ACTIVITY **1.2.3**

Data Storage

INTRODUCTION

In the last activity, you used an **array algorithm** to determine the average cost of a number of songs. In this activity, you will learn other common array algorithms and practice writing loops. Knowing the common array algorithms is an important part of becoming a computer scientist, but these algorithms can cause some of the most difficult and frustrating bugs in a program. Trying to access elements that don't exist, using the wrong index values, iterating too many or too few times, can all cause problems that occur during runtime. You will learn what causes some of these problems and how to fix them.

array
A collection or list of similar data items, accessed through a single variable reference.

algorithm
A sequence of instructions that accomplishes a task.

Materials

- Computer with BlueJ and Android™ Studio
- Android™ tablet and USB cable, or a device emulator

RESOURCES

Lesson 1.2 Reference Card for Strings
Resources available online

Procedure

Part I: Fix Some Bugs

To begin the activity, first experiment with some of the most common mistakes programmers make when dealing with arrays.

1. Review the most **Common Mistakes** regarding arrays.

2. Open your BlueJ MediaLib project and create a class called `Array Mistakes`. Replace the content created by BlueJ with the following (buggy) content:

```
public class ArrayMistakes
{
   public static void main()
   {
     int [] nums;

     System.out.print("\nForward");
     for (int i = 1; i <= nums.length(); i++) {
        System.out.print(": " + nums[i]);
     }

     System.out.print("\nBackward" );
     for (int i = nums.length; i > 0; i--) {
        System.out.print( ": " + nums[i]);
     }
}
```

This code has many problems with it—some are syntax errors, others occur as runtime errors.

3. Fix the bugs in this code so that the following output occurs:

```
Forwards: 6: 9: 14: 19
Backwards: 19: 14: 9: 6
```

Part II: Fix Logic Error 1

In *Activity 1.2.2 Today's Top 40*, you learned to use a for-each loop and a for loop to iterate over an array called `songs`. In this part, you will look more closely at these two approaches to iteration.

4. Observe each construct side-by-side:

for (Song s : songs)	for (int i=0; i < songs.length; i++)
{	{
s.setTitle("");	songs[i].setTitle("");
}	}

5. Compare the body of each loop.

- In the for-each loop, you access a method using a temporary variable `s`.
- In the standard for loop, you access the method with the array name `songs` and its index `i`.

Standard for loops are used when you need to know the index of an element in your array.

6 Open your MediaLib project and create a new class called `SongFinder`. Replace the code generated by BlueJ with the following:

```
 1: public class SongFinder
 2: {
 3:    public static int findTitle(Song[] songs, String target){
 4:      for (int i = 0; i < songs.length; i++){
 5:
 6: word = songs[i].getTitle();
 7:   if (word.equals(target)) {
 8:   return i;
 9:           }
10: else return -1;
11:       }
12:       return -1;
13:     }
14: }
```

The code above is similar to the `findString` method you just read about, with these notable differences:

- Line 3: The `songs` array is passed as the first parameter.
- Line 7: Because you have a `for` loop, the algorithm uses the new syntax `array[index].methodname()`as in `songs[i].getTitle()`.
- The algorithm has a bug!

7 In `ArrayMediaLib`, call `SongFinder.findTitle(…)` with the title of your third song and show the result, using the following code:

```
int index = SongFinder.findTitle(topTenSongs, "Mack the Knife");
if (index >=0 ) {
   System.out.println("Found " + topTenSongs[index].getTitle());
}
else {
   System.out.println("Song not found!");
}
```

If you have a different title, be sure to use it in your call to `findTitle`.

The song is not found, even though you know it is there.

To find the bug, a **code walk-through** may help. A code walk-through is the manual process of writing down what each line of

code walk-through
A process of learning or testing a section of code, one line at a time, while manually recording variable values and other important information.

code does. The following code walk-through represents what `findTitle` *should* do, *without* the bug.

Assume `target` is the title of the third song in the array:

i	word	return value
0	the title of your first song	Target does not match, n/a
1	the title of your second song	n/a
2	the title of your third song	2

8 To see what `findTitle` *really* does, perform your own walk-through of `findTitle` with your third song title as `target`:

i	word	return value

Your code walk-through should have shown you where the problem exists in `findTitle`.

9 Fix the bug and test to see whether your algorithm works using the test cases below. Also, fix any new problems that turn up.

- When a `target` can be found in the array of songs.
- When a `target` cannot be found.
- When one or both parameters are null as in `SongFinder.findTitle(null,null);`

Part III: Fix Logic Error 2

Learning to walk through code to identify logic errors is a very valuable programming skill, and one that gets better with practice. Follow the steps below to explore another common logic error.

10 In your `SongFinder` class, create a new method `getIndexLastDiscount`, which introduces another logic error common to arrays and loops.

```
public static int getIndexLastDiscount(Song[] songs, double compare){
   if (songs == null) return -1;

int found = -1;
   for (int i = songs.length - 1; i >=0; i--) {
       if (songs[i].getPrice() < compare) {
           found = i;
       }
       else {
         found = -1; // to show none found
      }
   }
   return found;
}
```

11 Call this method in `ArrayMediaLib` somewhere after you have applied the $.99 discounted price to some of your songs:

```
index = SongFinder.getIndexLastDiscount(topTenSongs, 1.00);
if (index >= 0 ) {
    System.out.println("Discount found " + topTenSongs[index].getTitle());
}
else {
   System.out.println("No songs are discounted");
}
```

The call to `getIndexLastDiscount` indicates a discounted song could not be found, even though you have a few.

