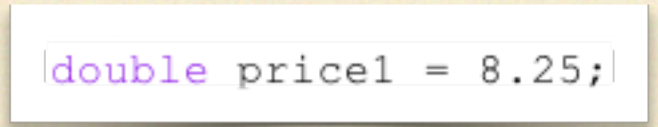
Image Credit: Julius Schorzman

# Java Lesson 8 Tutorial

Produced by Byron Minick

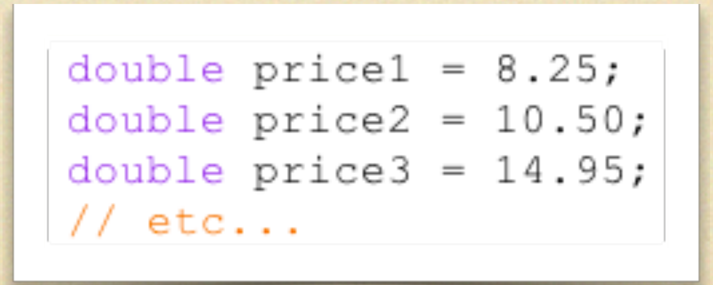
# Arrays

- In the past, you learned that a variable could store only one value at a time, as demonstrated here:



```
double price1 = 8.25;
```

- **However, this lead to a problem:** What if we need to store hundreds of values? Well, this meant that the programmer would have to manually create a brand new variable for each piece of information that needs stored in the computer's memory. Of course, is would *not* be very efficient or desirable.

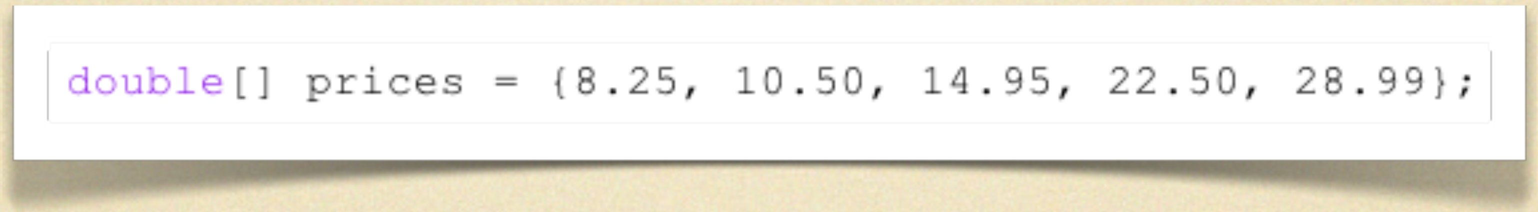


```
double price1 = 8.25;
double price2 = 10.50;
double price3 = 14.95;
// etc...
```

# Arrays to the Rescue

- To solve this problem, Java developers included a powerful feature called the array! An array is simply a list of values that you can assign to a single variable name.

- For example, say we wanted to create a list of prices for items at the grocery store. With an array, it's easy to do:

