ARRAYS

Introduction

We have seen how to store single pieces of data in variables. What happens when we need to store a group of data? What if we have a list of students in a classroom? Or a ranking of the top 10 horses finishing a horse race?

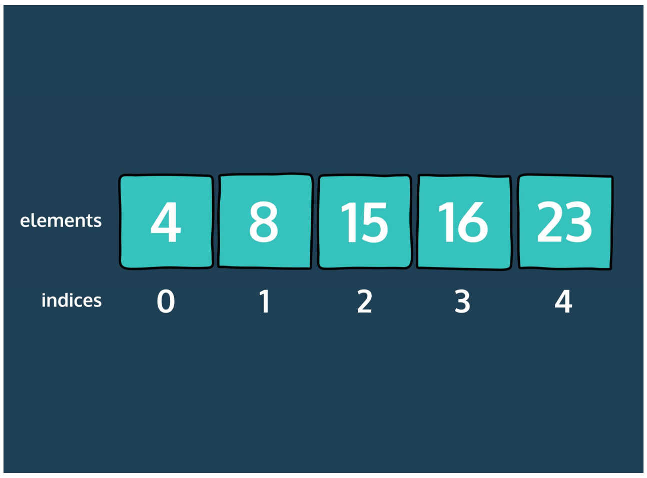
If we were storing 5 lottery ticket numbers, for example, we could create a different variable for each value:

```
int firstNumber = 4;
int secondNumber = 8;
int thirdNumber = 15;
int fourthNumber = 16;
int fifthNumber = 23;
```

That is a lot of ungainly repeated code. What if we had a hundred lottery numbers? It is more clean and convenient to use a Java *array* to store the data as a list.

An array holds a fixed number of values of one type. Arrays hold `double`s, `int`s, `boolean`s, or any other primitives. Arrays can also contain `String`s as well as object references!

Each index of an array corresponds with a different value. Here is a diagram of an array filled with integer values:



Notice that the indexes start at 0! The element at index 0 is `4`, while the element at index 1 is `8`. This array has a length of 5, since it holds five elements, but the highest index of the array is 4.

Let's explore how to create and use arrays in Java, so that we can store all of our Java data types.

Instructions

1. In the code editor, we have a `Newsfeed` class to manage trending articles with their views and ratings.

Throughout this lesson, you'll learn how to create such Java programs with Java arrays.

Run the code and see how the arrays are used.

```java
public class Main{
  public static void main(String[] args){
    String[] robotArticles = {"Oil News", "Innovative Motors",
"Humans: Exterminate Or Not?", "Organic Eye Implants", "Path
Finding in an Unknown World"};
    int[] robotViewers = {87, 32, 13, 11, 7};
    double[] robotRatings = {2.5, 3.2, 5.0, 1.7, 4.3};
    Newsfeed robotTimes = new Newsfeed(robotArticles,
robotViewers, robotRatings);

    robotTimes.viewArticle(2);
    robotTimes.viewArticle(2);
    System.out.println("The top article is " +
robotTimes.getTopArticle());
    robotTimes.changeRating(3, 5);
  }
}
```

```
The article 'Humans: Exterminate Or Not?' has now been viewed 14 times!
The article 'Humans: Exterminate Or Not?' has now been viewed 15 times!
The top article is Oil News
The article 'Organic Eye Implants' is now rated 5.0 stars!
```

Creating an Array Explicitly

Imagine that we're using a program to keep track of the prices of different clothing items we want to buy. We would want a list of the prices and a list of the items they correspond to. To create an array, we provide a name and declare the type of data it holds:

```java
double[] prices;
```

Just like with variables, we can declare and initialize in the same line. This allows us to explicitly initialize the array to contain the data we want to store :

```java
double[] prices = {13.15, 15.87, 14.22, 16.66};
```

We can use arrays to hold `String`s and other objects as well as primitives:

```java
String[] clothingItems = {"Tank Top", "Beanie", "Funny Socks", "Corduroys"};
```

Instructions

1.For now, our `Newsfeed` class doesn't have any methods.