What is the logic error in the `getIndexLastDiscount` algorithm? Remember, you can use a code walk-through to help you find the bug.

12. Fix the logic error in `getIndexLastDiscount`.

Part IV: Fix Logic Error 3

In Part IV, you will explore index values in an array and gain experience with what happens when the algorithm tries to access an index that is “out of bounds”.

13. Create your last, buggy method in `SongFinder`:

```
/**
 * Search through all songs, checking for blank titles.
 * If blank title is found, return -1 to indicate an error.
 * If all titles are "well-defined", return the last index in the array.
 */
public static int getIndexLastTitle(Song[] songs)
{
    int i;
    if (songs == null) {
       return -1;
    }
    for (i = 0; i < songs.length; i++) {
       // skip if no title
       if (songs[i].getTitle().equals("")) {
           return -1;
       }
    }
    return i;
}
```

14. Read the comments to understand the purpose of this method.

15. Invoke `getIndexLastTitle` method like you did with your other `SongFinder` methods.

```
System.out.println("--Find last song --");
index = SongFinder.getIndexLastTitle(topTenSongs);
if (index >= 0 ) {
System.out.println("Last title: " + topTenSongs[index].getTitle());
}
else {
System.out.println("You have a blank title!");
}
```

16 Run your program to test your algorithm.

There is a problem in `getIndexLastTitle`, it causes an "array index out of bounds" exception.

17 To find this type of problem, here are a few suggestions:

a. First, confirm that `i` is initialized correctly.

b. Next, check that the for loop uses `songs.length` properly.

c. Then, determine the actual value of array index that is out of bounds. What is the value?

18 Look at your `for` loop and try to determine why `i` is *one more than* the last index of your array.

In `getIndexLastTitle`, the condition statement in the for is `i < songs.length`. At the end of the iteration, this condition *must evaluate to false* so that the loop terminates. A walk-through can make this clearer:

i	result
0	0 is < 10, code in the for loop executes index++ executes
1	1 is < 10, code in the for loop executes index++ executes
2	2 is < 10, code in the for loop executes index++ executes
…	…
9	9 is < 10, code in the for loop executes index++ executes
10	10 is **not** < 10; code in for loop does not execute; the for loop terminates and execution resumes at the line after the for loop's closing }.

The indexing variable `i` ends up 1 greater than the last index of the array. This illustrates an important concept with for loops: *When a for loop visits every element of an array, at the termination of the loop, the value of its index variable is one greater than the last index of the array*.

19 Fix the bug. Be sure to define some test conditions and test your algorithm with them.

Part V: Bugs Mean Security Risks

Have you ever heard of a Zero Day vulnerability? It is a term used for a bug in a program that the author doesn't know about. Java, the JVM in particular, has had many bugs, and people are finding them all the time. In 2015 alone, users and programmers found 119 new vulnerabilities. But Java is not alone. QuickTime and iTunes also have bugs and vulnerabilities.

One Java bug led to a large breach called a Trojan horse. Similar to the Greek Trojan horse, users were invited to install friendly- or useful-looking software that actually contained harmful software called **malware**. The malware was discovered by an anti-virus program and the bug was fixed, but not before more than 600,000 computers were infected.

malware
Friendly- or useful-looking software that is actually harmful to you or your computer, your phone, mobile device, etc.

20 Research other types of malware and report on two of the types. Describe how the malware works and how users can prevent malware from harming their systems.

21 When the bugs are discovered or reported, companies issue a “patch” to fix the problem and users are supposed to install the fix. Many users do not.

a. Why do you think users do not install patches?

b. Why is it a bad idea not to install a patch?

Part VI: Common Array Algorithms – Highest and Lowest

Now that you have some practice with arrays (and their problems!), you will write some common array algorithms.

22 Create a new class in your MediaLib project called `Algorithms` and create a `main` method as you have done before.

23 Copy/reuse code from your `ArrayMediaLib` class that creates 10 songs in an array and initializes it. Modify the initialization list so that each song has another parameter for rating and rate the songs 1–10. Mix up the ratings so that songs are *not* rated in order 1–10. This will require yet another `Song` constructor, one that takes a `title` and a `rating` as its parameters.

24 Use a `for` or a for-each loop to display all songs with their rating, one song per line.

Consider how you might find your favorite song, whose rating would be #1, the lowest value. You would need to check each song’s rating and use a **best-so-far** variable to store the lowest value. You would iterate to check each song and determine whether a particular song’s rating is *better* than the best rating so far. This is demonstrated in the **pseudocode** below. Pseudocode is a form of documentation that is structured like a programming language but without the syntax of the language.

pseudocode
A documentation style that is structured like a programming language but does not use the syntax of the language.

```
declare and initialize best-so-far
for all songs in an array
    if this song's rating is lower than best-so-far
        best-so-far = this song's rating
```

Think about what might be a good initial value for `bestRating`. A very high number might be a good value, but users may choose a scale of 1 to 100, 1 to 1000, or whatever. Rather than trying to predict a specific value, you can initialize `bestRating` to the rating of the *first* song and then compare all other ratings to it.

25 Continuing in your `Algorithms` class, write the "find best" algorithm and print the best rated song in your array of songs.

26 Once you have found the best song, modify the algorithm to print the title.

27 Use the same loop to find the least favorite song and its title. You will need some new variables and choose a good initial value for the worst rating.