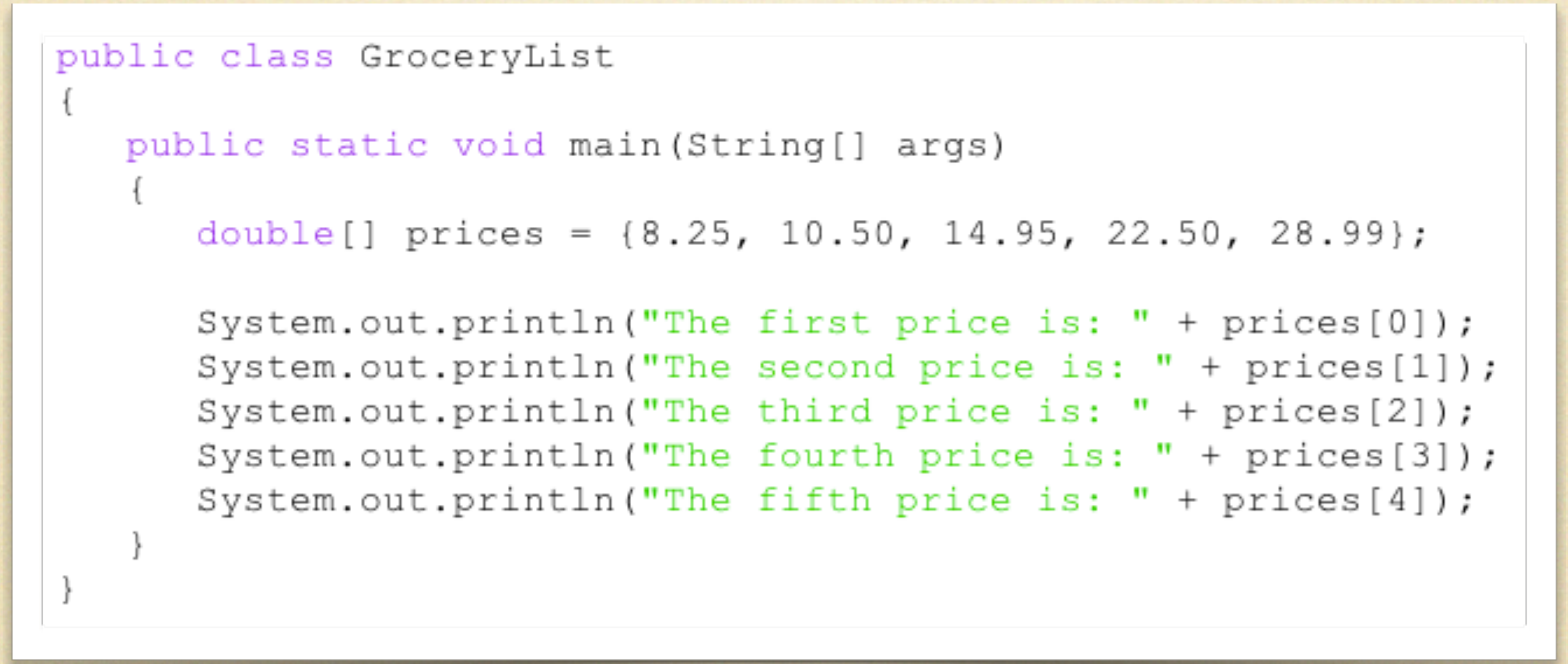


```
double[] prices = {8.25, 10.50, 14.95, 22.50, 28.99};
```

- By creating an array, the *prices* variable is now able to store **5** values (prices), instead of only one!

# Arrays

- Working with arrays in Java is easy.  Consider the following example, which displays all five prices:



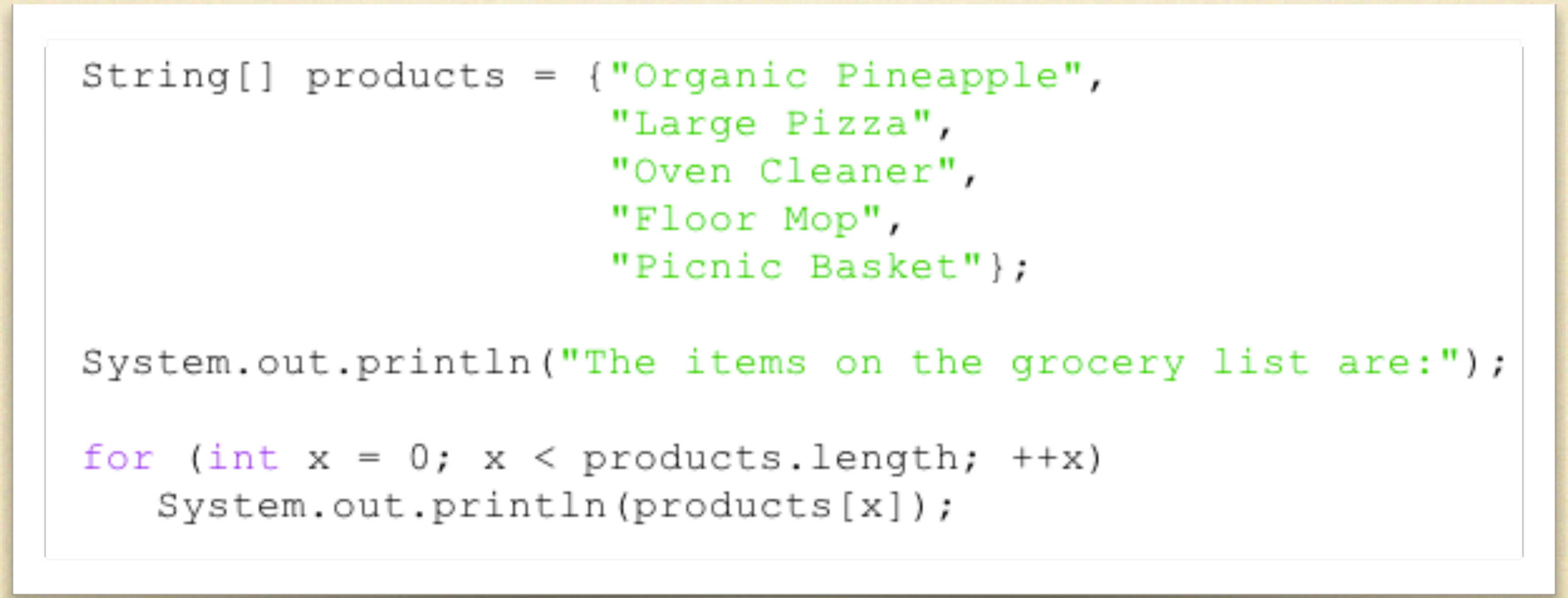
```java
public class GroceryList
{
   public static void main(String[] args)
   {
      double[] prices = {8.25, 10.50, 14.95, 22.50, 28.99};

      System.out.println("The first price is: " + prices[0]);
      System.out.println("The second price is: " + prices[1]);
      System.out.println("The third price is: " + prices[2]);
      System.out.println("The fourth price is: " + prices[3]);
      System.out.println("The fifth price is: " + prices[4]);
   }
}
```

