

Making Your Own iHart Application

Requirements:

- iHart installed on computer (Windows or Mac)
- Adobe Flash CS6
- webcam
- speakers (for iHart sounds)
- decent-sized display(for iHart display)
- projector aimed at wall or floor (for iHart projection)

Setup:

Open Adobe Flash CS6

Create a **new .fla** file by clicking on ActionScript 3.0 under “Create New”

Create a **new .as** file by clicking File > New > ActionScript 3.0 Class (under Type)

Name your .as file whatever you want your application to be called and **name** your .fla file with the same name

For our demo we have named the files *DemoApplication.fla* and *DemoApplication.as*

Setup for .as file:

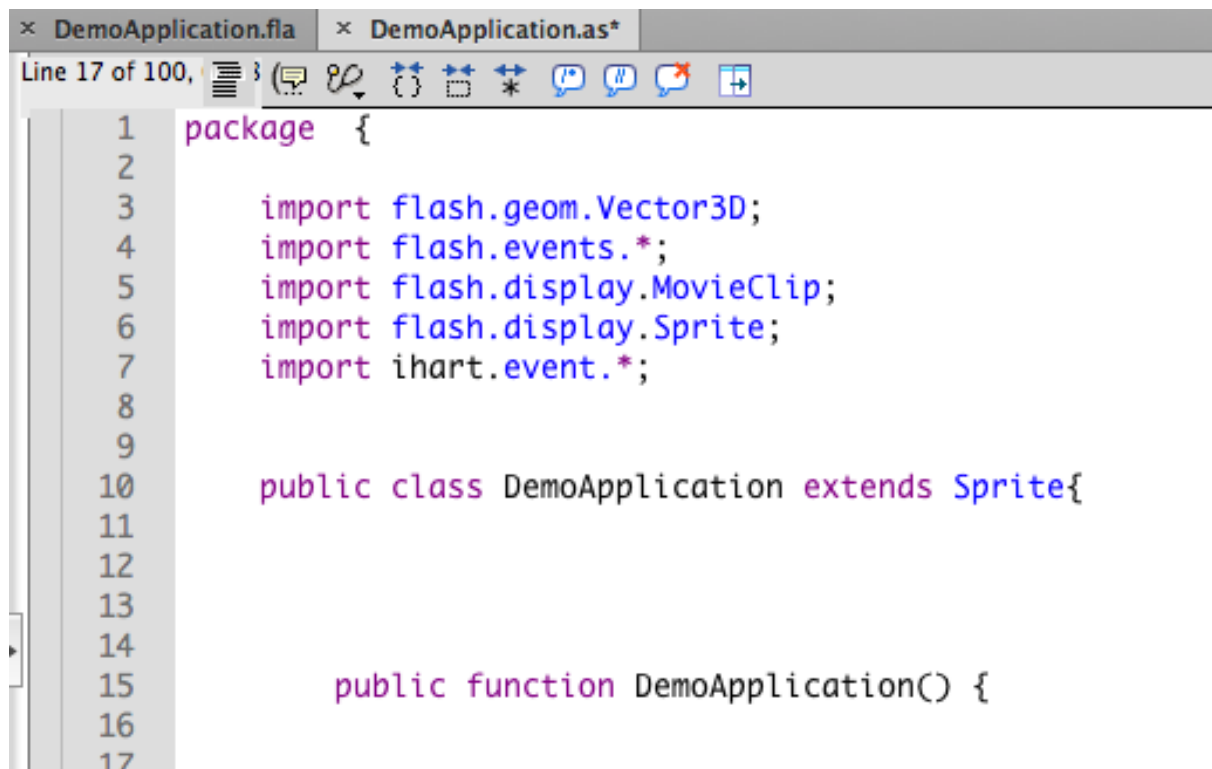
Now go to your .as file in Flash, you will see that Flash has already created some things for you. In order for stuff to work you will need to **add some stuff** to your code yourself:

Importing Libraries:

-copy+paste these import statements

```
import flash.geom.Vector3D;  
import flash.events.*;  
import flash.display.MovieClip;  
import flash.display.Sprite;  
import ihart.event.*;
```

-place them in your code after “package” like so:



```
1 package {
2
3     import flash.geom.Vector3D;
4     import flash.events.*;
5     import flash.display.MovieClip;
6     import flash.display.Sprite;
7     import iHart.event.*;
8
9
10    public class DemoApplication extends Sprite{
11
12
13
14
15        public function DemoApplication() {
16
17
```

