Array Lists

What is an “ArrayList”?

An *ArrayList* stores a variable number of objects. This is similar to making an array of objects, but with an *ArrayList*, items can be easily added and removed from the *ArrayList* so it is resized dynamically. This can be very convenient, but it's slower than making an array of objects when using many elements.

How do I use an “ArrayList”?

An *ArrayList* has many methods used to control and search its contents. For example, the length of the *ArrayList* is returned by its ***size()*** method, which is an integer value for the total number of elements in the list. An element is added to an *ArrayList* with the ***add()*** method and is deleted with the ***remove()*** method. The ***get()*** method returns the element at the specified position in the list.

Why should I use an “ArrayList”?

You may be asking the question to yourself, “why are *ArrayLists* important and why should I use them?”. Well it’s good to be asking yourself these questions! To answer these questions let’s look at an example.

Note: If you need to refresh your memory on how Classes work, look back at the “*Bouncing Balls with Classes*” lesson or the “*Targets with Classes - Revision Lesson*”.

In the example below we have a ***Snowman*** class that contains all of the variables and methods used to draw a snowman (although not a very good one!) on the screen.

SnowmanArrayListExample | Snowman

```
class Snowman {
 float xPos, yPos;
 float size;

 Snowman(float xPos, float yPos, float size) { // snowman constructor
   this.xPos = xPos;
   this.yPos = yPos;
   this.size = size;
 }

 void display() { // method that will draw the snowman
   noStroke();
   fill(255);
   ellipse(xPos, yPos, size, size); //lower body
   ellipse(xPos, yPos-size/2, size/2, size/2); // middle body
   ellipse(xPos, yPos-size/1.2, size/4, size/4); // top body
   fill(0);
   ellipse(xPos-size/15, yPos-size/1.2-size/15, size/30, size/30); // left eye
   ellipse(xPos+size/15, yPos-size/1.2-size/15, size/30, size/30); // right eye
   ellipse(xPos, yPos-size/20, size/30, size/30); // button 1
   ellipse(xPos, yPos-size/5, size/30, size/30); // button 2
   ellipse(xPos, yPos-size/2.75, size/30, size/30); // button 3
 }
}
```

Let's say that we want to draw four snowmen on the screen in different locations and of different sizes. Until now, we have had to declare four snowman variables, initialize each one, and call the ***display()*** method four times, like in the example below:

SnowmanArrayListExample | Snowman

```
Snowman snowman1;
Snowman snowman2;
Snowman snowman3;
Snowman snowman4;

void setup() {
    size(1000,1000);
    snowman1 = new Snowman(random(0,width),random(0,height),random(50,400));
    snowman2 = new Snowman(random(0,width),random(0,height),random(50,400));
    snowman3 = new Snowman(random(0,width),random(0,height),random(50,400));
    snowman4 = new Snowman(random(0,width),random(0,height),random(50,400));
}

void draw() {
    snowman1.display();
    snowman2.display();
    snowman3.display();
    snowman4.display();
}
```

Obviously this way of doing things can be very time-consuming - which is where *ArrayLists* come in very handy!

The code in the example below will work the exact same way as the example above, however, as you can see there is less code to type when using an *ArrayList*.

SnowmanArrayListExample | Snowman

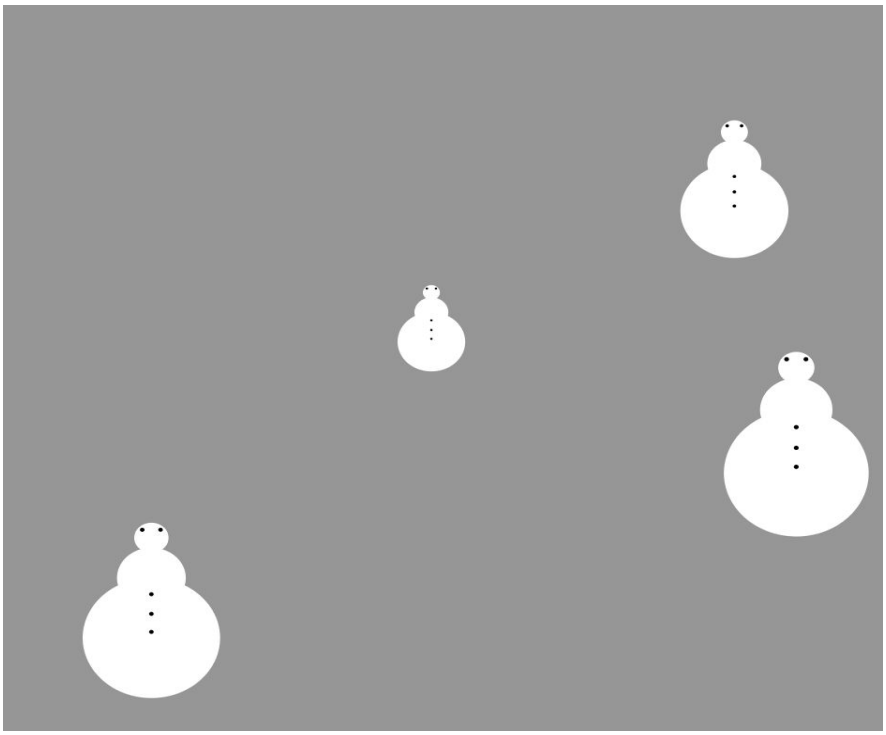
```
ArrayList<Snowman> snowmanList;

void setup() {
    size(1000,1000);
    snowmanList = new ArrayList<Snowman>(); // initialize snowman list

    for(int i=0; i<4; i++) {
        // load new snowman into snowman list
        snowmanList.add(new Snowman(random(50,width-50),random(50,height-50),random(50,400)));
    }
}

void draw() {
    for(int i=0; i<snowmanList.size(); i++) {
        snowmanList.get(i).display();  // call display() method of each snowman
    }
}
```

Regardless of what method you use, you will get the following result (the snowmen will be in different locations and different sizes):



Tasks:

1. Download the “Snowman ArrayList Example” zip folder containing the Snowman class and the main file. This can be found at http://www.theacademyofcode.com/resources.
2. Save the program as “**Snowman ArrayLists**”.
3. Add a method to the Snowman class called “***move()***” that makes the snowman move horizontally across the screen to the right.
4. Create a function in your main window that will **remove** a snowman from the list if the **x-position** of the snowman is **greater than the width of the screen plus half the width of the snowman**.

   To remove a snowman from the *ArrayList*, instead of saying ***snowmanList.get(i)*** when you are accessing a snowman on the list, you say ***snowmanList.remove(i)***.

   Note: It is important to remove items from an *ArrayList* when we are finished with them (when they disappear off the screen). If we do not remove them then the *ArrayList* will become very large and the programme will start to slow down significantly.

5. Create a function in your main window that will **add** a new snowman to the list if the **size of the *ArrayList* is less than 4**.
6. Call the new snowman class method and two new functions in *void draw()*.

Extra Tasks:

1. Increase the amount of snowmen that we see on the screen at any one time.
2. Make each snowman a different colour. To do this you should pass variables for the amount of ***red***, ***green***, and ***blue*** into the snowman **constructor** as arguments.
3. Make each snowman have a different **opacity** (transparency). This will follow the same steps as the previous task.