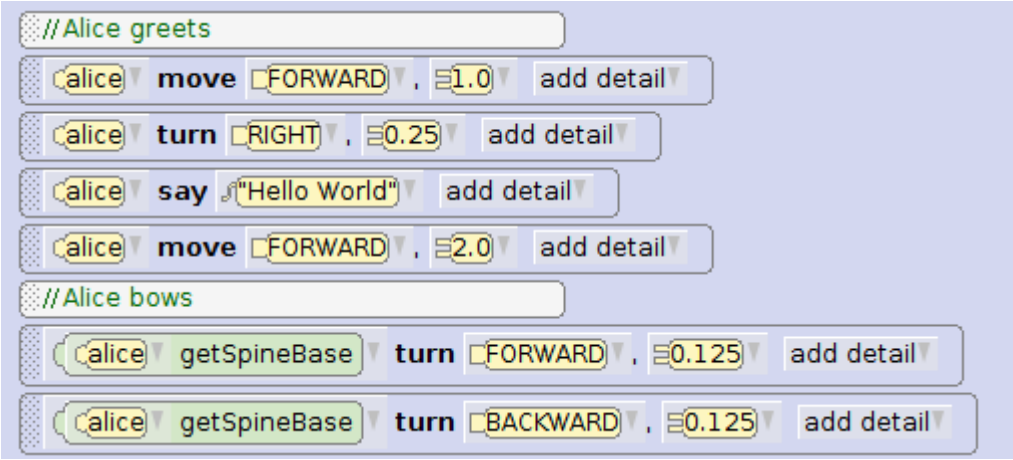
Setup a new scene with alice and magic wand. Set the ‘opacity’ of magic wand to 0 so that it does not appear.



All the statements inside the do-together block are executed in parallel (**multi-threading**).

Sequencing: Hello World

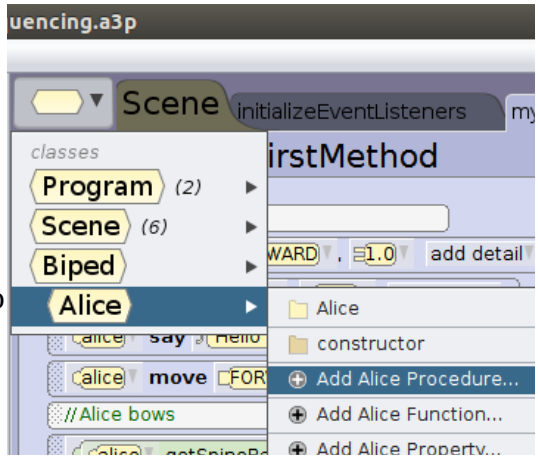
Continue ‘Hello World.a3p’.



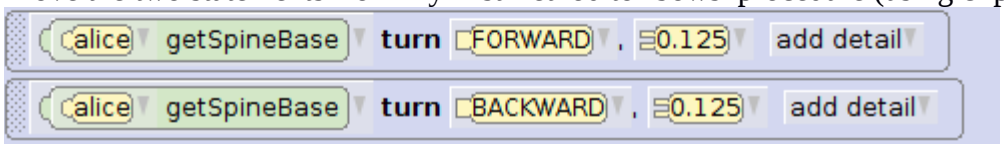
All the procedures in the above program are executed sequentially. You may also add more sequencing procedures to make ‘alice’ wave right hand.

User defined Procedure

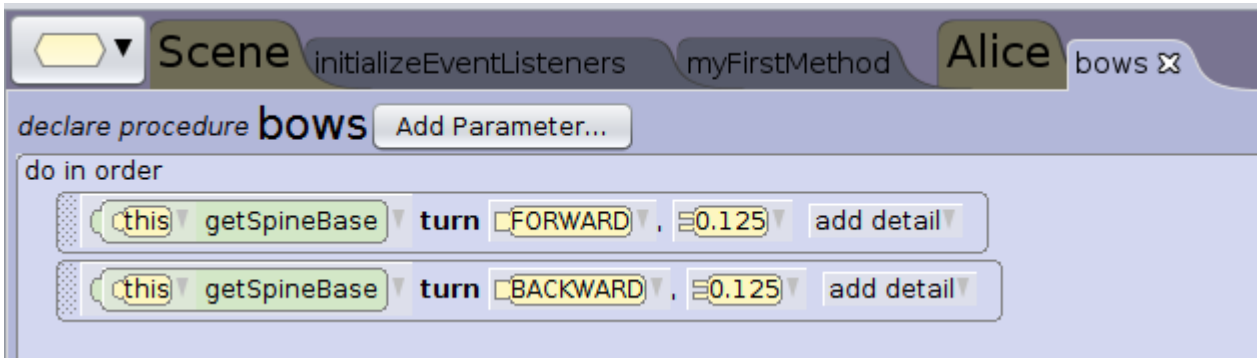
Procedures are actions performed by the particular object. Procedures associated with the particular object are defined in its respective class. If you want to create your own procedure such as ‘bows’, you need to create in the class of ‘alice’ which is ‘Alice’.



Move the two statements from myFirstMethod to ‘bows’ procedure (using clipboard)



and change ‘alice’ to ‘this’ object. As ‘alice’ object is not visible inside her own class, we refer all the objects that belong to ‘Alice’ class by ‘this’ pointer. ‘this’ refers to the corresponding object which calls the ‘bows’ procedure.



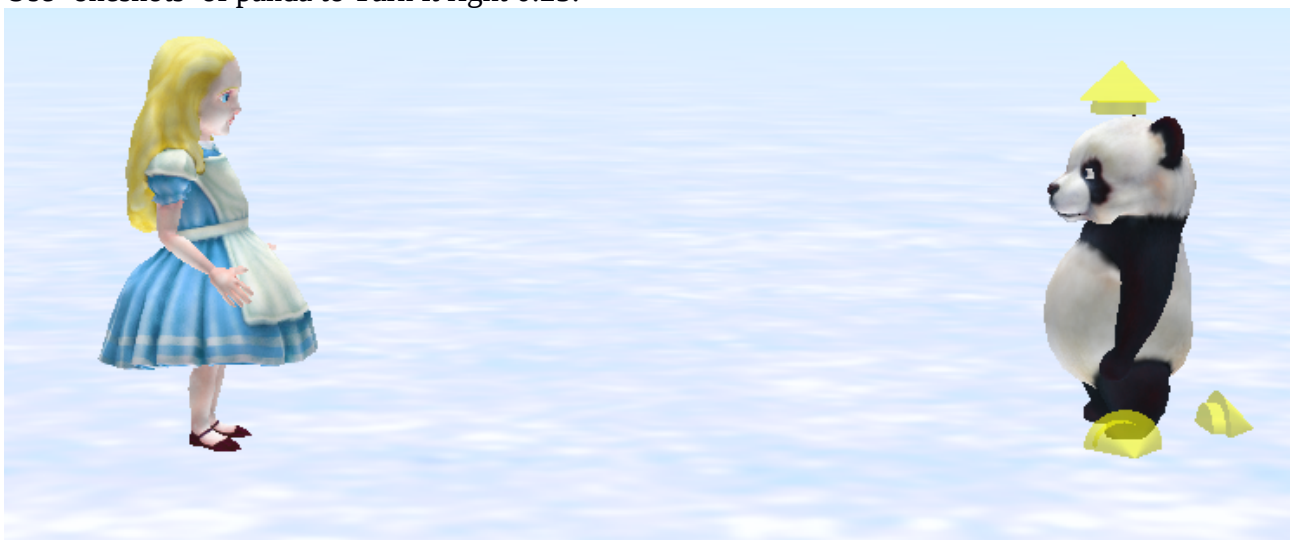
Similarly, you may create ‘wavesRightHand’ procedure as shown:

Now, the program becomes much simpler. Also, you can recall the above procedures whenever required.

This is known as '**Abstraction**'.

Do it together

Now, goto ‘setup scene’ and add another object ‘panda’.
Use ‘oneshots’ of panda to Turn it right 0.25.



```
//Alice and panda greets
do together
    alice move FORWARD , 1.0 add detail
    panda move FORWARD , 1.0 add detail
do together
    alice turn RIGHT , 0.25 add detail
    panda turn LEFT , 0.25 add detail
do together
    alice say "Hello World" add detail
    panda say "Hello World" add detail
do together
    alice move FORWARD , 2.0 add detail
    panda move FORWARD , 2.0 add detail
```

But, panda doesn’t have procedures to bow and wave hands. We can make the procedures accessible to all objects of class ‘bipad’, if we define the procedures in ‘bipad’ class, which is **parent class** for both ‘alice’ and ‘panda’. This is known as **Inheritance**.

Move the statements from ‘bows’ procedure of ‘Alice’ to ‘bows’ procedure of ‘bipad’.
Similarly do it for ‘waveRightHand’ for ‘bipad’.

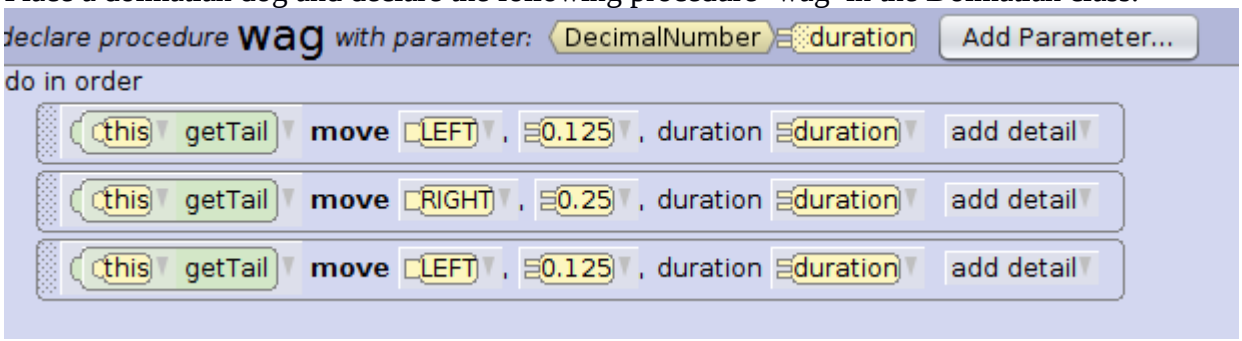
Now, try to run the following code in MyFirstMethod.

OOPs. Alice does nothing. Because, we forget to delete the empty procedures remaining in 'Alice' class. When a method of same name is found in child class, it overrides the method of parent class. This is known as 'Method overriding'. To rectify the issue, delete the empty procedures 'bows' and 'wavesRightHand' in Alice class.

Note: You need to remove procedure calls first, before removing corresponding proedures.

Dog wags its tail

Place a dolmatian dog and declare the following procedure ‘wag’ in the Dolmatian class.