*Note: throughout this demo wherever you see “DemoApplication” will be whatever name you gave your .as and .fla files

Next add “extends Sprite” after the class declaration “DemoApplication” as seen above

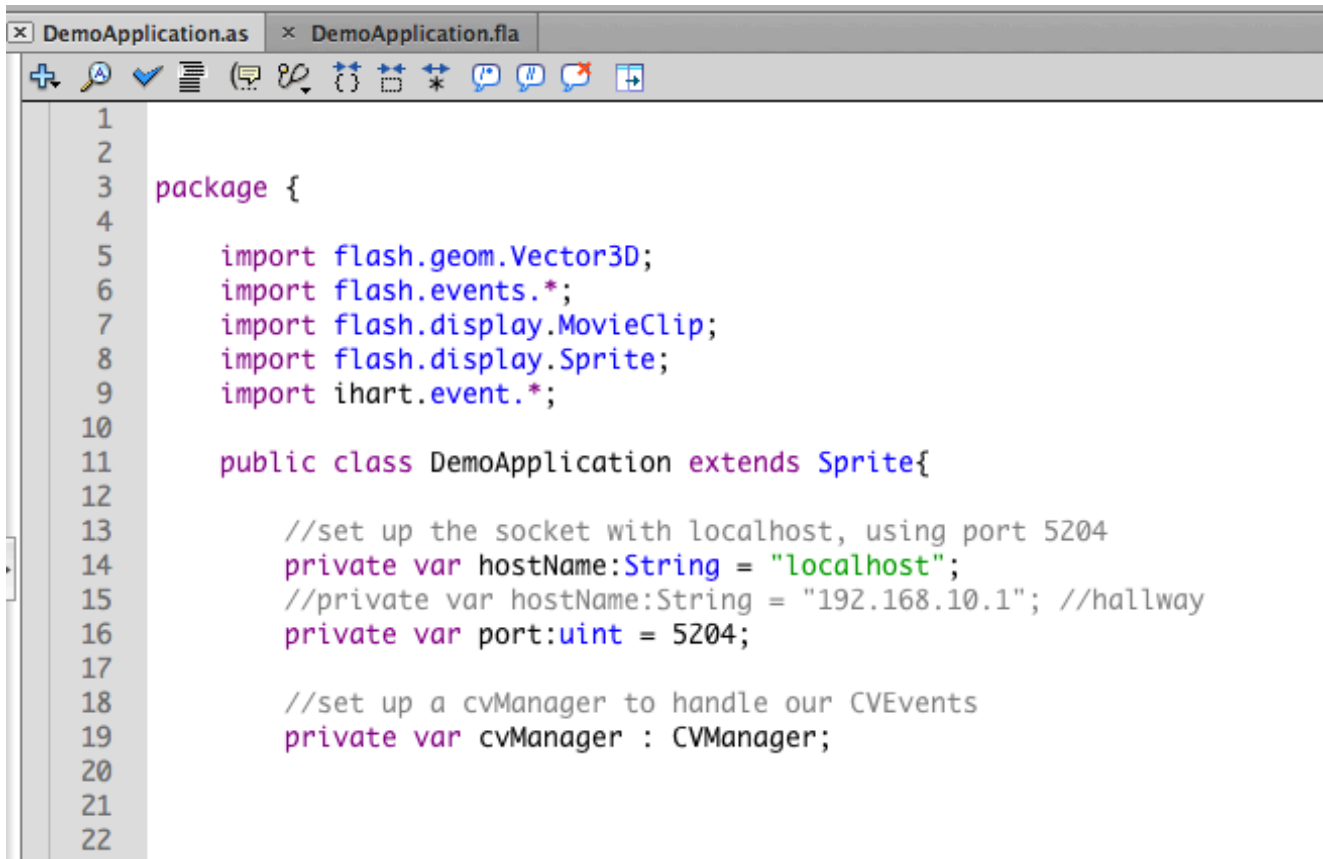
Defining global variables and constructor:

Next we will be adding certain “terms” or variables that need to be defined for all iHart applications for the code to work

Copy+paste the global variables below

```
private var hostName:String = "localhost"; //the host name will generally be localhost
private var port:uint = 5204; // the port number is 5204
private var cvManager : CVManager; //cvManager will handle our CVEvents
```