- **Note:** Since you can store multiple values in an array, you need to tell Java which item you desire to use.  Items in an array are referred to by a ***subscript***, which always starts at 0 (zero) and goes up from there, as you can see in the example above.

# Arrays

- You can create arrays of integers and Strings too!  Consider the following items on the shopping, product list:

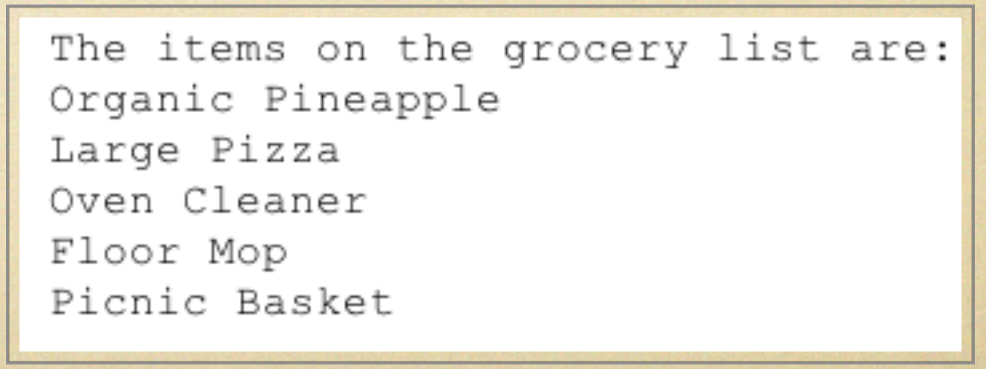


```
String[] products = {"Organic Pineapple",
                     "Large Pizza",
                     "Oven Cleaner",
                     "Floor Mop",
                     "Picnic Basket"};

System.out.println("The items on the grocery list are:");

for (int x = 0; x < products.length; ++x)
   System.out.println(products[x]);
```