Selection: Who Jumps

Place alice, a playingcard and a cheshire cat on the snow ground as shown.

```
declare procedure myFirstMethod
do in order
    if this getBooleanFromUser "Do you want to Alice to jump?" is true then
        this jump who: alice , height: 0.5
    else
        if this getBooleanFromUser "Do you want to cardMan to jump?" is true then
            this jump who: playingCard , height: 0.25
        else
            this jump who: cheshireCat , height: 1.0
```

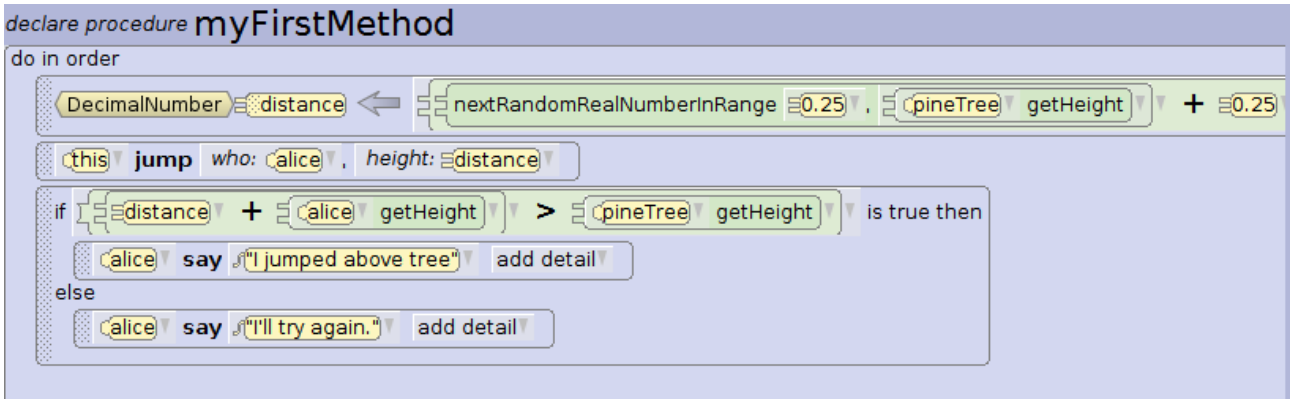
The ‘jump’ procedure is declared in the ‘Scene’ class, as we need to pass objects attached with the scene object as parameter (who).

```
declare procedure jump with parameters: SBiped who , DecimalNumber height Add Parameter...
do in order
    who move UP , height , duration 1.0 add detail
    who move DOWN , height , duration 1.0 add detail
```

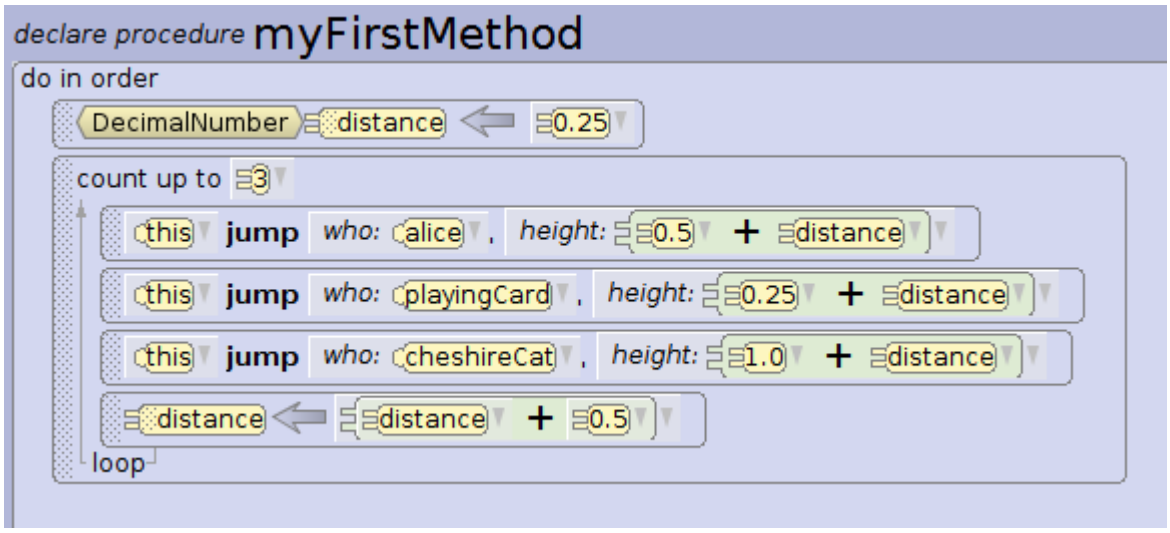
Selection: Who Jumps Wins

Insert a pine tree in the above example.

Let us set a random number between 0.25 and the height of the tree as the distance to jump. Distance is increased by 0.25 to increase the distance further. If the height of the person who jumps, when added with the distance he/she jumps, exceed the height of the tree, he/she wins. Else, he/she will try again. (restart).



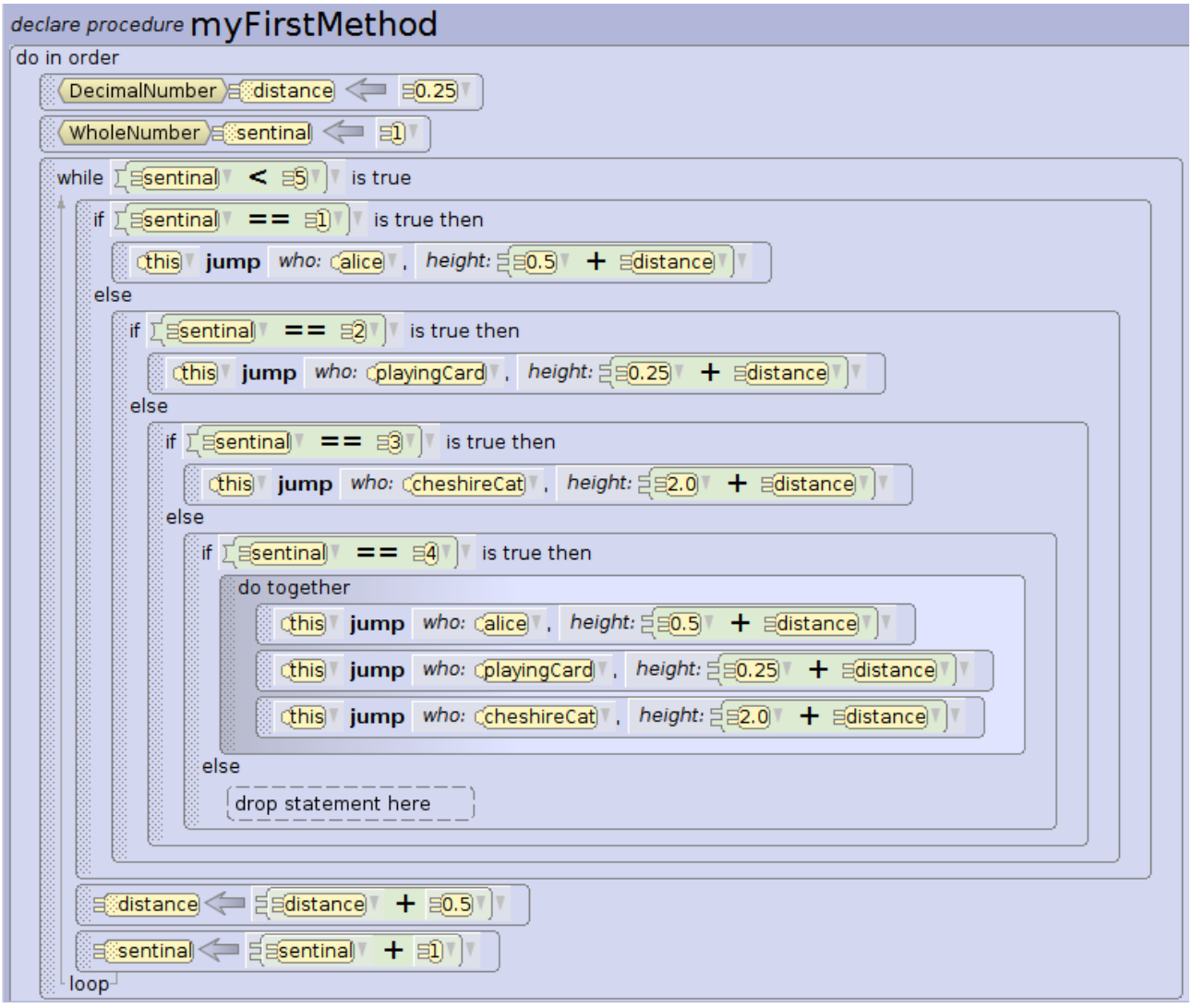
Iteration: Who jumps in loop



Selection: Who jumps in order in the loop

In the while loop,
Iteration 1: PlayingCard jumps (sentinal value ==1)
Iteration 2: Alice jumps (sentinal value==2)
Iteration 3: Cat jumps (sentinal value==3)
Iteration 4: All jump (sentinal value==4)
Iteration 5: While loop ends as “sentinal <5” condition becomes false.

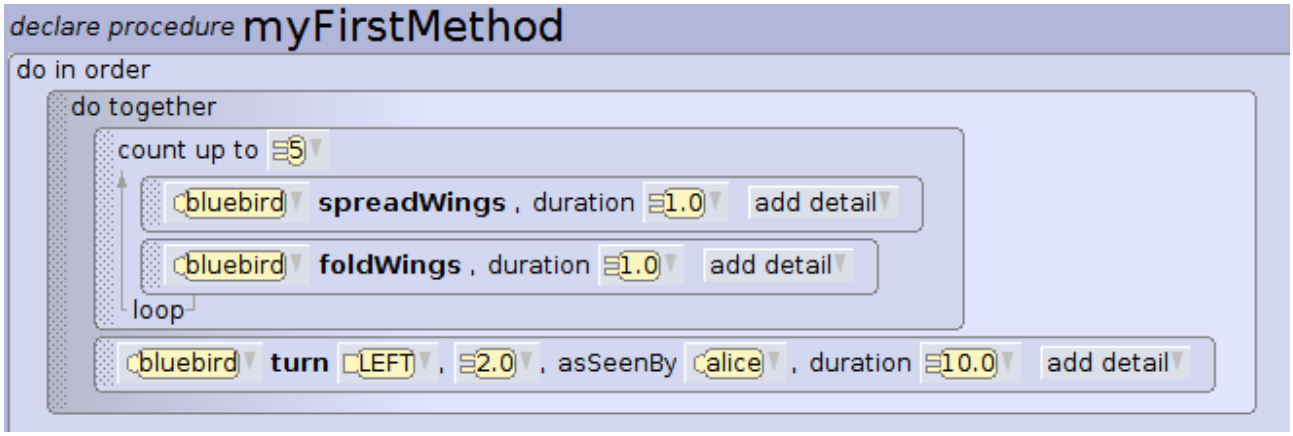
After each iteration, the distance is increased by 0.5



Iteration: Bird Circles Alice

Position a bird and alice as shown.