28 Which for loop (standard or enhanced) did you use? Why?

Part VII: Common Array Algorithms– Count Matches

Another common array algorithm is to count the number of items in an array that match a certain value. Explore this type of algorithm in the steps below.

29 In your initialization list, change your ratings so that you have a few ties for #2.

30 In a new for-each loop:

a. Count the number of songs that are tied for second.

a. Using a `String` variable and string concatenation, keep track of the song titles tied for second.

31 When the loop finishes, report the number of #2 songs and their titles. Why was a for-each loop recommended for this step (as opposed to a standard `for` loop)?

Part VIII: Common Array Algorithms – Delete and Insert

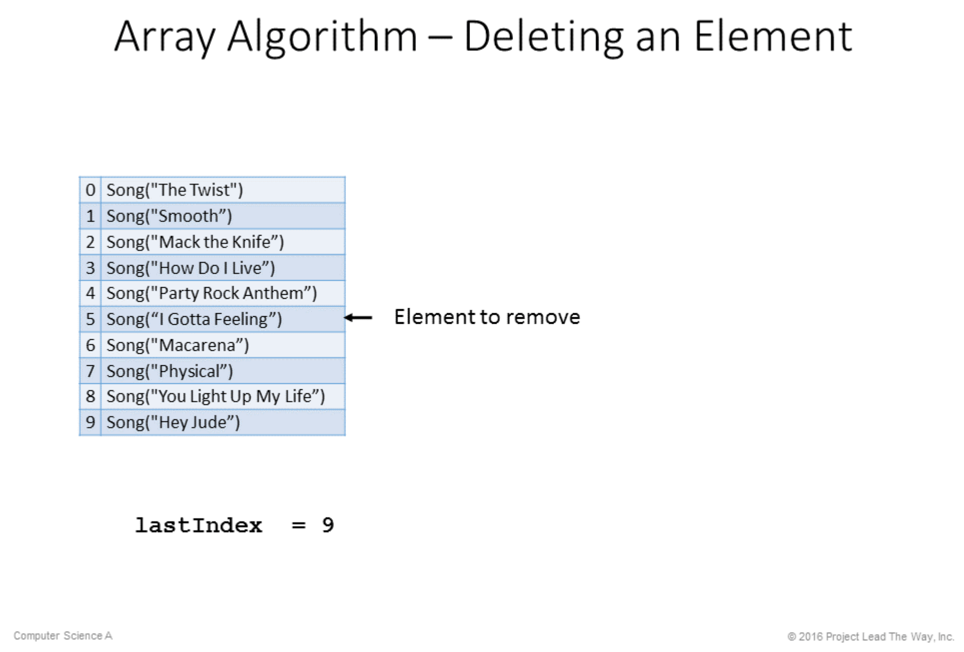
Consider how you would go about deleting and inserting elements in an array. First, remember that when you create an array, it has a *fixed size*; you cannot change it. The values in an array can be null, or the empty string, or zero, but the array elements still exist. This means, an array can be **partially filled**.

For example, you have created your songs array of 10 songs, but later decide to delete the sixth song. Your array might end up looking like:

index	contents
0	`Song ("The Twist")`
1	`Song ("Smooth")`
2	`Song ("Mack the Knife")`
3	`Song ("How Do I Live")`
4	`Song ("Party Rock Anthem")`
5	`null`
6	`Song ("Macarena")`
7	`Song ("Physical")`
8	`Song ("You Light Up My Life")`
9	`Song ("Hey Jude")`

This is considered partially filled; it seems as if the array has a hole or a gap at index 5. If you were to iterate over this array, the null entry could cause a problem. So, to delete elements in an array, you must rearrange them so there are no gaps or holes.

32 Review the slideshow on deleting an array element.



Array Algorithm – Deleting an Element

0	Song("The Twist")
1	Song("Smooth")
2	Song("Mack the Knife")
3	Song("How Do I Live")
4	Song("Party Rock Anthem")
5	Song("I Gotta Feeling")
6	Song("Macarena")
7	Song("Physical")
8	Song("You Light Up My Life")
9	Song("Hey Jude")

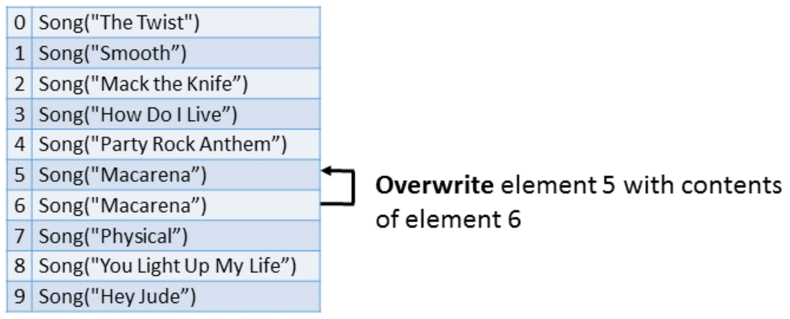
Element to remove

```
lastIndex  = 9
```

Computer Science A

© 2016 Project Lead The Way, Inc.