Add them to your code after “public class DemoApplication { ” like so



```
1
2
3 package {
4
5     import flash.geom.Vector3D;
6     import flash.events.*;
7     import flash.display.MovieClip;
8     import flash.display.Sprite;
9     import iHart.event.*;
10
11     public class DemoApplication extends Sprite{
12
13         //set up the socket with localhost, using port 5204
14         private var hostName:String = "localhost";
15         //private var hostName:String = "192.168.10.1"; //hallway
16         private var port:uint = 5204;
17
18         //set up a cvManager to handle our CVEvents
19         private var cvManager : CVManager;
```

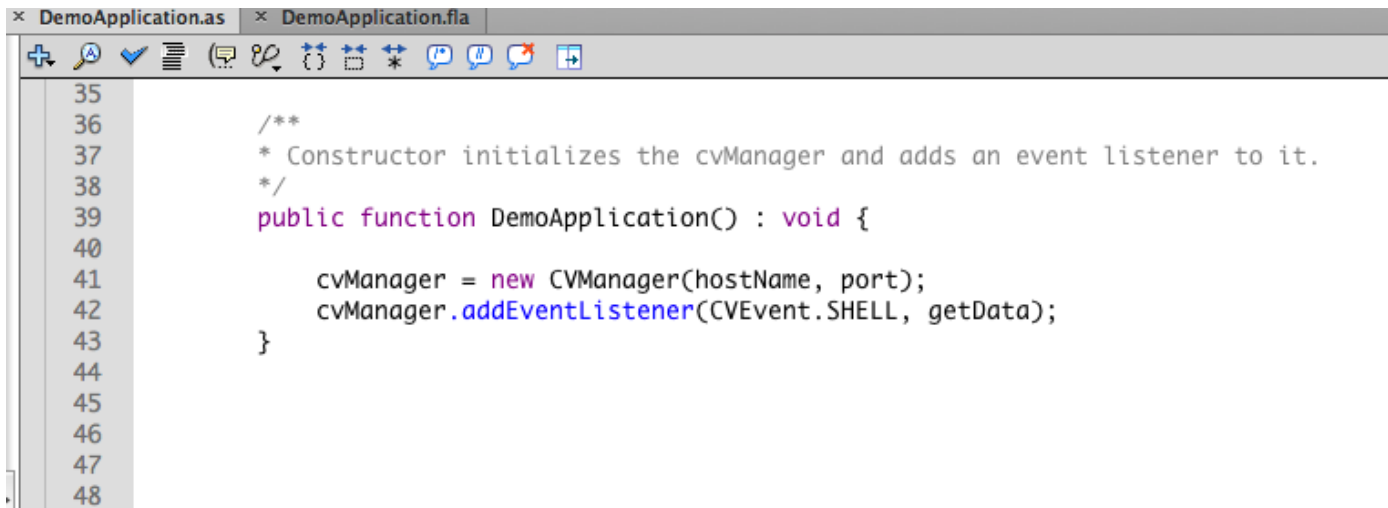
*Note: throughout this demo wherever you see “DemoApplication” will be whatever name you gave your .as and .fla files

Copy+paste these lines of code to add to your constructor

```
cvManager = new CVManager(hostName, port);
cvManager.addEventListener(CVEvent.SHELL, getData);
```

This will initialize your cvManager and add an event listener to it (all necessary for iHart to work!)

Add them to your code in your after “public function DemoApplication() { “ (your constructor) like so



```
35
36
37  /**
38   * Constructor initializes the cvManager and adds an event listener to it.
39   */
39  public function DemoApplication() : void {
40
41      cvManager = new CVManager(hostName, port);
42      cvManager.addEventListener(CVEvent.SHELL, getData);
43  }
44
45
46
47
48
```

Now notice that we will be defining the function `getData` to determine what happens with user movements. In other words this function will save the data of users movements and react to them however you would like.

