Arrays and For Loops

Variables

Variables allow us to use symbolic names to represent stored data.

Every variable must be declared before use by specifying its type and name.

Anywhere in our code that we can use a literal value (such as 5, "hello", 8.2, or false) we can also use a variable that stores a value of the matching type.

If we need to store more data, we just declare more variables.

Until now...

The problem: Excessive verbosity

You're programming a pinball game. You know you need to declare variables to store the location of the ball.

```
int xPosition;
int yPosition;
```

But what if you wanted to create a level of the game where multiple pinballs are on the table at the same time?

```
int xPos2;
int yPos2;
```

Seems like we're going to be declaring a lot of variables.

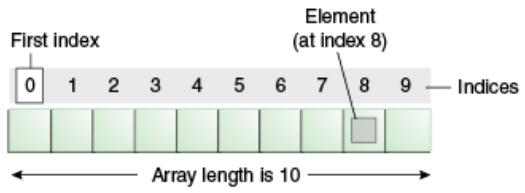
The solution: New syntax

Instead of declaring many different variables that all store the same general kind of information, we can use an array:

```
int[] xPosition = new int[100];
```

This array can hold 100 integers. Their names, which work just like any other variable, are as follows:

```
xPosition[0]
xPosition[1]
xPosition[2]
...
xPosition[99]
```



The vocabulary

These are square brackets: [ ] When we declare an array, they come after the type, and before the variable name:

```
int[] xPosition = new int[100];
```

Note that when we initialize the array, we also specify the size in square brackets.

According to the declaration above, this array can hold 100 `int` *elements*. Each of the elements can be accessed individually by specifying an index in square brackets after the array name:

```
xPosition[5] = 200;
```

1-D Array Initialization

- Which of these are valid ways to assign a reference to an array?
  - double[] data = new double[25];
  - double data[] = new double[25];
  - double[] data;
    data = new double[25];
- All are three are valid!

One Dimensional array

Initialization `int a[] = new int [12];`

`System.out.print(a[5]);` Output: 6

1-D initializing values

- Small arrays can have values initialized with either of the following ways:
  - int[] coins = new int[4];
    coins[0] = 1;
    coins[1] = 5;
    coins[2] = 10;
    coins[3] = 25;
  - int[] coins = {1, 5, 10, 25};

Array Length

- length is a variable of arrays:
  String[] names = new String[25];
  names.length; //returns 25
- Array indices go from 0 to names.length-1 (i.e. 0 to 24)
- length is not a method for arrays; length is a method for Strings

The application

We can use these variables just like any other **`int`** variables:

```
xPosition[5] += xSpeed;

if(xPosition[5] > 640)
{
  xPosition[5] = 0;
}
```

Using **`xPosition[5]`** this way is no different from using an individual variable named, say, **`xPos5`**. The advantage is that we declared all 100 variables (**`xPosition[0]`** to **`xPosition[99]`**) with a single line of code.

The abstraction

So arrays allow us to declare many variables of a given type under a single name with a single line of code:

```
int[] xPosition = new int[100];
```

And the index used to access elements of the array is an **`int`**:

```
xPosition[5] = 200;
```

This means we can also use a variable as the index:

```
int i = 5;
xPosition[i] = 200;
```

The extension

We also know that we can use loops to repeat code. Here, we will use one variable as both a loop counter and array index:

```
int i = 0;
while( i < 100 )
{
  xPosition[i] = (int)(Math.random() * 640);
  i++;
}
```

After this code runs, the **xPosition** array will contain 100 ints with random values in the range 0-640. But we didn't have to declare 100 variables or write 100 lines of code. The array's index variable is what makes this concision possible.

While vs. For loops

While loops

```
int i = 0;
while (i < 100)
{
   //repeated code
   i++;
}
```

For loops

```
for(int i = 0; i<100; i++)
{
   //repeated code

}
```

Traversing an array using a loop

While loops

```
int i = 0;
while (i < 100)
{
   xPosition[i] =
      (int)(Math.random() * 640);
   i++;
}
```

For loops

```
for(int i = 0; i<100; i++)
{
   xPosition[i] =
      (int)(Math.random() * 640);

}
```

What to do with arrays

- You need to be able to read and write code that accomplishes each of the following:
  - Counting elements
  - Printing elements
  - Summing elements
  - Swapping elements
  - Finding the minimum or maximum
  - Inserting elements
  - Deleting elements