Create a method `getTopics()` that accepts no parameters, returns a `String` array, and is accessible by other classes.

Leave the body empty.

*Note: it is okay for there to be an error claiming there is no `main()` method. This method is defined in **Main.java**.*

2.Inside the `getTopics()` method, create a `String` array called `topics` that contains these elements:
"Opinion", "Tech", "Science", "Health"
Remember to keep the order the same.

Then, return the `topics` array at the end of the method!

To see the output of the program, switch over to the **Main.java** file and click “Run”.

```
public class Newsfeed {
  public Newsfeed(){

  }
  // Create getTopics() below:
    public String[] getTopics(){
      String[] topics = {"Opinion","Tech","Science","Health"};
      return topics;
    }
}
```

```
public class Main {
  public static void main(String[] args) {
    Newsfeed sampleFeed = new Newsfeed();
    String[] topics = sampleFeed.getTopics();
    System.out.println(topics);
  }
}
```

```
[Ljava.lang.String;@7ad041f3
```

Importing Arrays

When we printed out the array we created in the last exercise, we saw a memory address that did not help us understand what was contained in the array.

If we want to have a more descriptive printout of the array itself, we need a `toString()` method that is provided by the Arrays *package* in Java.

```
import java.util.Arrays;
```

We put this at the top of the file, before we even define the class!

When we import a package in Java, we are making all of the methods of that package available in our code.

The `Arrays` package has many useful methods, including `Arrays.toString()`. When we pass an array into `Arrays.toString()`, we can see the contents of the array printed out:

```
import java.util.Arrays;

public class Lottery(){

  public static void main(String[] args){
    int[] lotteryNumbers = {4, 8, 15, 16, 23, 42};
    String betterPrintout = Arrays.toString(lotteryNumbers);
    System.out.println(betterPrintout);
  }
}
```

This code will print:

```
[4, 8, 15, 16, 23, 42]
```

Instructions

1.If you run the code now, You'll see something like this: [Ljava.lang.String;@2aae9190.

This is the memory address of the array and not the actual String array.

To make the code print the actual elements, use the toString() method from the Arrays package.

Import the Arrays package from java.util at the top of the **Main.java** file.

2.Now, use the toString() method from the Arrays package to print array topics in the main() method of **Main.java**.

```java
public class Newsfeed {
  public String[] getTopics(){
    String[] topics = {"Opinion", "Tech", "Science", "Health"};
    return topics;
  }}
```

```java
// import the Arrays package here
import java.util.Arrays;

public class Main {
  public static void main(String[] args) {
    Newsfeed sampleFeed = new Newsfeed();
    String[] topics = sampleFeed.getTopics();
    System.out.println(topics);
     System.out.println(Arrays.toString(topics));
  }}
```

```
[Ljava.lang.String;@7ad041f3
[Opinion, Tech, Science, Health]
```

Get Element By Index

Now that we have an array declared and initialized, we want to be able to get values out of it.

We use square brackets, [and], to access data at a certain index:

```
double[] prices = {13.1, 15.87, 14.22, 16.66};

System.out.println(prices[1]);
```

This command would print:

```
15.87
```

This happens because 15.87 is the item at the 1 index of the array. Remember, the index of an array starts at 0 and ends at an index of one less than the number of elements in the array.

If we try to access an element outside of its appropriate index range, we will receive an ArrayIndexOutOfBoundsExceptionerror.

For example, if we were to run the command `System.out.println(prices[5])`, we'd get the following output:java.lang.ArrayIndexOutOfBoundsException: 5

Instructions

1. Inside the `Newsfeed` class, create a method called `getFirstTopic()` that returns a `String` and accepts no parameters.
Inside the `getFirstTopic()` method, return the first element of the `topics` array.

2. We have added an array called `views` to keep track of how many viewers visit a topic.

Every time someone views a topic, we want to increase the value of the corresponding element in `views` by 1.

For example, if someone views an "Opinion" piece (index of 0 in topics), we will increase the value of the 0th index of views by 1.

Inside the viewTopic() method, implement this functionality. The parameter topicIndex represents the location of the element in topics that was viewed.