So you need to **declare** the **getData function** like this

```
public function getData (e : CVEvent) : void { }
```

Add it after the “public function DemoApplication()” last bracket “ } ”

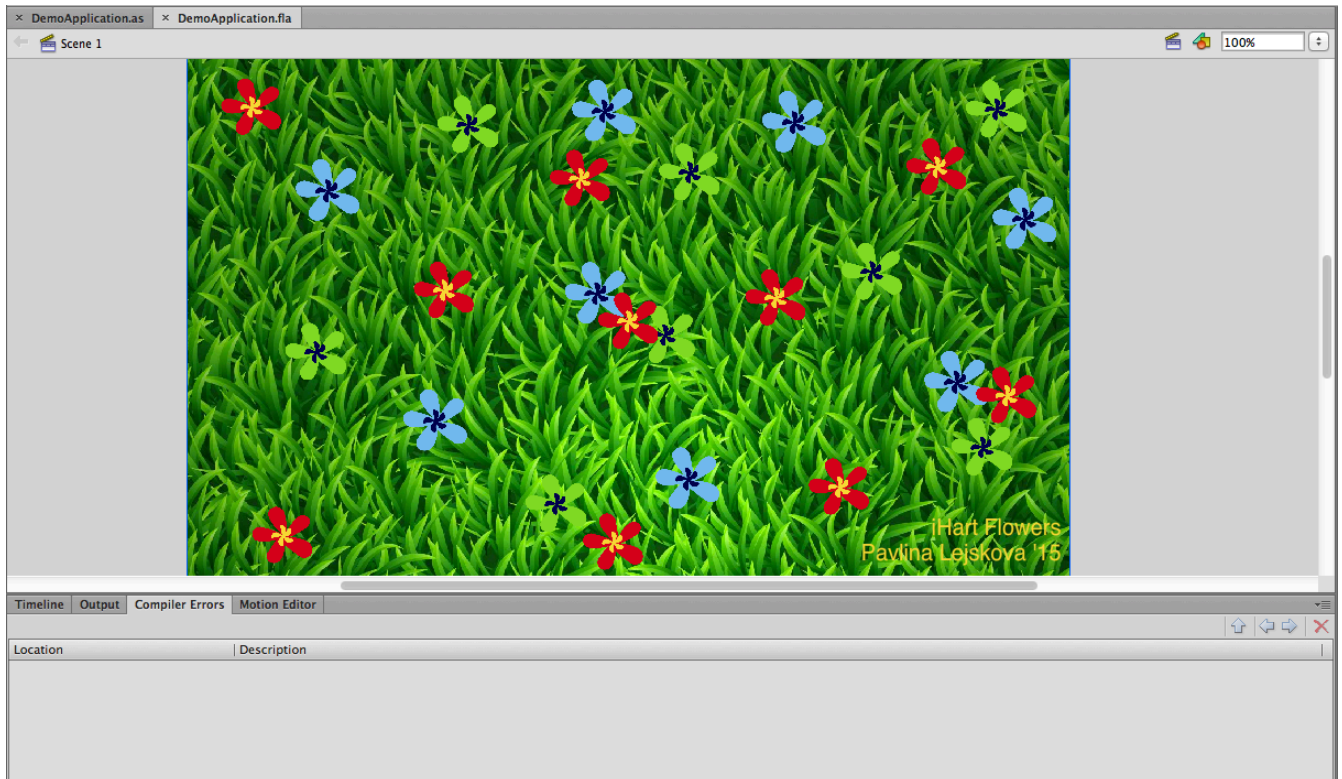
Setup for .fla file:

When creating your application think about what kind of background you want for your application. You also want to choose graphics that will appear as the user interacts with your iHart application. Choose some images save them in the same folder that your .fla and .as are located in!

Choose an image for your .fla file. To **import the image** onto to the Scene go to

File > Import > Import to stage

and then select the image you saved. It should appear on the white background like so