Array Algorithm – Deleting an Element



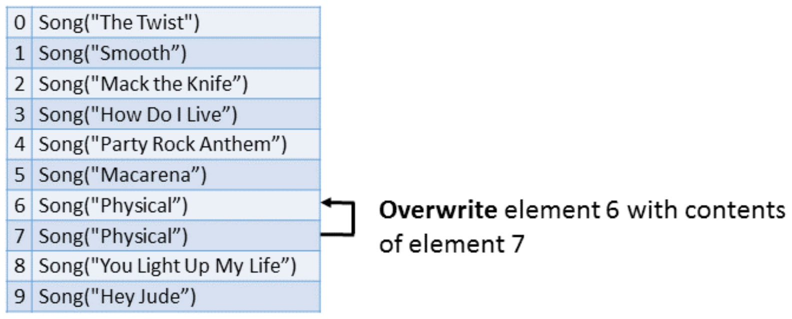
0	Song("The Twist")
1	Song("Smooth")
2	Song("Mack the Knife")
3	Song("How Do I Live")
4	Song("Party Rock Anthem")
5	Song("Macarena")
6	Song("Macarena")
7	Song("Physical")
8	Song("You Light Up My Life")
9	Song("Hey Jude")

Overwrite element 5 with contents of element 6

```
lastIndex  = 9
```

Computer Science A

Array Algorithm – Deleting an Element



0	Song("The Twist")
1	Song("Smooth")
2	Song("Mack the Knife")
3	Song("How Do I Live")
4	Song("Party Rock Anthem")
5	Song("Macarena")
6	Song("Physical")
7	Song("Physical")
8	Song("You Light Up My Life")
9	Song("Hey Jude")

Overwrite element 6 with contents of element 7

```
lastIndex  = 9
```

Computer Science A

Array Algorithm – Deleting an Element

0	Song("The Twist")
1	Song("Smooth”)
2	Song("Mack the Knife”)
3	Song("How Do I Live”)
4	Song("Party Rock Anthem”)
5	Song("Macarena”)
6	Song("Physical”)
7	Song("You Light Up My Life”)
8	Song("You Light Up My Life”)
9	Song("Hey Jude”)

Overwrite element 7 with contents of element 8

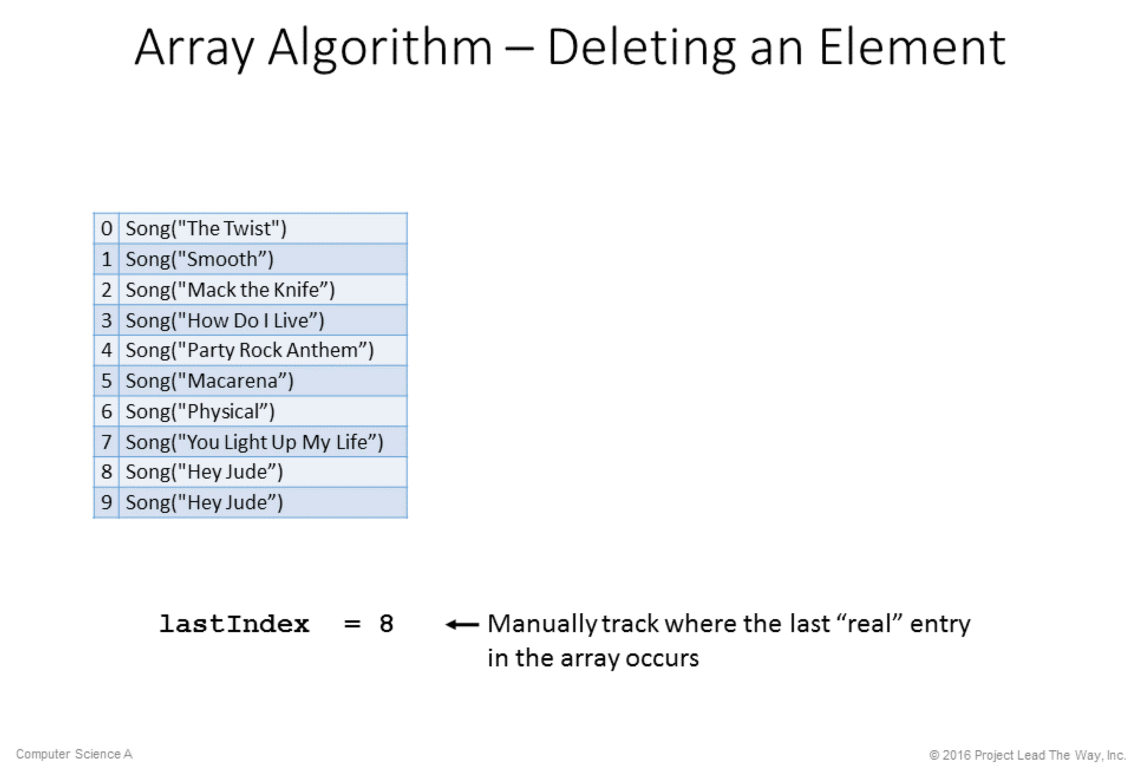
```
lastIndex  = 9
```

Array Algorithm – Deleting an Element

0	Song("The Twist")
1	Song("Smooth”)
2	Song("Mack the Knife”)
3	Song("How Do I Live”)
4	Song("Party Rock Anthem”)
5	Song("Macarena”)
6	Song("Physical”)
7	Song("You Light Up My Life”)
8	Song("Hey Jude”)
9	Song("Hey Jude”)