The bird flies around alice twice for 10 seconds. It also folds and spreads its wings five times while flying.



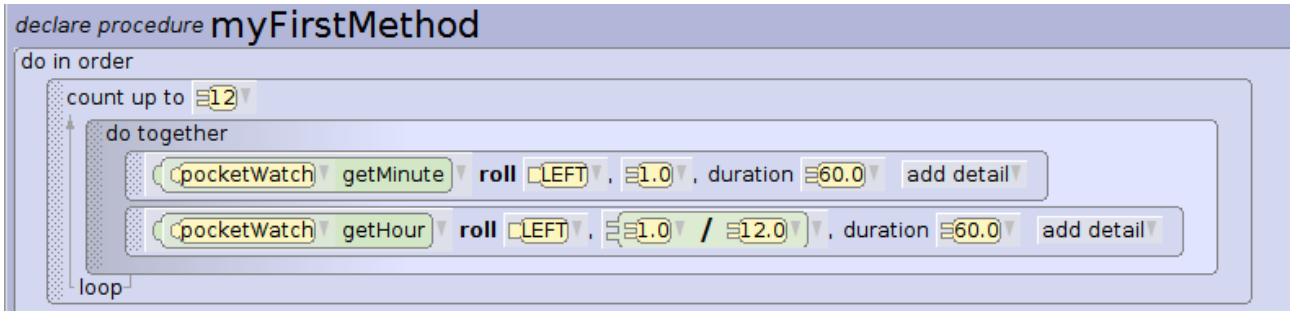
Note: The bird flies (turns) around the object passed as ‘asSeenBy’ parameter. Set the initial ‘Y’ coordinate for the bird as 1 m (above ground).

The duration of all statements inside do-together block shall be matched to be synched. In this case, count statement (5x(1+1)) and ‘turn’ have the same duration of 10 seconds.

Iteration: Clock Ticks

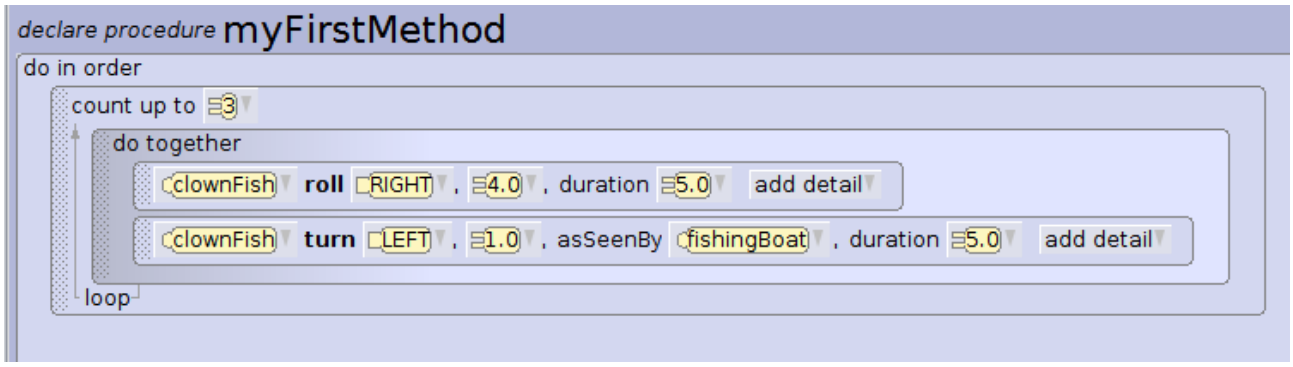
Setup a scene with a clock as follows.

Let us move the Minute needle to turn one complete rotation with the duration of 60 seconds. If you want to complete in actual 60 minutes, set the duration to 3600 seconds. Let the hour needle move 1/12 th of the rotation for every complete rotation of the minute needle. Repeat the code for 12 times, so that hour needle completes one full rotation (12 hours).



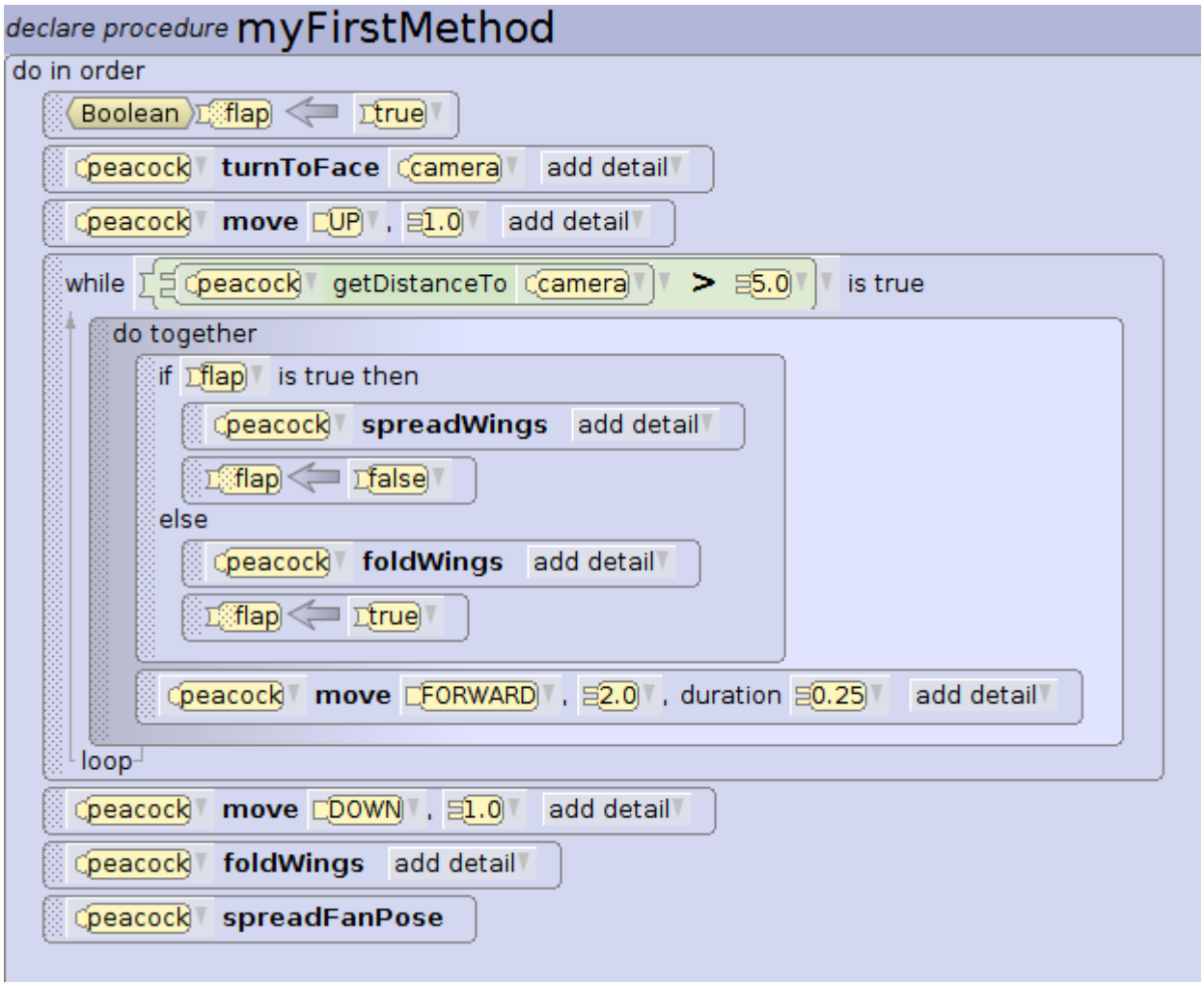
Iteration: Fish Circles Alice

Setup a scene as follows with fish, boat and alice on the sea surface ground.



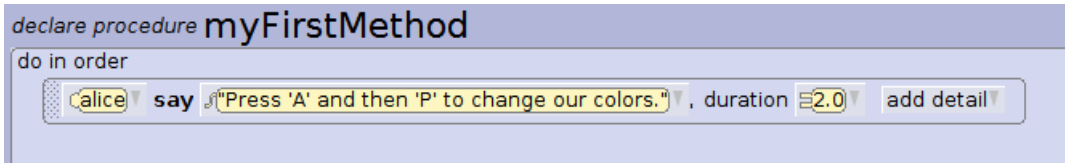
Selection: Peacock

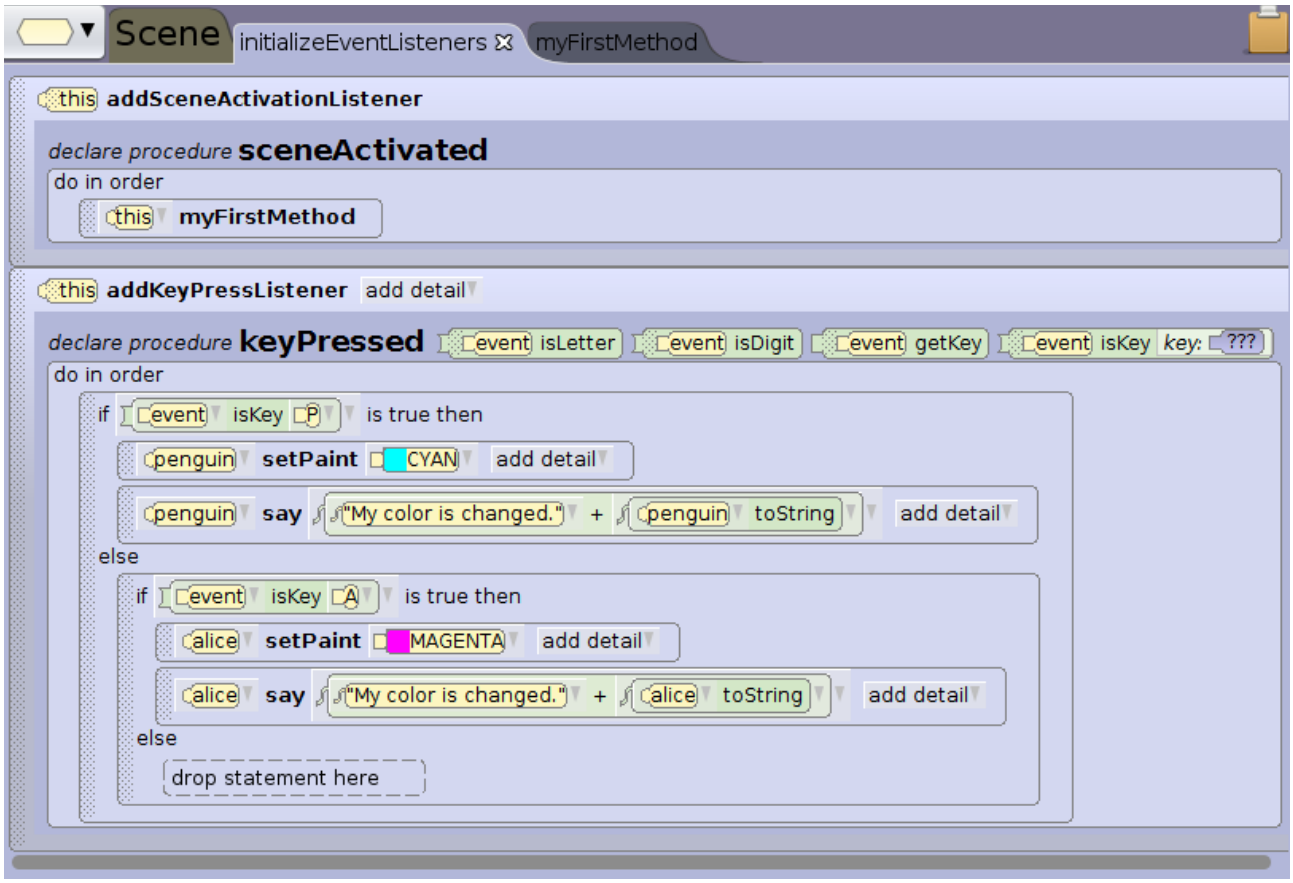
Place peacock on the grass ground.