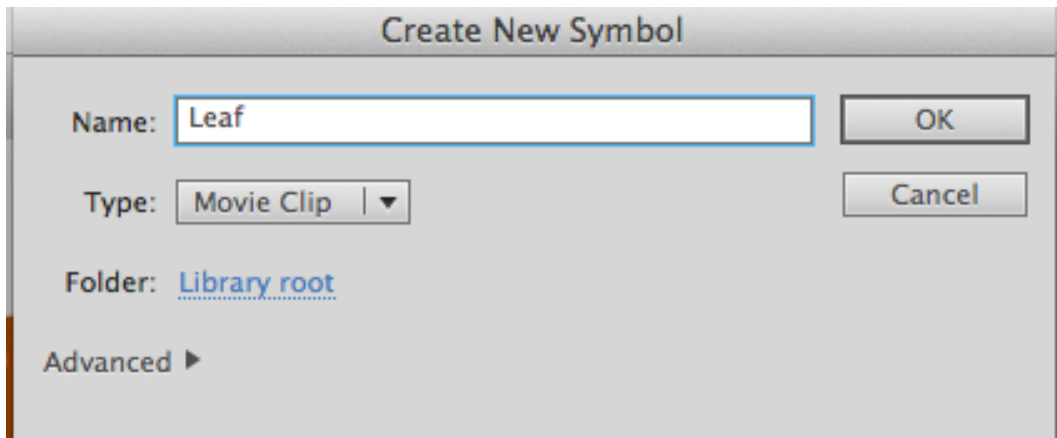


To **create your graphic** in your .fla file go to

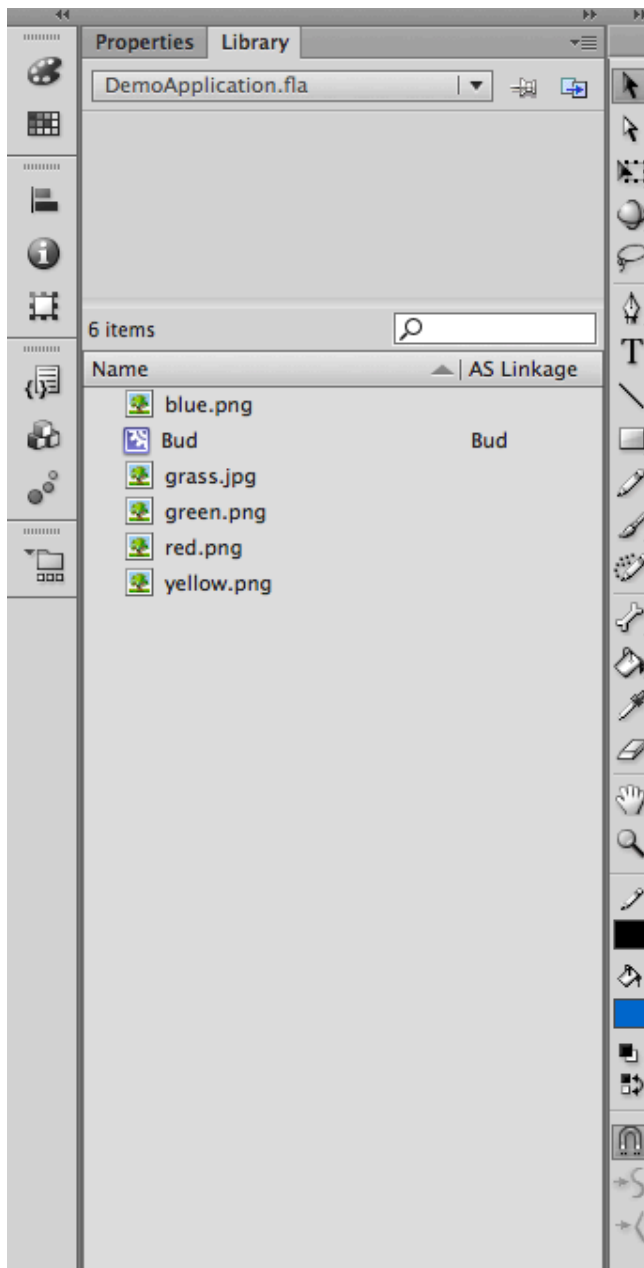
Insert> New Symbol

Name your symbol anything you like, but remember that you will be referring to in your .as file

Choose MovieClip as the type of symbol and name your symbol like so



Once you have created the symbol **import your image graphic** by going to
File > Import > Import to Stage



The new symbol and background image
should appear under the Library tab on the
right in your .fla file

Be sure to edit AS Linkage by double
clicking under “AS Linkage” beside the
symbol and change that to the same name
as the Symbol

Linking the graphics to the code in your .as file:

Back to .as file under your global variables you will be creating another variable to link your graphics to your code.