Overwrite element 8 with contents of element 9

```
lastIndex  = 9
```



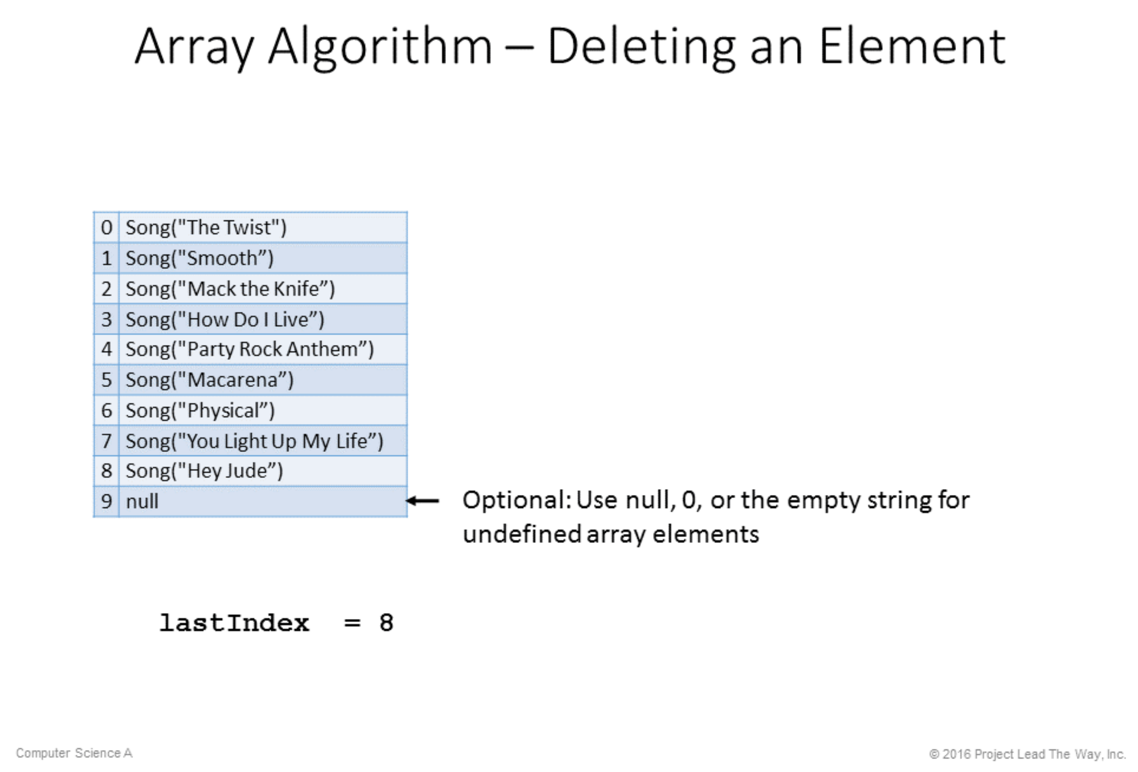
Array Algorithm – Deleting an Element

0	Song("The Twist")
1	Song("Smooth")
2	Song("Mack the Knife")
3	Song("How Do I Live")
4	Song("Party Rock Anthem")
5	Song("Macarena")
6	Song("Physical")
7	Song("You Light Up My Life")
8	Song("Hey Jude")
9	Song("Hey Jude")

```
lastIndex  = 8
```

← Manually track where the last "real" entry in the array occurs

Computer Science A



Array Algorithm – Deleting an Element

0	Song("The Twist")
1	Song("Smooth")
2	Song("Mack the Knife")
3	Song("How Do I Live")
4	Song("Party Rock Anthem")
5	Song("Macarena")
6	Song("Physical")
7	Song("You Light Up My Life")
8	Song("Hey Jude")
9	null

← Optional: Use null, 0, or the empty string for undefined array elements

```
lastIndex  = 8
```

Computer Science A

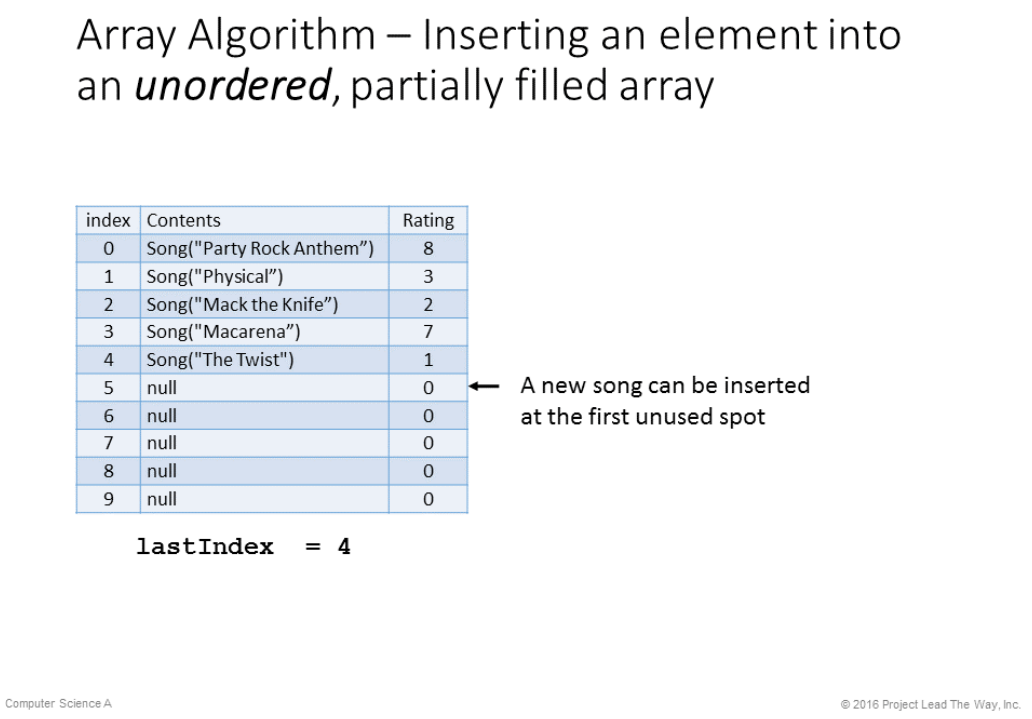
33 Continuing in the MediaLib `Algorithms` class, create a for loop to iterate over the length of your songs array. Choose a song title that you will delete from your array, preferably from somewhere in the middle, and use the `equals` method to find that song title.