*Note: switch over to the **Main.java** file to see the output of the program you wrote.*

```java
public class Newsfeed {
  String[] topics = {"Opinion", "Tech", "Science", "Health"};
  public int[] views = {0, 0, 0, 0};

  public Newsfeed(){

  }
  public String[] getTopics(){
    return topics;
  }

  public String getFirstTopic(){
    return topics[0];
  }
  public void viewTopic(int topicIndex){
    views[topicIndex]++;
  }
}
```

```java
import java.util.Arrays;
public class Main {
  public static void main(String[] args) {
    Newsfeed sampleFeed = new Newsfeed();

    System.out.println("The top topic is " +
sampleFeed.getFirstTopic());

    sampleFeed.viewTopic(1);
    sampleFeed.viewTopic(1);
    sampleFeed.viewTopic(3);
    sampleFeed.viewTopic(2);
    sampleFeed.viewTopic(2);
    sampleFeed.viewTopic(1);

    System.out.println("The " + sampleFeed.topics[1] + " topic has
been viewed " + sampleFeed.views[1] + " times!");
  }
}
```

```
The top topic is Opinion
The Tech topic has been viewed 3 times!
```

Creating an Empty Array

We can also create empty arrays and then fill the items one by one. Empty arrays have to be initialized with a fixed size:

```
String[] menuItems = new String[5];
```

Once you declare this size, it cannot be changed! This array will always be of size `5`.

After declaring and initializing, we can set each index of the array to be a different item:

```
menuItems[0] = "Veggie hot dog";
menuItems[1] = "Potato salad";
menuItems[2] = "Cornbread";
menuItems[3] = "Roasted broccoli";
menuItems[4] = "Coffee ice cream";
```

This group of commands has the same effect as assigning the entire array at once:

```
String[] menuItems = {"Veggie hot dog", "Potato salad",
"Cornbread", "Roasted broccoli", "Coffee ice cream"};
```

We can also change an item after it has been assigned! Let's say this restaurant is changing its broccoli dish to a cauliflower one:

```
menuItems[3] = "Baked cauliflower";
```

Now, the array looks like:

```
["Veggie hot dog", "Potato salad", "Cornbread", "Baked cauliflower",
"Coffee ice cream"]
```

Keep Reading: AP Computer Science A Students

When we use new to create an empty array, each element of the array is initialized with a specific value depending on what type the element is:

Data Type	Initialized Value
int	0
double	0.0
boolean	false
Reference	null

For example, consider the following arrays:

```
String[] my_names = new String[5];
int[] my_ages = new int[5];
```

Because a String is a reference to an Object, my_names will contain five nulls. my_ages will contain five 0s to begin with.

Instructions

1. We've declared a String array called favoriteArticles as an instance field.
We'll keep track of the user's top 10 favorite articles in this string array.

In the constructor, Newsfeed(), initialize favoriteArticles as a new empty String array of size 10.

2. We have created a setFavoriteArticle() method that accepts favoriteIndex and newArticle as parameters.

Inside setFavoriteArticle(), set the value of the favoriteArticles array at index favoriteIndex to the value of newArticle.

For example, if we called `setFavoriteArticle(2, "Celebrity Hands Throughout the Decades")`, the value of `favoriteArticles` at index 2 would be set to “Celebrity Hands Throughout the Decades”.

Switch over to **Main.java** and run the code.

```
public class Newsfeed {
  String[] topics = {"Opinion", "Tech", "Science", "Health"};
  int[] views = {0, 0, 0, 0};
  String[] favoriteArticles;

  public Newsfeed(){
    // Initialize favoriteArticles here:
   favoriteArticles = new String[10];
  }

  public void setFavoriteArticle(int favoriteIndex, String
newArticle){
    // Add newArticle to favoriteArt
    favoriteArticles[favoriteIndex] = newArticle;
  }}
```

```
import java.util.Arrays;
public class Main {
  public static void main(String[] args) {
    Newsfeed sampleFeed = new Newsfeed();

    sampleFeed.setFavoriteArticle(2, "Humans: Exterminate Or
Not?");
    sampleFeed.setFavoriteArticle(3, "Organic Eye Implants");
    sampleFeed.setFavoriteArticle(0, "Oil News");

  System.out.println(Arrays.toString(sampleFeed.favoriteArticles));
  }}
```

```
[Oil News, null, Humans: Exterminate Or Not?, Organic Eye
Implants, null, null, null, null, null, null]
```