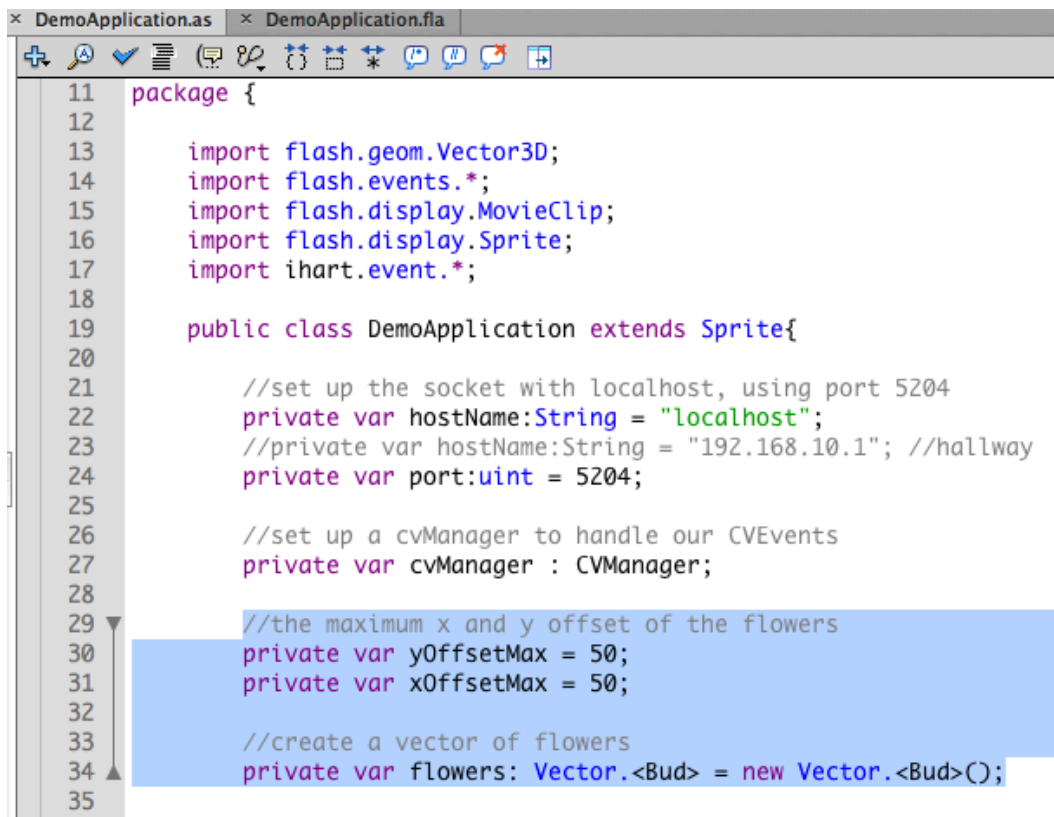
Create a **new instance of your symbol** by inserting the following line of code in the list of variables:

```
private var variable_name: Vector.<Symbol_Name> = new Vector.<Symbol_Name>();
```

(side note: replace variable name with any name you want but be aware that you will be using it again later in the code)

Add these two lines of code to your variables (after public class DemoApplication extends Sprite):

```
//the maximum x and y offset of the symbols  
private var yOffsetMax = 50;  
private var xOffsetMax = 50;
```



```
11 package {  
12  
13     import flash.geom.Vector3D;  
14     import flash.events.*;  
15     import flash.display.MovieClip;  
16     import flash.display.Sprite;  
17     import ihart.event.*;  
18  
19     public class DemoApplication extends Sprite{  
20  
21         //set up the socket with localhost, using port 5204  
22         private var hostName:String = "localhost";  
23         //private var hostName:String = "192.168.10.1"; //hallway  
24         private var port:uint = 5204;  
25  
26         //set up a cvManager to handle our CVEvents  
27         private var cvManager : CVManager;  
28  
29         //the maximum x and y offset of the flowers  
30         private var yOffsetMax = 50;  
31         private var xOffsetMax = 50;  
32  
33         //create a vector of flowers  
34         private var flowers: Vector.<Bud> = new Vector.<Bud>();  
35  
36     }
```

Now its time to **define the getData function!**

For this function you will be renaming certain variables for your own purposes

Add the following code

Be aware that you have to rename **any of the code in orange** for the purposes of your specific application

1. For the variable **currentFlower** come up with your own name (e.g. currentBall, currentTriangle, currentHeart, etc.) for a new variable that will hold your Symbol
2. Make this variable the same type as your Symbol (so whatever name you gave your Symbol in the .fla file)
3. Add the Symbol to your variable by using `currentName = new SymbolName();` using your own variable and Symbol names
4. For the rest of the orange lines replace “currentFlower” with whichever name you have for your variable
5. It is also important to update the comments so that they reflect your specific application. All the lines that come after double slashes (`//`) or after a slash and two asterisks

(`/**`)