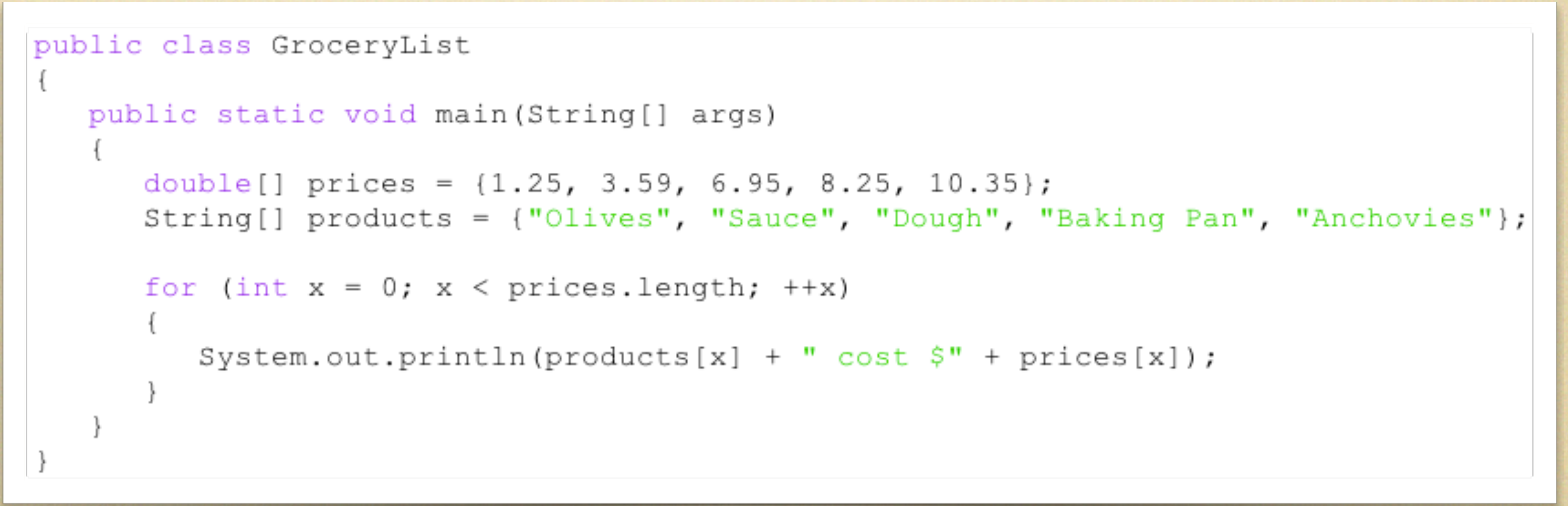
- Notice the *for()* loop.  It displays all of the items stored in the array.



```
The items on the grocery list are:
Organic Pineapple
Large Pizza
Oven Cleaner
Floor Mop
Picnic Basket
```

# Parallel Arrays

- Parallel Arrays are a useful feature when you want to display multiple pieces of information that are associated. For example, a grocery list might contain the product's name and how much it costs.
- Consider the following:

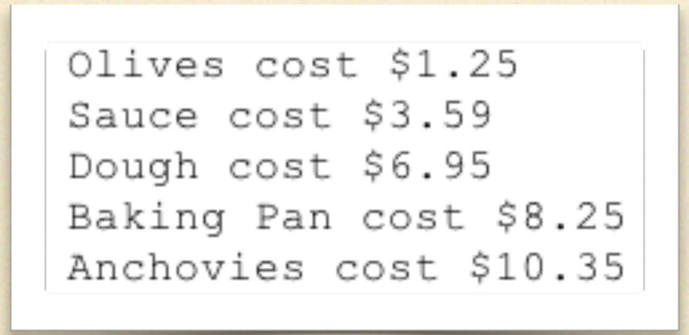


```
public class GroceryList
{
   public static void main(String[] args)
   {
      double[] prices = {1.25, 3.59, 6.95, 8.25, 10.35};
      String[] products = {"Olives", "Sauce", "Dough", "Baking Pan", "Anchovies"};

      for (int x = 0; x < prices.length; ++x)
      {
         System.out.println(products[x] + " cost $" + prices[x]);
      }
   }
}
```

# Parallel Arrays

- When you run the *GroceryList* program, you'll see the following results:

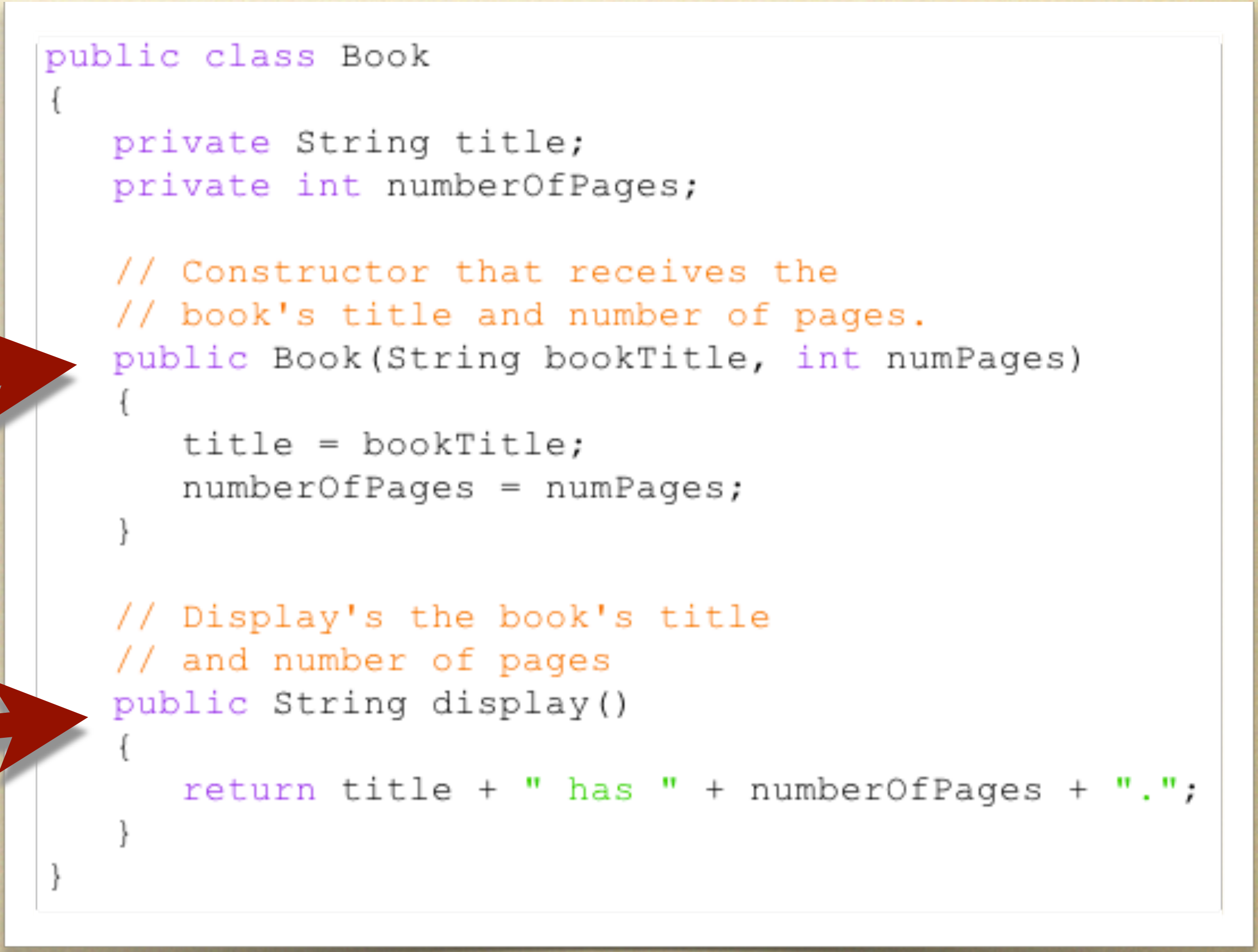


```
Olives cost $1.25
Sauce cost $3.59
Dough cost $6.95
Baking Pan cost $8.25
Anchovies cost $10.35
```

- As you can see, parallel arrays make it easy to display groups of related data.

# Array of Objects

- One of the best features of arrays is that you're not limited to just working with primitive data types, such as integers or doubles. You can also create an array of objects!
- Consider the following ***Book*** class:
- It allows you to store the *book's title* and how many *page numbers* that it contains.
- It also contains a method to display the book's info.