34 Copy the next song in the array to the current location. Continue iterating, replacing the next song into the current location until you reach the end of the array.

35 Write a new for loop to display the contents of your new, shorter song list. Remember, you will need to reference the variable that has the index of the last defined element in your partially filled array.

Inserting into a partially filled array reverses the deletion process.

36 Review the slideshow on inserting an array element.



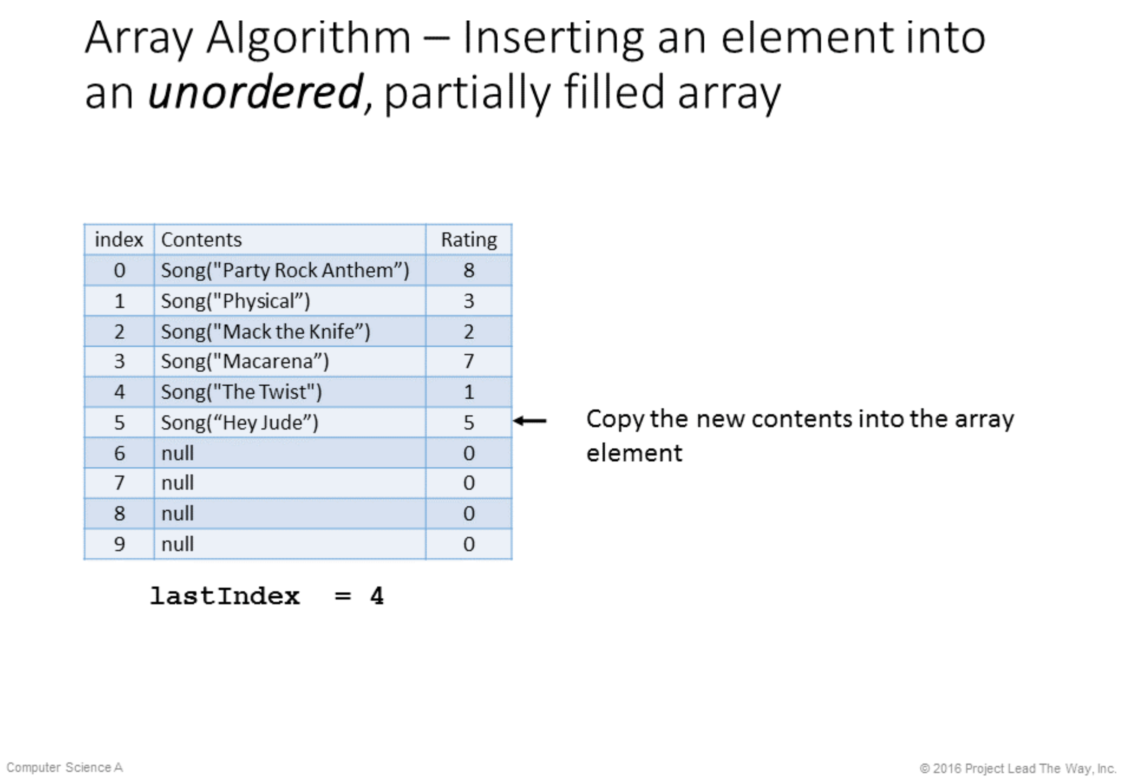
Array Algorithm – Inserting an element into an *unordered*, partially filled array

index	Contents	Rating
0	Song("Party Rock Anthem")	8
1	Song("Physical")	3
2	Song("Mack the Knife")	2
3	Song("Macarena")	7
4	Song("The Twist")	1
5	null	0
6	null	0
7	null	0
8	null	0
9	null	0

A new song can be inserted at the first unused spot

```
lastIndex  = 4
```

Computer Science A © 2016 Project Lead The Way, Inc.



Array Algorithm – Inserting an element into an *unordered*, partially filled array

index	Contents	Rating
0	Song("Party Rock Anthem")	8
1	Song("Physical")	3
2	Song("Mack the Knife")	2
3	Song("Macarena")	7
4	Song("The Twist")	1
5	Song("Hey Jude")	5
6	null	0
7	null	0
8	null	0
9	null	0

Copy the new contents into the array element

```
lastIndex  = 4
```

Computer Science A © 2016 Project Lead The Way, Inc.