Length

What if we have an array storing all the usernames for our program, and we want to quickly see how many users we have? To get the length of an array, we can access the length field of the array object:

```
String[] menuItems = new String[5];
System.out.println(menuItems.length);
```

This command would print 5, since the menuItems array has 5 slots, even though they are all empty.

If we print out the length of the prices array:

```
double[] prices = {13.1, 15.87, 14.22, 16.66};

System.out.println(prices.length);
```

We would see 4, since there are four items in the prices array!

Instructions

1. We have created a method getNumTopics() to fetch how many topics exist in the array.
Inside getNumTopics(), return the length of the topics array.

```
public class Newsfeed {
  String[] topics = {"Opinion", "Tech", "Science", "Health"};
  int[] views = {0, 0, 0, 0};

  public Newsfeed(){

  }
  public String[] getTopics(){
    return topics;
  }
  public int getNumTopics(){
    return topics.length;
  }
}
```

```
import java.util.Arrays;
public class Main {
  public static void main(String[] args){
    Newsfeed sampleFeed = new Newsfeed();
    System.out.println("The number of topics is "+
sampleFeed.getNumTopics());
  }
}
```

```
The number of topics is 4
```

String[] args

When we write `main()` methods for our programs, we use the parameter `String[] args`. Now that we know about array syntax, we can start to parse what this means.

A `String[]` is an array made up of `String`s. Examples of `String` arrays:

```
String[] humans = {"Talesha", "Gareth", "Cassie", "Alex"};
String[] robots = {"R2D2", "Marvin", "Bender", "Ava"};
```

The `args` parameter is another example of a `String` array. In this case, the array `args` contains the arguments that we pass in from the terminal when we run the class file. (So far, args has been empty.)

So how can you pass arguments to main()? Let's say we have this class HelloYou:

```
public class HelloYou {
  public static void main(String[] args) {
    System.out.println("Hello " + args[0]);
  }
}
```

When we run the file HelloYou in the terminal with an argument of "Laura":

```
java HelloYou Laura
```

We get the output:

```
Hello Laura
```

The String[] args would be interpreted as an array with one element, "Laura".

When we use args[0] in the main method, we can access that element like we did in HelloYou.
Instructions
1.Let's make users have the option to personalize the Newsfeed object for either robots or humans.
The main() in **Main.java** will take either "Robot" or "Human" as an argument when we run the file.

If the args[0] array holds "Human", we will initialize the feed with human topics.

If the args[0] array holds "Robot", we will initialize the feed with robot topics.

In the conditional statement on line 6, Replace the blank to check if the `args` has the value `Robot` or not.