Event Listener: Key Press

Setup a scene with alice and penguin as follows:





Event Listener: Object Mover (with Arrow Keys)

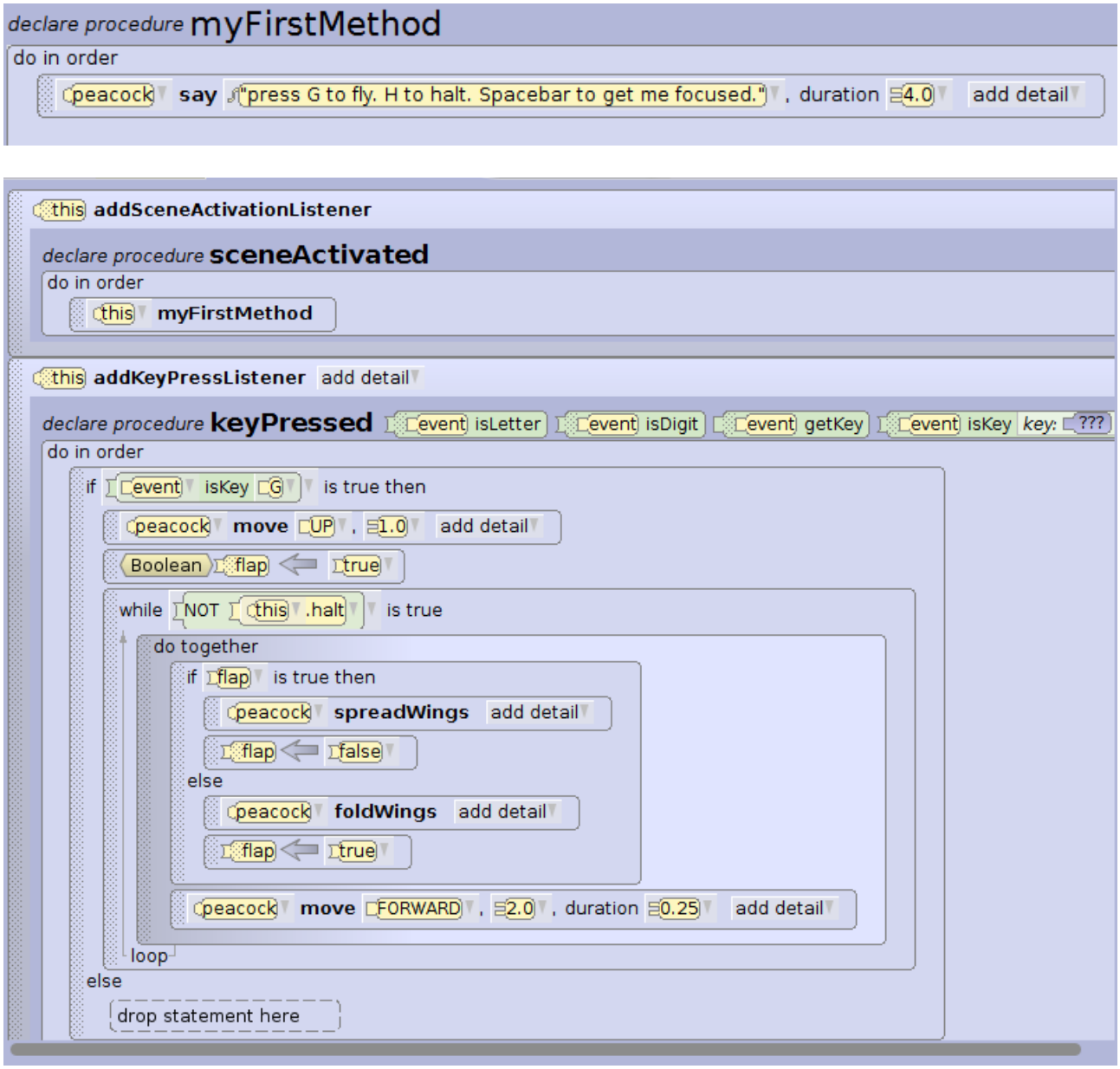
Place Alice in a ground of your choice.

You can now control Alice movement with arrow keys.

Event Listener: TimeElapsed

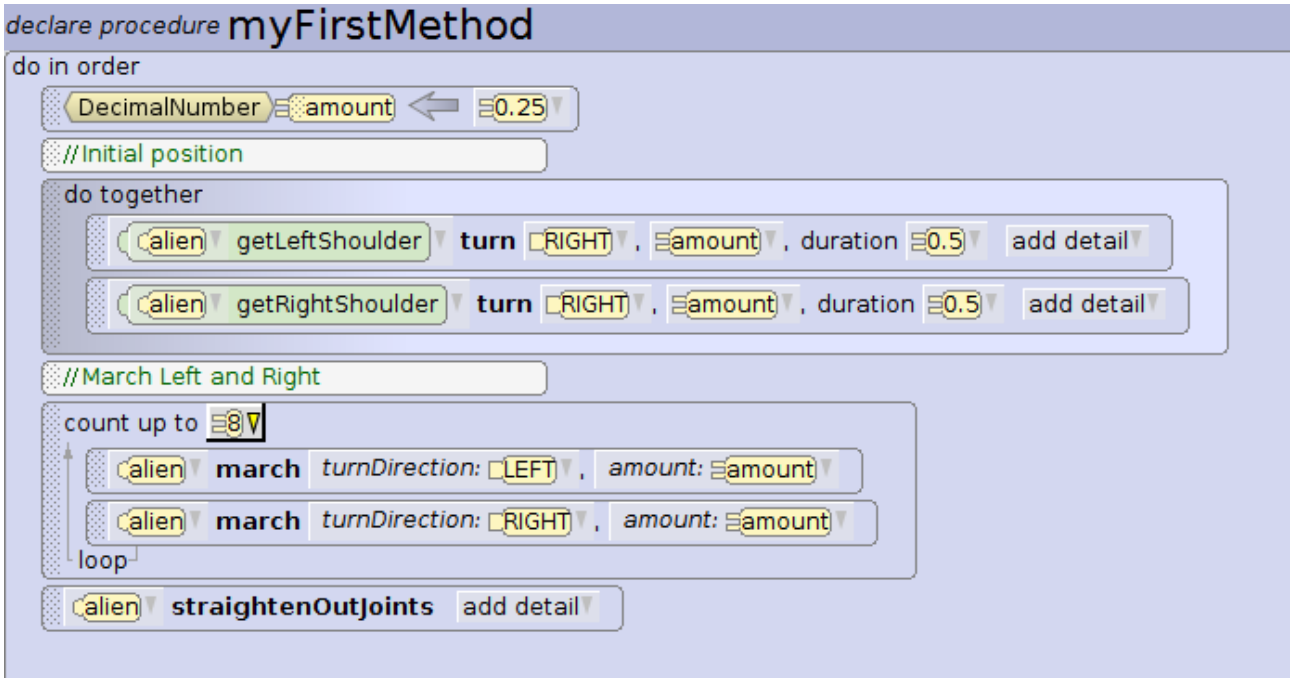
Alice will remind you every 10 seconds, when a timeListener is added as follows:

EventListener: Peacock Control



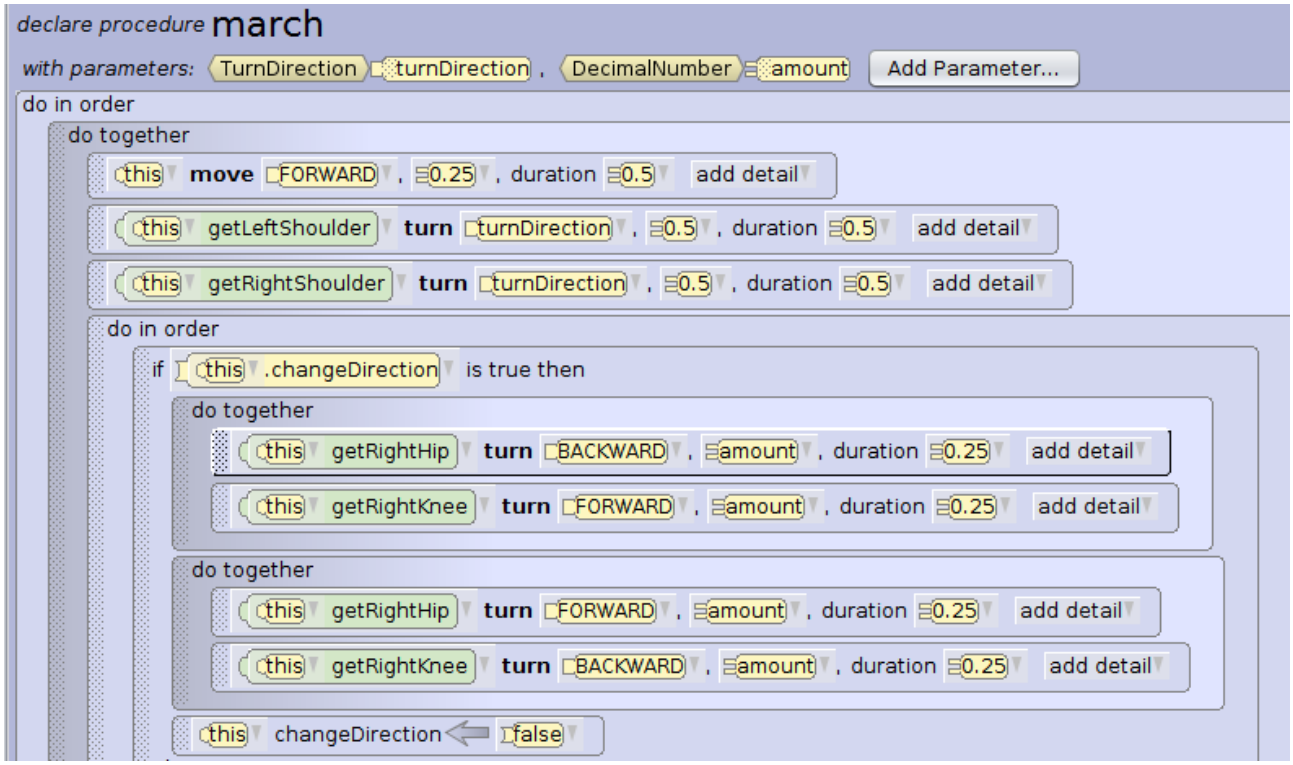
Iteration: AlienMarches

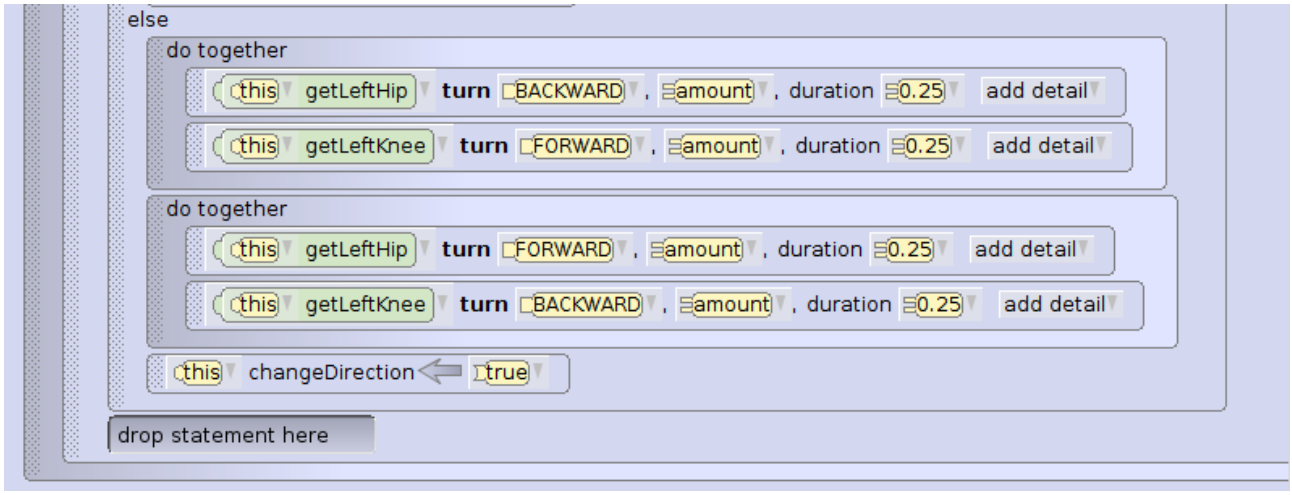
Complete Scene Setup with Mars ground and alien standing as shown.



Note:
StraightenoutJoints straightens all the joints to original position. Hands are taken to initial position, before start marching.
A **variable** named ‘amount’ is used in the firstMethod to pass the amount of legTurn required.

March procedure (defined in Bipad class)
Arguments : **turnDirection** of type TurnDirection and
amount of type DecimalNumber





Note:
First three statements (move, leftShoulder turn and rightShoulder turn) are executed together in the same duration 0.25 seconds.
Fourth statement is conditional, which uses the **user defined property** of bipad class known as changeDirection of type Boolean with default value as true.
If changeDirection is true, then march with right leg.
Else, march with left leg.

Toggle changeDirection for each call of march procedure.

Alice rides the boat

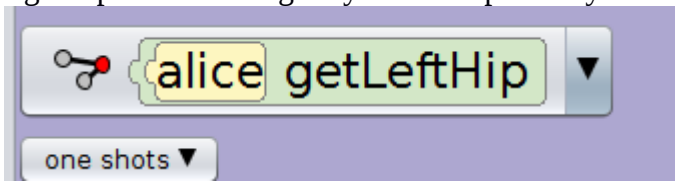
Place alice, island and the boat on the seasurface ground, as shown.


```
declare procedure myFirstMethod
do in order
    alice place ABOVE , woodenBoat add detail
    alice setVehicle woodenBoat
    camera setVehicle woodenBoat
    woodenBoat moveAwayFrom island , 10.0 add detail
    camera moveAwayFrom woodenBoat , 5.0 add detail
    woodenBoat turn LEFT , 4.0 , asSeenBy island , duration 10.0 , animationStyle BEGIN_AND_END_GENTLY
```

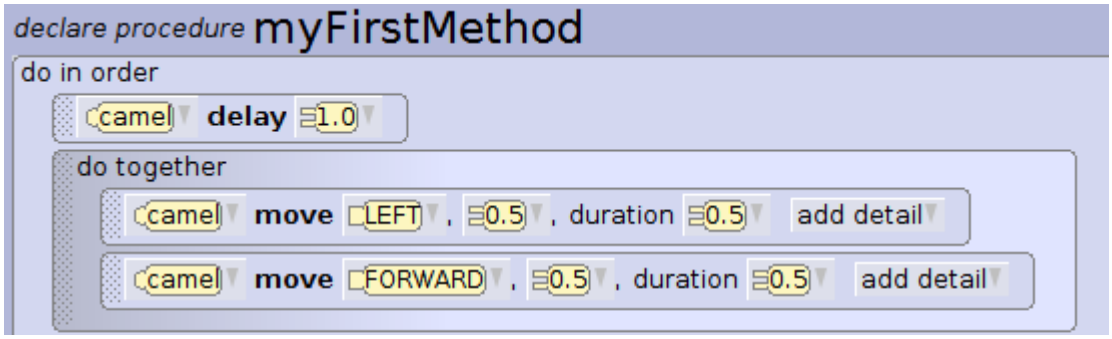
Animation style controls how the boat should start and stop. “Begin and End Gently” gives realistic inertia resisted start and stop movement.
“SetVehicle” attaches alice to the boat. Then alice moves whenever boat moves.
“Camera” object can be controlled interactively in the code to visualize different shots in the scene.

Alice rides the camel

Place camel and alice on the desert ground as shown. Use one shot procedures for turing leftHip and rightHip to left and right by 0.125 respectively.

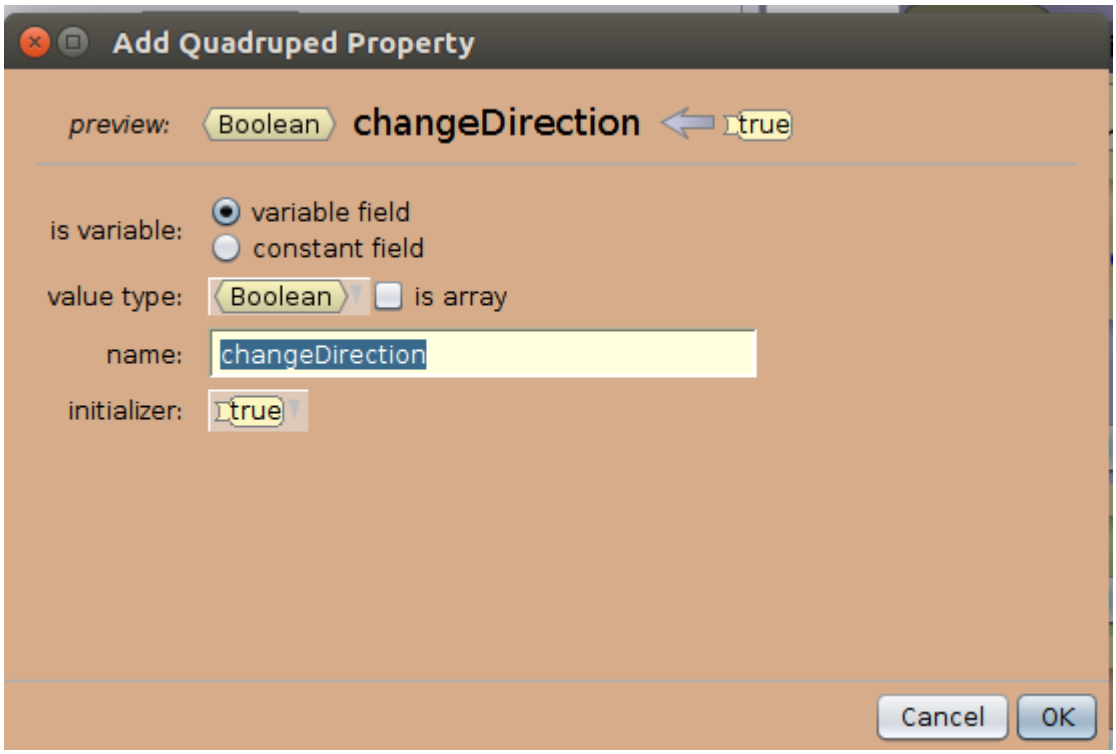


Set the vehicle for alice as camel, afer rightly positioning her above camel.