`/**`

`* Gets data about a CVEvent and creates flowers based on that data.`

`*/`

`public function getData (e : CVEvent) : void {`

`trace("Acquiring data... ");`

`var numBlobs : int = e.getNumBlobs();`

`var blobX : Number;`

`var blobY : Number;`

`var rot : Number;`


```

        var currentFlower : Bud;

        //for every blob there is on the screen
        for (var i : int = 0; i < numBlobs; i++) {

            currentFlower = new Bud();

            //save the blob's x and y values
            blobX = e.getX(i);
            blobY = e.getY(i);

            //add a random rotation to the flower (between 0 and 60degrees)
            rot = Math.random() * 60;
            currentFlower.rotation = rot;

            //add the new flower to the scene
            addChild(currentFlower);

            //generate the new flower's x and y based on the blob's x and y
            currentFlower.x = generateOffset(blobX, xOffsetMax);
            currentFlower.y = generateOffset(blobY, yOffsetMax);

            //add the new flower to the vector
            flowers.push(currentFlower);

            //remove any old flowers
            removeOld();
        }
    }

```

Defining Additional Functions:

In `getData()` we made calls or references to two other functions that we have not define yet:

`removeOld();`

This function removes flowers that have disappeared from view from the screen and the vector.

```
generateOffset();
```

This function generates a random offset for a flower's x or y value.

In order to **define these functions** we must **add** these function declarations:

```
public function removeOld () : void { }
```

and

```
public function generateOffset (initialNum: Number, maxOffset: Number): Number { }
```

In between the brackets we will be inserting code to define what these functions do

```
{ INSERT CODE HERE }
```

For removeOld() **insert** the following code in between the brackets { }

Again you must edit the **orange code** slightly to fit your application

1. Replace “flowers” with the name you chose for the variable declaration you made earlier in your code (reminder: private var **variable_name**: Vector.<Symbol_Name> = new Vector.<Symbol_Name>();)

```
//for every flower in the vector
for (var i : int = 0; i < flowers.length; i++)
{

    //if the reference portion of the flower is not visible
    if (flowers[i].currentFrame == 20)
    {
        //remove the flower from the screen
        removeChild(flowers[i]);
    }
}
```

```

        //remove the flower from the vector
        flowers.splice(i, 1);
    }
}

```

For generateOffset() **insert** the following code in between the brackets { }

```

//set the amount of offset randomly
var offset : Number = Math.random() * maxOffset;

//set the sign of the offset (0 is negative, 1 is positive)
var sign : int = Math.floor(Math.random() * 2);

if (sign == 0) {
    return initialNum - offset;
}

return initialNum + offset;

```

You have finished coding your iHart application!!


Now just **review** and make sure you have everything (Note: these are modeled after the demo application so not everything will be the same):

- make sure your class extends Sprite
- import statements
- variable declarations
- lines of code in the constructor instantiating and adding the cvManager that calls getData()
- declaration and definition of function getData()
- declaration and definition of function removeOld()
- declaration and definition of function generateOffset()

Finally in your .fla file you will need to change the Publish Settings for your flash application to use iHart

File > Publish Settings

Under **Local playback security** change it to **Access network only**

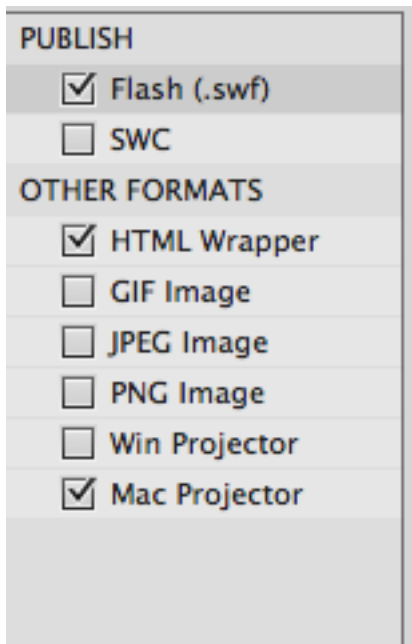
Then on the top of the Publish Settings page look for this symbol  and click it

Then click the icon that looks like a folder from this toolbar



A browsing window will show up and from here you should navigate to the folder holding your iHart Application then **Applications > flash > ihart**

Stop here and click **Choose**



Still in the Publish Settings page choose Win projector or Mac projector (depending on which system you are working on) from the Publish menu on the left side. This should create a .app file with your application name.

←

Now your application should be ready to use! Run iHart and then click on the .app file of your application to see your creation working!