2. In the terminal, run the **Main.java** with the argument `Robot`.

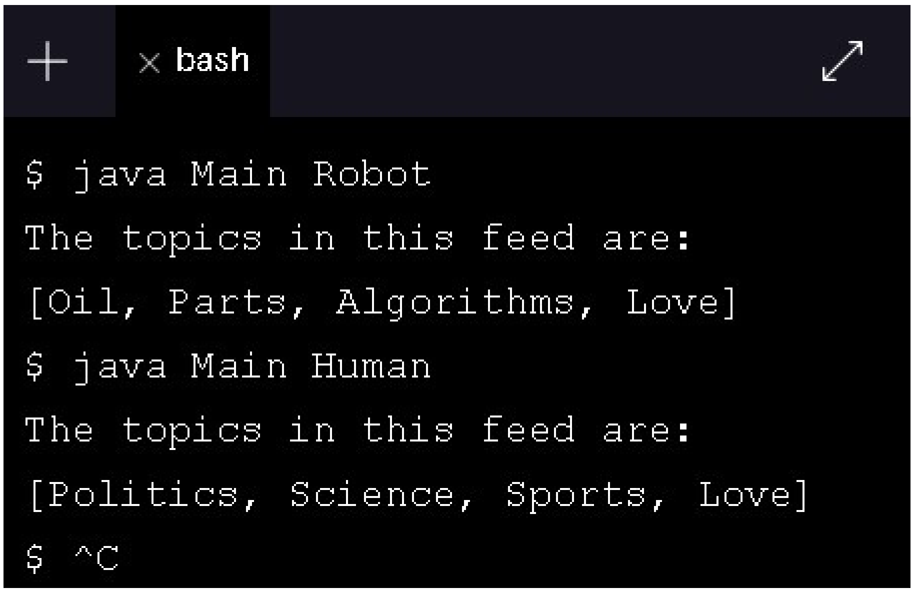
Note: Do not use quotation marks for the argument. Passing in "Robot" as an argument will not work.

3. Now, run the **Main.java** file with the argument `Human`.

Note: Do not use quotation marks for the argument. Passing in "Human" as an argument will not work.

```java
public class Newsfeed {

  String[] topics;

  public Newsfeed(String[] initialTopics){
    topics = initialTopics;
  }
}
```

```java
import java.util.Arrays;

public class Main{
  public static void main(String[] args){
    Newsfeed feed;
    if(args[0].equals("Robot")){
      //topics for a Robot feed:
      String[] robotTopics = {"Oil", "Parts", "Algorithms", "Love"};
      feed = new Newsfeed(robotTopics);
    }
    else if(args[0].equals("Human")){
      //topics for a Human feed:
      String[] humanTopics = {"Politics", "Science", "Sports", "Love"};
      feed = new Newsfeed(humanTopics);
    }
    else{
      String[] genericTopics = {"Opinion", "Tech", "Science",
"Health"};
      feed = new Newsfeed(genericTopics);
    }

    System.out.println("The topics in this feed are:");
    System.out.println(Arrays.toString(feed.topics));
  }
}
```



```
$ java Main Robot
The topics in this feed are:
[Oil, Parts, Algorithms, Love]
$ java Main Human
The topics in this feed are:
[Politics, Science, Sports, Love]
$ ^C
```

Review

We have now seen how to store a list of values in arrays. We can use this knowledge to make organized programs with more complex variables.
Throughout the lesson, we have learned about:

- Creating arrays explicitly, using `{` and `}`.
- Accessing an index of an array using `[` and `]`.
- Creating empty arrays of a certain size, and filling the indices one by one.

- Getting the length of an array using `length`.
- Using the argument array `args` that is passed into the `main()` method of a class.

Instructions

1. Let's practice what we've learned throughout this lesson.

First, inside `main()` in **Main.java**, create a String array `students` and initialize it with these elements:

  - "Sade"
  - "Alexus"
  - "Sam"
  - "Koma"

2. Now, to store the scores of the recent math test, create an array called `mathScores` of type `int` and set it to an empty array of size 4.

Each spot in this array will represent the math test score of the student in the corresponding spot in the `students` array. For example: `mathScores[2]` will represent the score `students[2]` which is "Sam".

3. The students had the following scores on their most recent math test:

  - Sade got a 64
  - Sam got a 76
  - Alexus got a 57
  - Koma got a 98

Store these values in the appropriate spots in the `mathScores` array.

4.Print out the math scores of each student. Print each score on a new line and use the following format:
Scott: 88

```java
import java.util.Arrays;
public class Main {
  public static void main(String[] args) {
    String[] students = {"Sade", "Alexus", "Sam", "Koma"};
    int[] mathScores = new int[4];
    mathScores[0] = 64;
    mathScores[2] = 76;
    mathScores[1] = 57;
    mathScores[3] = 98;

    for(int i = 0; i < students.length; i++){
      System.out.println(students[i] + ": " + mathScores[i]);
    }
  }
}
```

```
Sade: 64
Alexus: 57
Sam: 76
Koma: 98
```