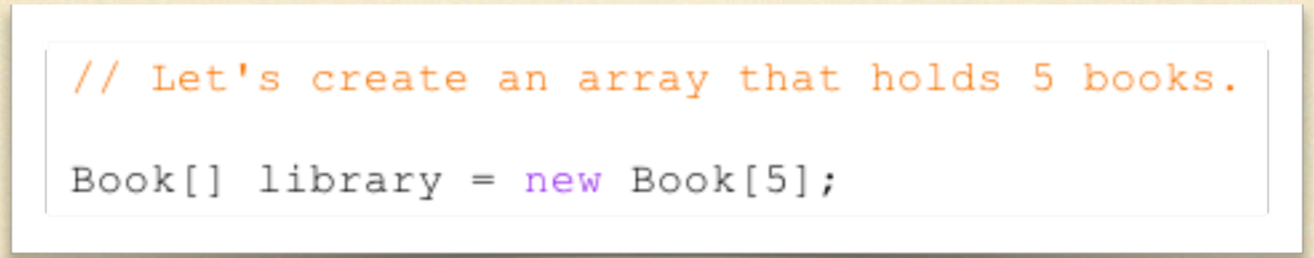
```java
public class Book
{
    private String title;
    private int numberOfPages;

    // Constructor that receives the
    // book's title and number of pages.
    public Book(String bookTitle, int numPages)
    {
        title = bookTitle;
        numberOfPages = numPages;
    }

    // Display's the book's title
    // and number of pages
    public String display()
    {
        return title + " has " + numberOfPages + ".";
    }
}
```

# Array of Objects

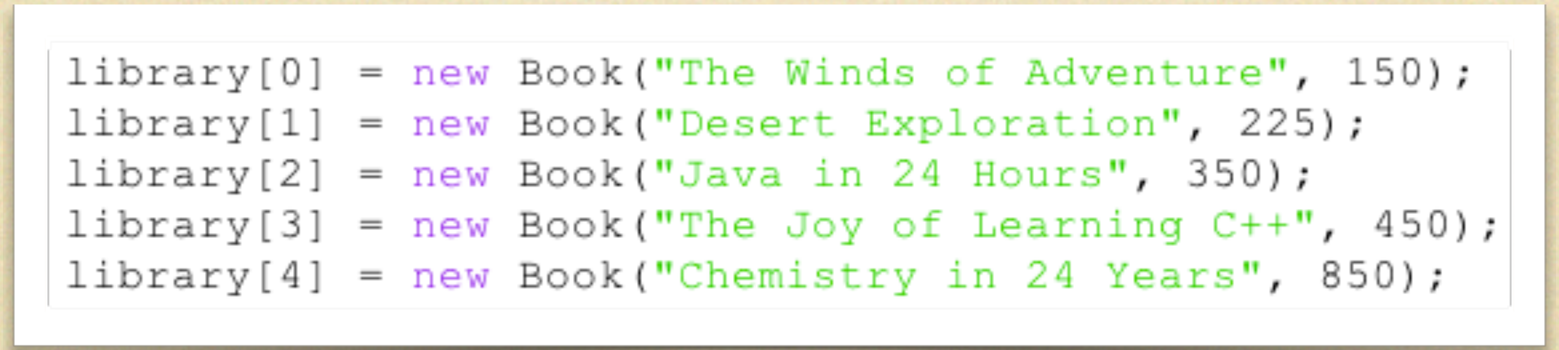
- To create an array of books, you first need to specify how many books will be stored in the array itself.



```java
// Let's create an array that holds 5 books.

Book[] library = new Book[5];
```

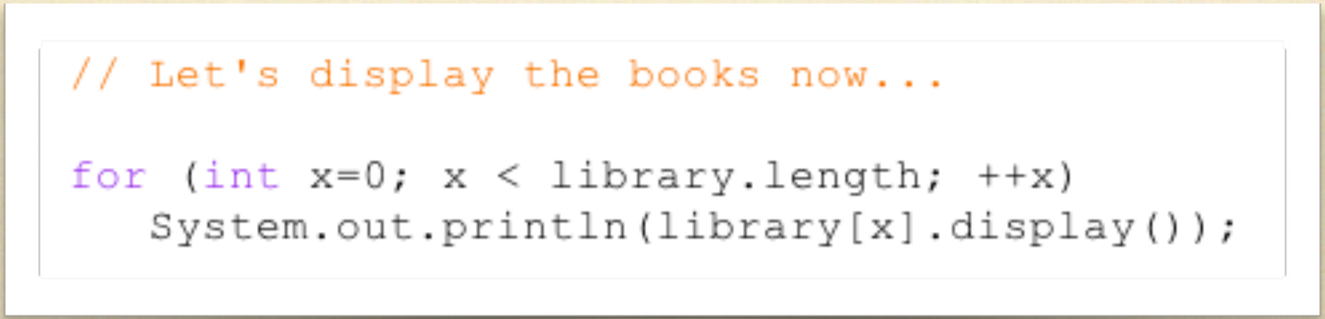
- Once the book array is defined, you then need to instantiate the individual book objects. Here's an example:



```java
library[0] = new Book("The Winds of Adventure", 150);
library[1] = new Book("Desert Exploration", 225);
library[2] = new Book("Java in 24 Hours", 350);
library[3] = new Book("The Joy of Learning C++", 450);
library[4] = new Book("Chemistry in 24 Years", 850);
```