Array Algorithm – Inserting an element into an *unordered*, partially filled array

index	Contents	Rating
0	Song("Party Rock Anthem")	8
1	Song("Physical")	3
2	Song("Mack the Knife")	2
3	Song("Macarena")	7
4	Song("The Twist")	1
5	Song("Hey Jude")	5
6	null	0
7	null	0
8	null	0
9	null	0

`lastIndex = 5`

Manually track where the last "real" entry in the array occurs

Array Algorithm – Inserting an element into an *ordered*, partially filled array

index	Contents	Rating
0	Song("Party Rock Anthem")	1
1	Song("Physical")	4
2	Song("Mack the Knife")	6
3	Song("Macarena")	8
4	Song("The Twist")	9
5	null	0
6	null	0
7	null	0
8	null	0
9	null	0

← A new song has a rank of 5, it should be inserted at index 2

`lastIndex = 4`

Array Algorithm – Inserting an Element

index	Contents	Rating
0	Song("Party Rock Anthem")	1
1	Song("Physical")	4
2	Song("Mack the Knife")	6
3	Song("Macarena")	8
4	Song("The Twist")	9
5	Song("The Twist")	9
6	null	0
7	null	0
8	null	0
9	null	0

Overwrite element 5 with contents of element 4

`lastIndex = 4`

Array Algorithm – Inserting an Element

index	Contents	Rating
0	Song("Party Rock Anthem")	1
1	Song("Physical")	4
2	Song("Mack the Knife")	6
3	Song("Macarena")	8
4	Song("Macarena")	8
5	Song("The Twist")	9
6	null	0
7	null	0
8	null	0
9	null	0

Overwrite element 4 with contents of element 3

`lastIndex = 4`

Array Algorithm – Inserting an Element

index	Contents	Rating
0	Song("Party Rock Anthem”)	1
1	Song("Physical”)	4
2	Song("Mack the Knife”)	6
3	Song("Mack the Knife”)	6
4	Song("Macarena”)	8
5	Song("The Twist“)	9
6	null	0
7	null	0
8	null	0
9	null	0

Overwrite element 3 with contents of element 2

```
lastIndex  = 4
```

Computer Science A

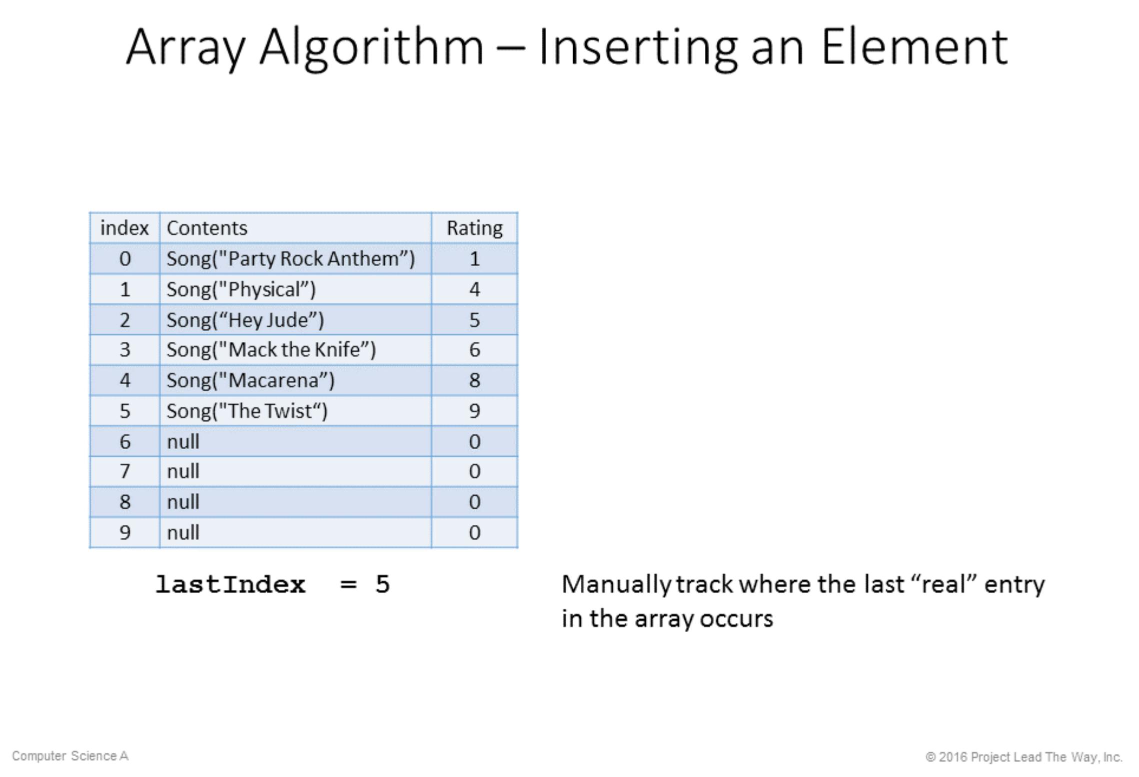
Array Algorithm – Inserting an Element

index	Contents	Rating
0	Song("Party Rock Anthem”)	1
1	Song("Physical”)	4
2	Song(“Hey Jude”)	5
3	Song("Mack the Knife”)	6
4	Song("Macarena”)	8
5	Song("The Twist“)	9
6	null	0
7	null	0
8	null	0
9	null	0

Overwrite element 2 with the new contents

```
lastIndex  = 4
```

Computer Science A



Array Algorithm – Inserting an Element

index	Contents	Rating
0	Song("Party Rock Anthem")	1
1	Song("Physical")	4
2	Song("Hey Jude")	5
3	Song("Mack the Knife")	6
4	Song("Macarena")	8
5	Song("The Twist")	9
6	null	0
7	null	0
8	null	0
9	null	0

```
lastIndex  = 5
```

Manually track where the last “real” entry in the array occurs

Computer Science A

37 Write an algorithm to insert a new song into a partially filled array of songs.

Part IX: Storing Preferences

In many Android apps, users often want to store some information about how they like to use their app. The most common and simple way to do this is with preferences. The **official documentation on preferences** will tell you all you need to know about preferences.