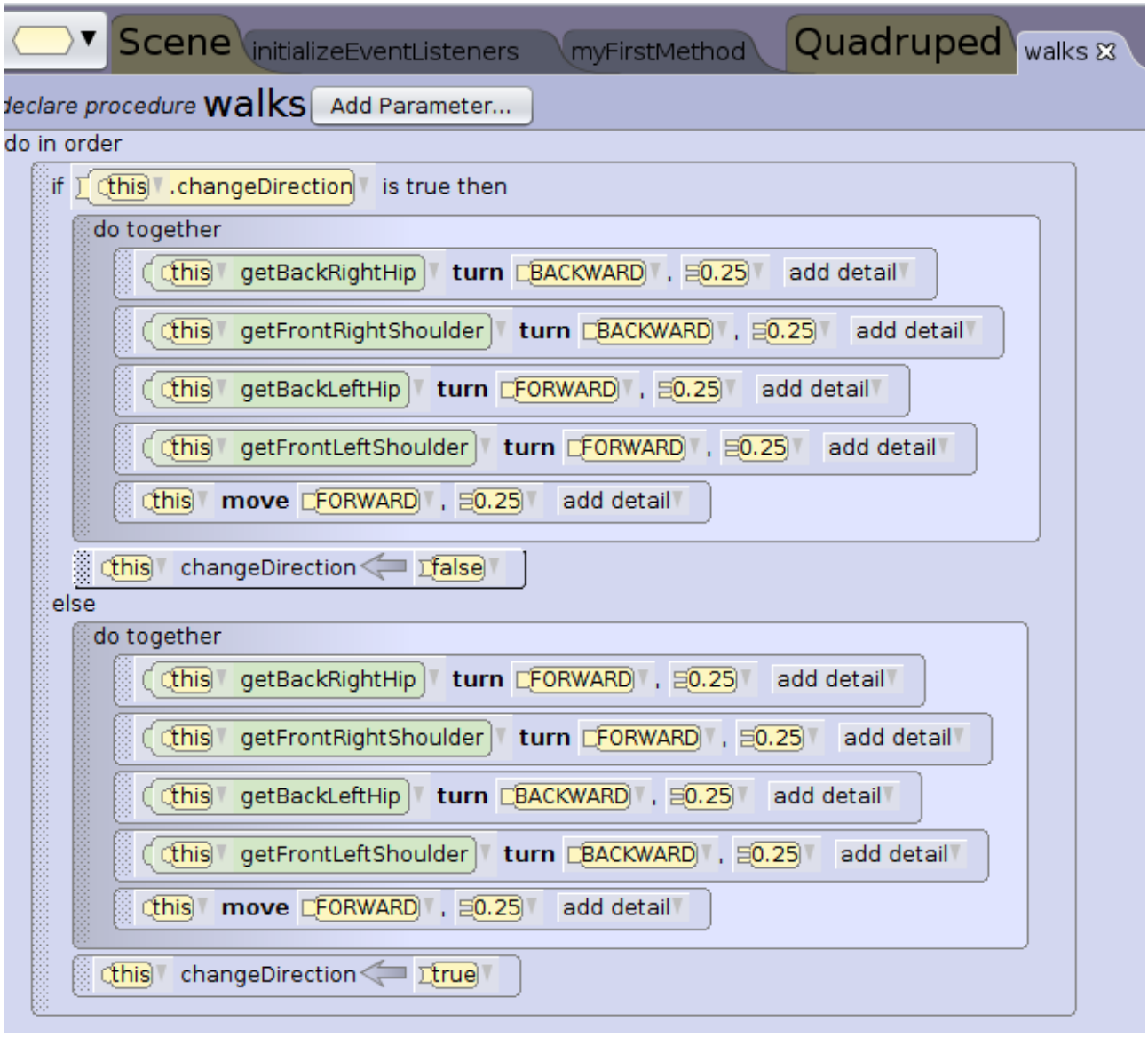


The initial delay is to produce the smooth animation rendering without missing any actions or shots during the initial loading. The duration of 0.5 seconds synchronizes both left and forward movement of camel when done together.

Create a property of boolean type in Quadruped class to change direction of camel legs during each walk step. Set its initial value as true.



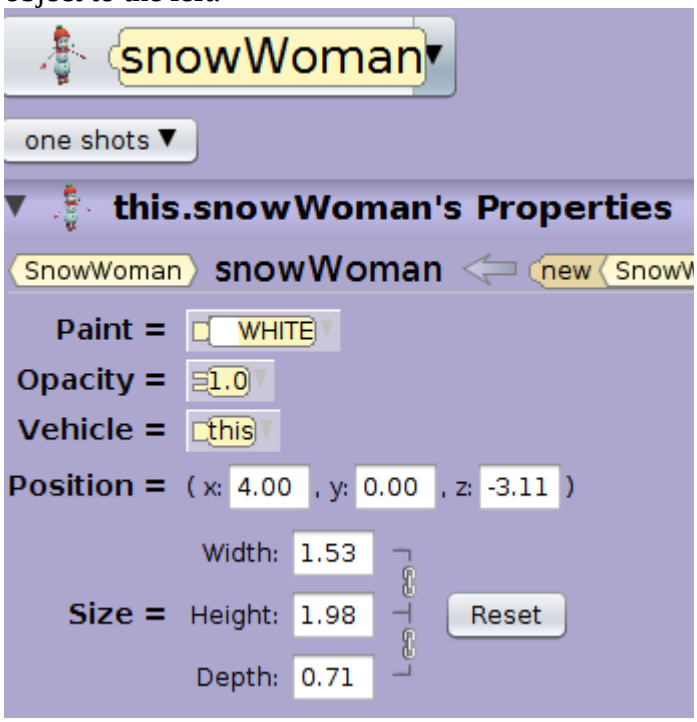
Create an user defined procedure ‘walk’ in the Quadruped class as shown.

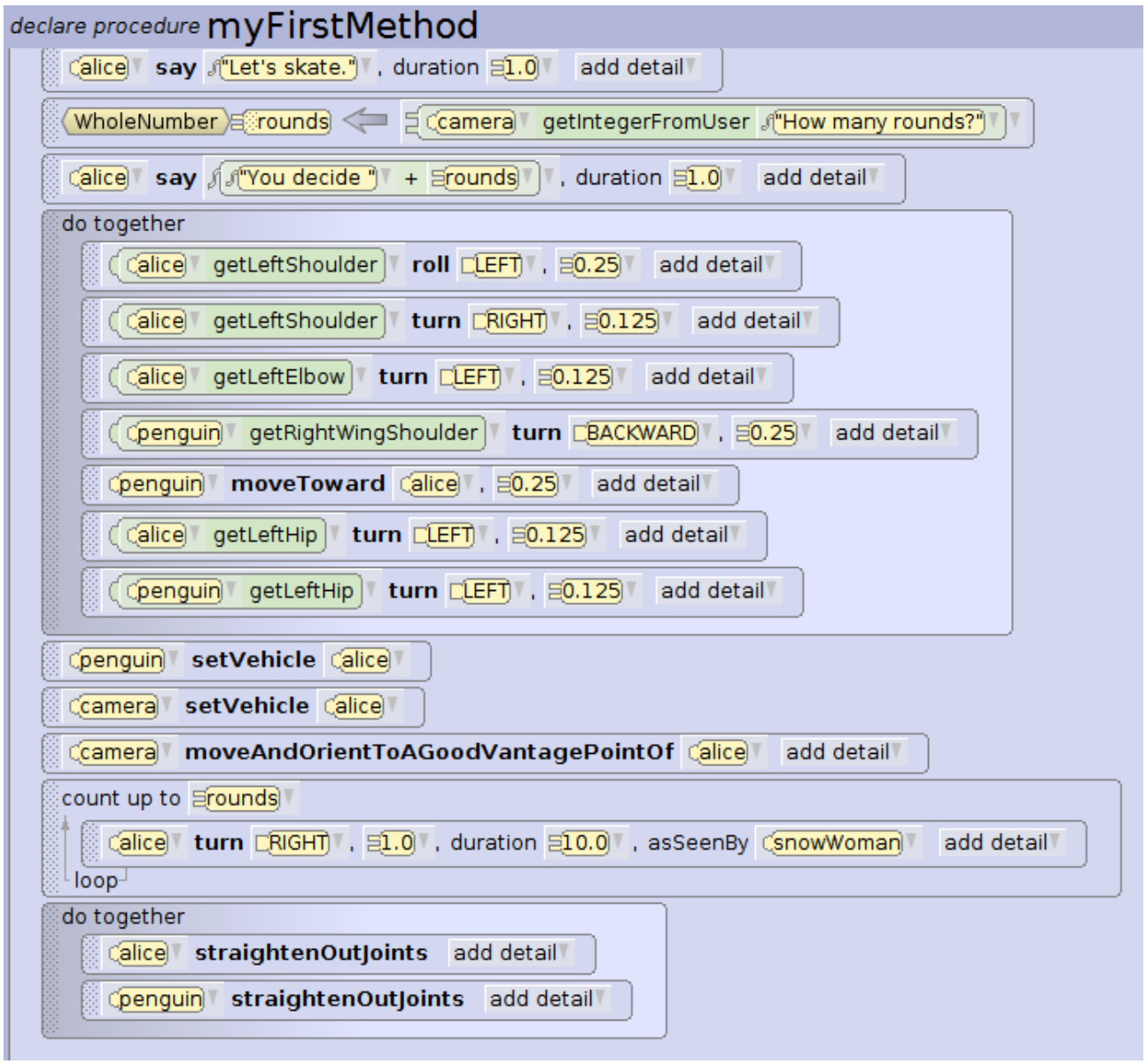

Scene
initializeEventListeners
myFirstMethod
Quadruped
walks
declare procedure walks
Add Parameter...
do in order
if this .changeDirection is true then
do together
this getBackRightHip turn BACKWARD , 0.25 add detail
this getFrontRightShoulder turn BACKWARD , 0.25 add detail
this getBackLeftHip turn FORWARD , 0.25 add detail
this getFrontLeftShoulder turn FORWARD , 0.25 add detail
this move FORWARD , 0.25 add detail
this changeDirection false
else
do together
this getBackRightHip turn FORWARD , 0.25 add detail
this getFrontRightShoulder turn FORWARD , 0.25 add detail
this getBackLeftHip turn BACKWARD , 0.25 add detail
this getFrontLeftShoulder turn BACKWARD , 0.25 add detail
this move FORWARD , 0.25 add detail
this changeDirection true

Alice skates

Setup the scene with alice, penguin and snowWoman(not in the camera view) on the snow ground.

Place Snow woman out of camera view using x coordinate setting as 4. Increasing x shifts the object to the left.




declare procedure myFirstMethod
alice say "Let's skate." , duration 1.0 add detail
WholeNumber rounds camera getIntegerFromUser "How many rounds?"
alice say "You decide " + rounds , duration 1.0 add detail
do together
alice getLeftShoulder roll LEFT , 0.25 add detail
alice getLeftShoulder turn RIGHT , 0.125 add detail
alice getLeftElbow turn LEFT , 0.125 add detail
penguin getRightWingShoulder turn BACKWARD , 0.25 add detail
penguin moveToward alice , 0.25 add detail
alice getLeftHip turn LEFT , 0.125 add detail
penguin getLeftHip turn LEFT , 0.125 add detail
penguin setVehicle alice
camera setVehicle alice
camera moveAndOrientToAGoodVantagePointOf alice add detail
count up to rounds
alice turn RIGHT , 1.0 , duration 10.0 , asSeenBy snowWoman add detail
loop
do together
alice straightenOutJoints add detail
penguin straightenOutJoints add detail