# Array of Objects

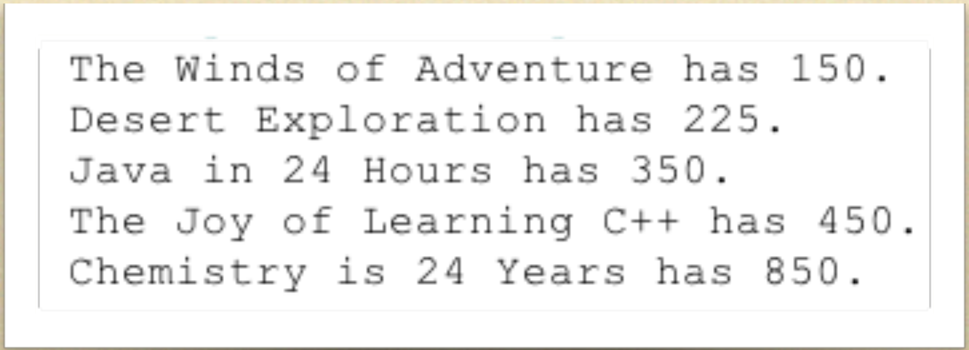
- Now that we have created our Book objects, we can now preview them. Notice the use of the *for()* loop.



```java
// Let's display the books now...

for (int x=0; x < library.length; ++x)
   System.out.println(library[x].display());
```

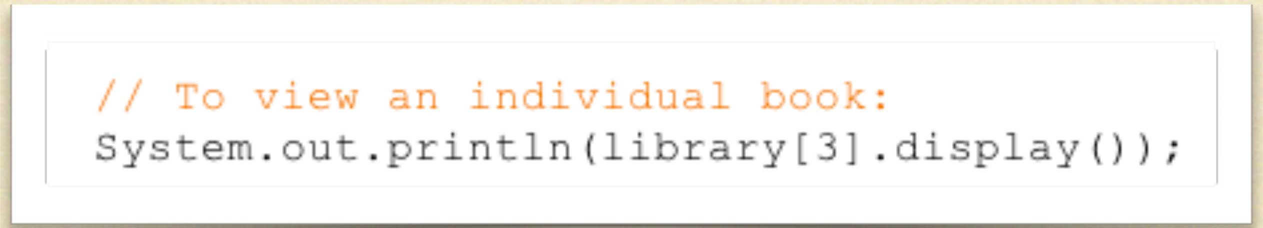
- When we run the program, here's the output:



```
The Winds of Adventure has 150.
Desert Exploration has 225.
Java in 24 Hours has 350.
The Joy of Learning C++ has 450.
Chemistry is 24 Years has 850.
```

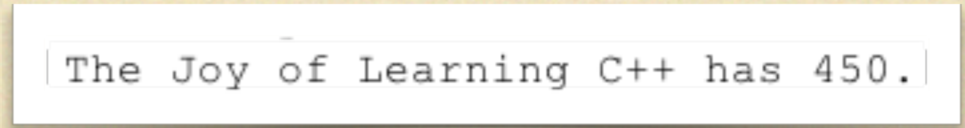
# Array of Objects

- Also, just like an array, you can also specify which particular Book object that you would like to see. Say we want to see the fourth Book object. Here's how to do this:



```
// To view an individual book:
System.out.println(library[3].display());
```

- Notice that when we run this code, it correctly outputs the following:



```
The Joy of Learning C++ has 450.
```