38 In this part of the activity, you will add a preference to the MediaLib app that you worked on in the last activity. Open your Android Studio MediaLib project. If you have not yet launched Android Studio or if you did not create a MediaLib project, complete the following:

a. In *Activity 1.1.1 Introduction to Android Studio*; Part III.

b. In *Activity 1.1.2 Your First Class*; Parts V, VI, and VII.

c. In *Activity 1.1.3 Making Objects*; Part IV and V.

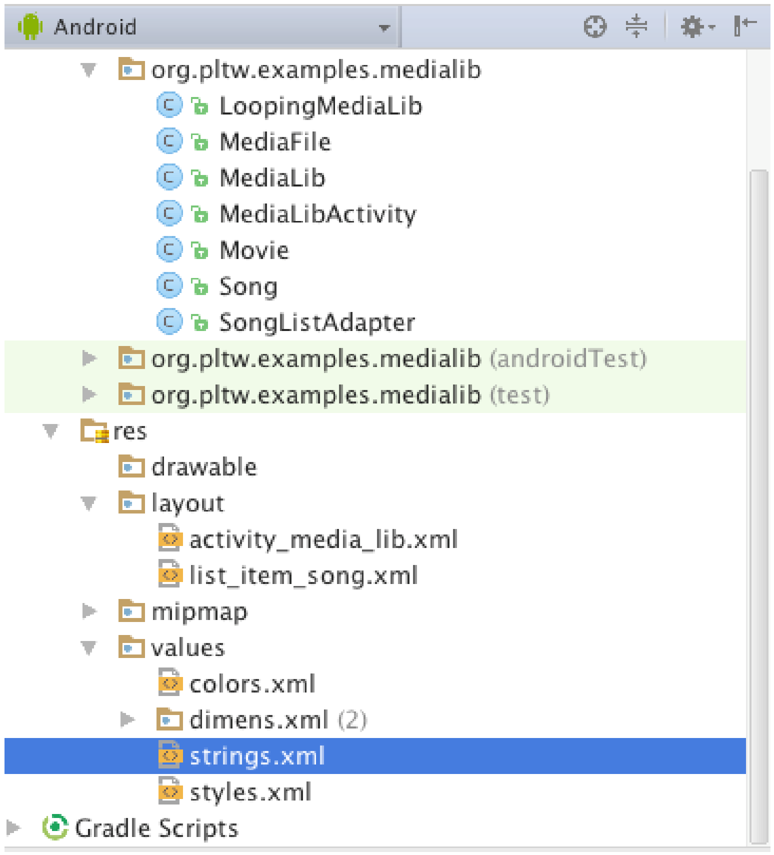
d. In *Activity 1.2.2 Today’s Top 40*; Part VI.

39 In your own words, describe when preferences should be used.

NOTE

strings.xml is a very important file in Android development, not only for preference management but in many other situations, as you'll see later in the course.

40 For now, open your `strings.xml` file found in your project structure.



```
org.pltw.examples.medialib
    LoopingMediaLib
    MediaFile
    MediaLib
    MediaLibActivity
    Movie
    Song
    SongListAdapter
org.pltw.examples.medialib (androidTest)
org.pltw.examples.medialib (test)
res
  drawable
  layout
    activity_media_lib.xml
    list_item_song.xml
  mipmap
  values
    colors.xml
    dimens.xml (2)
    strings.xml
    styles.xml
Gradle Scripts
```

41 Using the `app_name` string resource as an example, create another string resource in between the open and closed `resources` tags identified by `every_other_string`, with value `org.pltw.examples.every_other_string`.

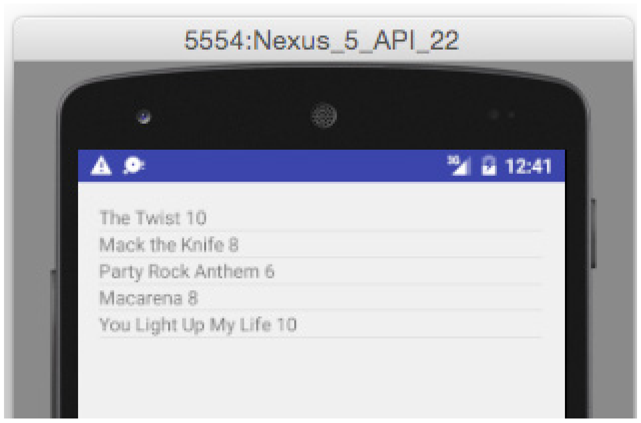
This string value will be used as a key to access the value of your preference in the shared preferences file.

42 Enter the following code immediately after the call to `setContentView` in `MediaLibActivity`.

```
Context context = this;
SharedPreferences sharedPreferences = this.getPreferences
(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putBoolean(getString(R.string.every_other_string),
everyOtherSong);
editor.commit();
```

43 Add a `private` instance variable of type `boolean`, identified by `everyOtherSong` to the `MediaLibActivity`, and set its value to `true`.

44 Use the **official documentation on preferences** and your `MediaLibActivity` code to change the functionality of the app, so that if the preference value for the key `every_other_string` is `true`, the `ArrayList songLibrary` will only contain `Song` objects from even-numbered indices in the `songs` array. Your finished solution should look like this:



```
The Twist 10
Mack the Knife 8
Party Rock Anthem 6
Macarena 8
You Light Up My Life 10
```

CONCLUSION

1. List four pieces of data that you believe might be stored in `SharedPreferences` in apps that you use, or four pieces of data that you would want to store in `SharedPreferences` in your own app.