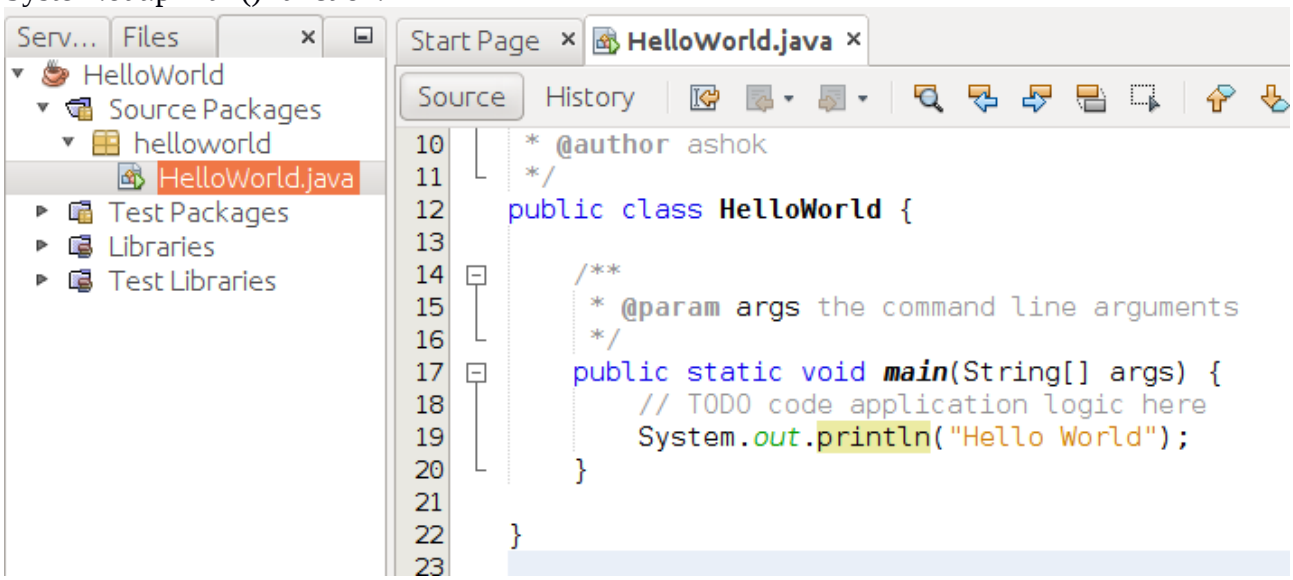
Array: FishGang

```
declare procedure myFirstMethod
do in order
    Fish[] fishGang ⟸ new Fish[] { clownFish, blueTang, carp, pajamaFish, piranha }
    for each Fish f in fishGang
        do together
            f move UP , 2.0 , duration 2.0 add detail
            f move FORWARD , 2.0 , duration 2.0 add detail
        do together
            f move DOWN , 2.0 , duration 2.0 add detail
            f move FORWARD , 2.0 , duration 2.0 add detail
    loop
    each Fish item in fishGang together
        do together
            item move UP , 2.0 , duration 2.0 add detail
            item move FORWARD , 2.0 , duration 2.0 add detail
        do together
            item move DOWN , 2.0 , duration 2.0 add detail
            item move FORWARD , 2.0 , duration 2.0 add detail
```

Java Basics

HelloWorld

Use NetBeans to create a new project “HelloWorld”. To see the output in console, use System.out.println() function.

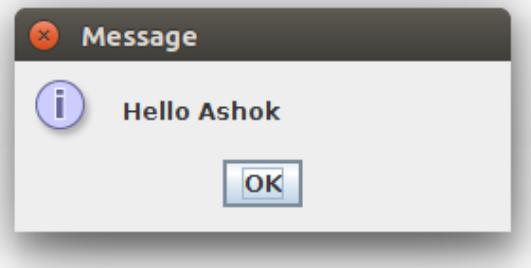


To generate a dialog with the message, use JoptionPane from swing.

```
package helloworld;
import javax.swing.JOptionPane;
/**
 *
 * @author ashok
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        //System.out.println("Hello World");
        JOptionPane.showMessageDialog(null, "Hello World");
    }
```

To get user input using dialog, use JoptionPane.showInputDialog.

```
*/
package helloworld;
import javax.swing.JOptionPane;
/**
 *
 * @author ashok
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String name=JOptionPane.showInputDialog("What's your name?");
        JOptionPane.showMessageDialog(null, "Hello "+name);
    }

}
```



Class: Person

Add a class file (New->Class) which creates Person.java in the project.

```java
/*
 * To change this license header, choose Li
 * To change this template file, choose Too
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author ashok
 */
public class Person {
    String name;
    int age;

    Person(String a, int b)
    {
        name=a;
        age=b;
    }

    public void setAge(int a){
        age=a;
    }

    public int getAge(){
        return age;
    }
}
```

Use the class to create its instance (object) p1 in the main program.

```
package helloworld;
import javax.swing.JOptionPane;
/**
 *
 * @author ashok
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String name=JOptionPane.showInputDialog("What's your name?");
        JOptionPane.showMessageDialog(null, "Hello "+name);
        Person p1=new Person(name, 38);
        JOptionPane.showMessageDialog(null, "Your age is "+p1.getAge());
    }

}
```

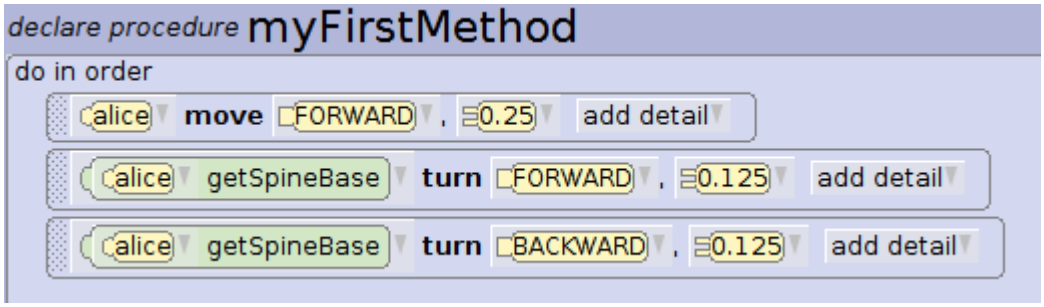
Exercise:
Get the age as the input from the user (JoptionPane) and set the age of the person with the user input.

Calculate Dog Age

```
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String name=JOptionPane.showInputDialog("What's your name?");
        JOptionPane.showMessageDialog(null, "Hello "+name);
        Person p1=new Person(name, 38);
        int dogAge=0;
        JOptionPane.showMessageDialog(null, "Your age is "+p1.getAge());

        /*  A one year old dog roughly corresponds to a fourteen year old
         *  A dog who is two years old corresponds to a 22 year old human
         *  Every further dog year corresponds to five human years
        */

        if(p1.getAge()<= 14)
            dogAge=1;
        else if (p1.getAge()<= 22)
            dogAge=2;
        else
            dogAge=(p1.getAge()-22)/5 + 2;
        JOptionPane.showMessageDialog(null,"Your dog Age is "+dogAge);


    }
```

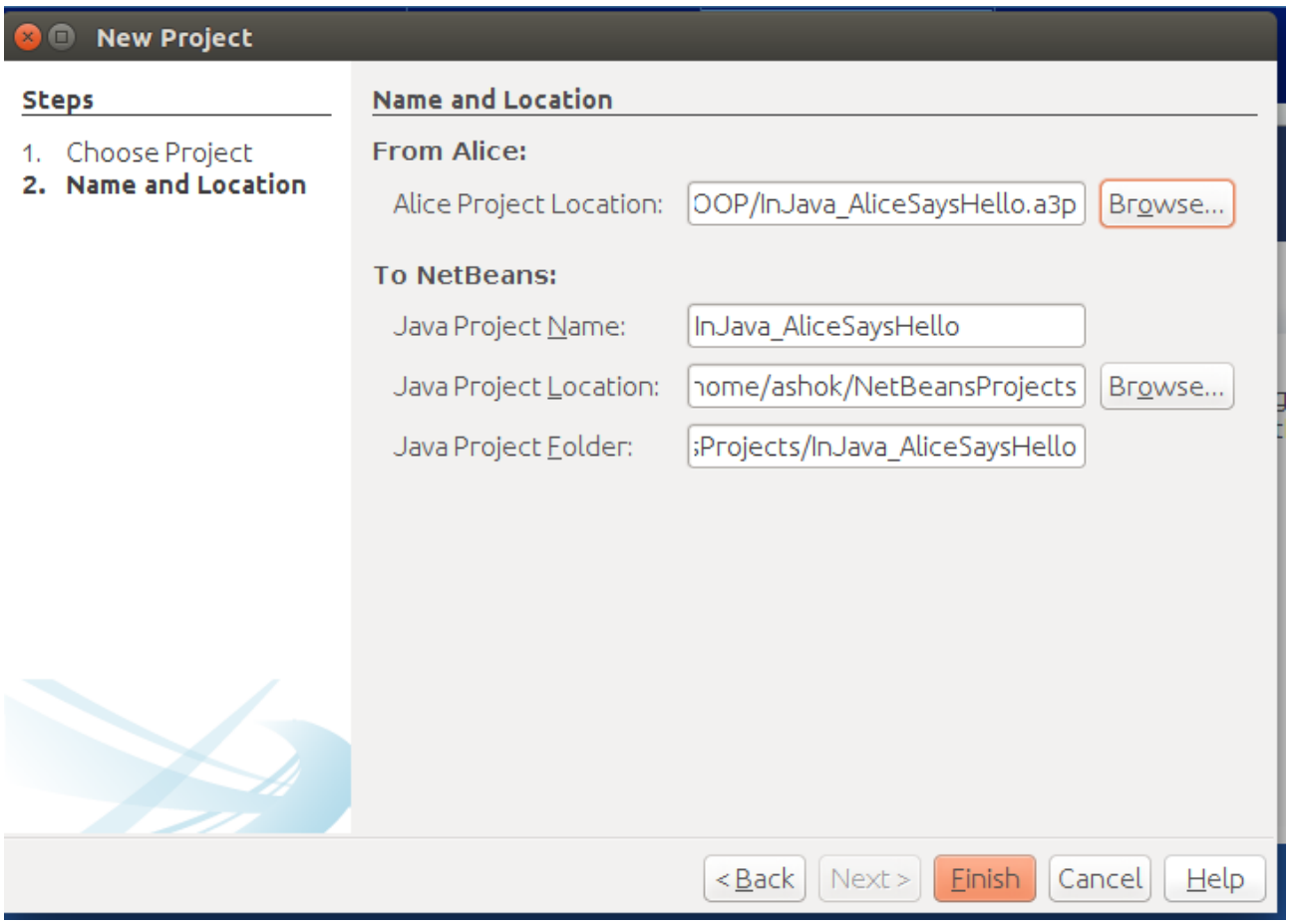
Alice 3 to Java

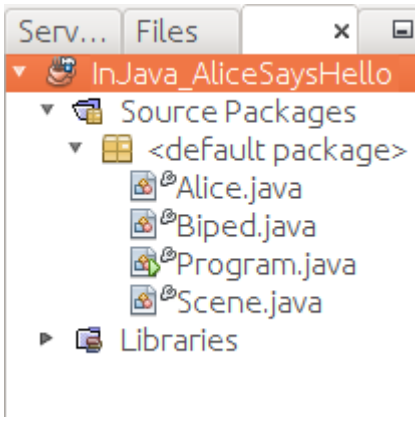
Setup a scene with Alice as follows.



Save the project as “InJava_AliceSaysHello”. It is saved with the extension .a3p.

Close Alice3. Open NetBeans. If Alice3 plugin is not installed, please install before proceeding.


New Project
Steps
1. Choose Project
2. Name and Location
Name and Location
From Alice:
Alice Project Location:
OOP/InJava_AliceSaysHello.a3p
Browse...
To NetBeans:
Java Project Name:
InJava_AliceSaysHello
Java Project Location:
ıome/ashok/NetBeansProjects
Browse...
Java Project Folder:
;Projects/InJava_AliceSaysHello
< Back
Next >
Finish
Cancel
Help


Serv...
Files
InJava_AliceSaysHello
Source Packages
<default package>
Alice.java
Biped.java
Program.java
Scene.java
Libraries

```
imports

class Scene extends SScene {

    /* Construct new Scene */
    public Scene() {
        super();
    }

    /* Event listeners */
    private void initializeEventListeners() {
        this.addSceneActivationListener((SceneActivationEvent event) -> {
            this.myFirstMethod();
        });
    }

    /* Procedures and functions for this scene */
    public void myFirstMethod() {
        this.alice.move(MoveDirection.FORWARD, 0.25);
        this.alice.getSpineBase().turn(TurnDirection.FORWARD, 0.125);
        this.alice.getSpineBase().turn(TurnDirection.BACKWARD, 0.125);
    }
    /* End procedures and functions for this scene */

    /* Scene fields  */

    /* Scene setup  */

    /* Procedures and functions to handle multiple scenes */
}
```

Add the following line.

this.alice.say(“Hello World”);

```
    /* Procedures and functions for this scene */
    public void myFirstMethod() {
        this.alice.move(MoveDirection.FORWARD, 0.25);
        this.alice.getSpineBase().turn(TurnDirection.FORWARD, 0.125);
        this.alice.getSpineBase().turn(TurnDirection.BACKWARD, 0.125);
        this.alice.say("Hello World");
    }
```

Run and see the output.

Constructor

In Alice.java

```
class Alice extends Biped {

    int age;

    public void setAge(int a){
        age=a;
    }

    public int getAge(){
        return age;
    }
```

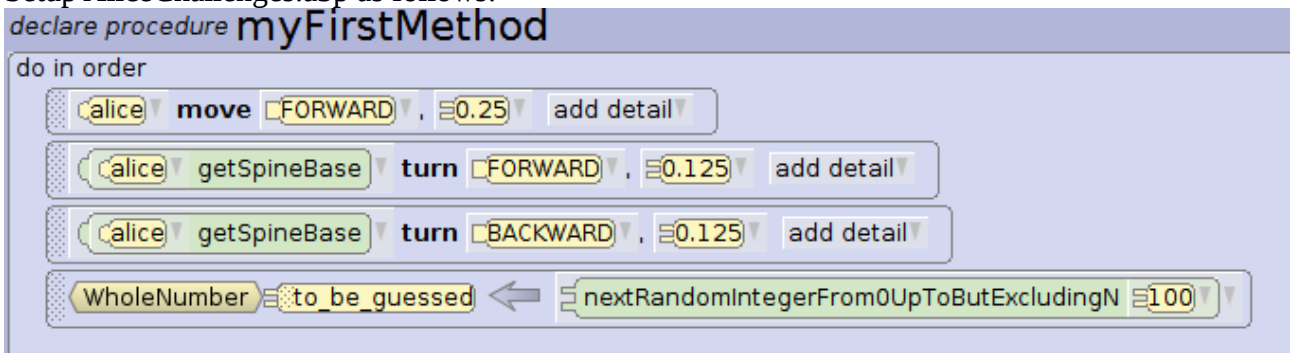
In Scene.java

```
    /* Procedures and functions for this scene */
    public void myFirstMethod() {
        this.alice.move(MoveDirection.FORWARD, 0.25);
        this.alice.getSpineBase().turn(TurnDirection.FORWARD, 0.125);
        this.alice.getSpineBase().turn(TurnDirection.BACKWARD, 0.125);
        this.alice.say("Hello World");
        this.alice.setAge(38);
        JOptionPane.showMessageDialog(null, "Alice age is "+this.alice.getAge());
    }
```

Output:

AliceChallenges

Setup AliceChallenges.a3p as follows:



Create a new java project using “AliceChallenges.a3p”.
In Scene.java,

```
    /* Procedures and functions for this scene */
    public void myFirstMethod() {
        this.alice.move(MoveDirection.FORWARD, 0.25);
        this.alice.getSpineBase().turn(TurnDirection.FORWARD, 0.125);
        this.alice.getSpineBase().turn(TurnDirection.BACKWARD, 0.125);
        Integer to_be_guessed = RandomUtilities.nextIntegerFrom0ToNExclusive(100);
        int guess = 0;
        while(guess != to_be_guessed){
            guess=this.alice.getIntegerFromUser("Guess my number (-9 to quit)");
            if (guess == -9)
            {
                this.alice.say("Sorry that you're giving up!");
                break;
            }
            else if (guess > to_be_guessed)
                this.alice.say("Number too large");
            else if (guess < to_be_guessed)
                this.alice.say("Number too small");
            else
                this.alice.say("Congratulation. You found it!");
        }
    }
```

Right Angle Triangles

For the right angle triangle, the square of hypotenuse is equal to the sum of the squares of the other two sides. The program finds the possible right angle triangles within the upper limit N.

In Scene.java, import sqrt() as static to be used without instance:

```
imports
import static java.lang.Math.sqrt;

class Scene extends SScene {
```

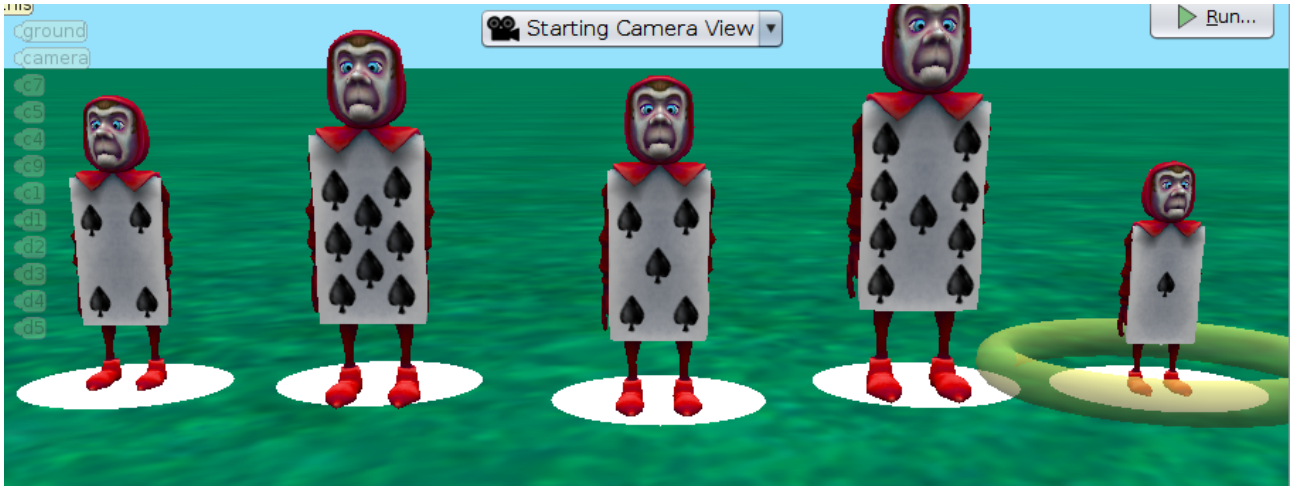
Edit myFirstMethod:

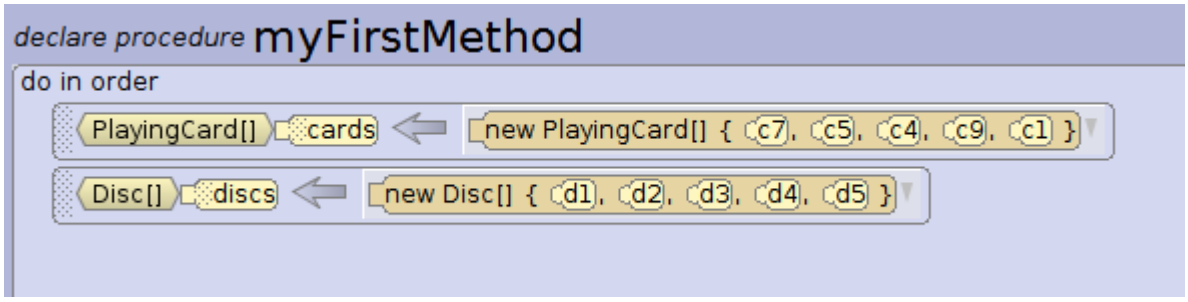
```
    /* Procedures and functions for this scene */
    public void myFirstMethod() {
        this.alice.move(MoveDirection.FORWARD, 0.25);
        this.alice.getSpineBase().turn(TurnDirection.FORWARD, 0.125);
        this.alice.getSpineBase().turn(TurnDirection.BACKWARD, 0.125);
        int N=this.alice.getIntegerFromUser("Enter Upper Limit N");
        int c;
        int c_square;
        String sides;
        this.alice.say("The possible right angle triangles are");
        for(int i=1;i<=N;i++){
            for(int j=i;j<N;j++){
                c_square=i*i+j*j;
                c=(int)sqrt(c_square);
                if(c*c==c_square){
                    sides=i+" , "+j+" and "+c;
                    this.alice.say("Triangle with sides "+sides+
                    " is the right angle triangle" );
                }
            }

        }
    }
```

Exercise:
Find the sum upto N.

BubbleSort

Setup a scene with playingCards as follows. Change “Height” property of each card to differ each other.





Save the file and close. Open Netbeans. Create new project using Alice plugin. Select the BubbleSortSetup.a3p file.

In Scene.java,

```
    /* Procedures and functions for this scene */
    public void myFirstMethod() {
        PlayingCard[] card = new PlayingCard[]{this.c7, this.c5, this.c4, this.c9, this.c1};
        Disc[] disc = new Disc[]{this.d1, this.d2, this.d3, this.d4, this.d5};

        PlayingCard temp;
        boolean swap=true;

        while(swap==true){
            swap=false;
            for(int i=0;i<card.length-1;i++){
                if(card[i].getHeight()>card[i+1].getHeight()){
                    card[i].moveTo(disc[i+1]);
                    card[i+1].moveTo(disc[i]);
                    temp=card[i+1];
                    card[i+1]=card[i];
                    card[i]=temp;
                    swap=true;
                }

            }

        }

    }
```