09-0: **Array Review**

- Arrays store a list of objects or primitive types
- Fixed size (can’t change size of an array)
- Get the length of the array, get at elements of an array

09-1: **Array Review**

- On board: What memory looks like for the following:

```java
public static void main(String args)
{
    int A[];
    int x;
    String B[];  <-- Show memory here
    A = new int[5];
    B = new String[3];  <-- Show memory here
}
```

09-2: **Array Review**

```java
int x;
String A[];
A = new String[5];
x = A[0].length();  <-- What happens here?
```

09-3: **Array Review**

```java
int x;
String A[];
A = new String[5];
x = A[0].length();  <-- Null Pointer Exception
```

09-4: **Array Review**

```java
int A[] = new int[10];
for (int i = 0; i < A.length; i++)
{
    A[i] = i;
}
```

- Anywhere we could use an integer variable, we can use `A[i]`, for any integer expression `i`
  - `A[i+2] = 4`
  - `foo(A[x+y])`

- Of course, we will get a run-time exception if the value of the expression is not between 0 and `A.length - 1`
09-5: **2D Arrays**

- We can create 2D arrays as well as 1D arrays
  - Like matrices
- 2D array is really just an array of arrays

09-6: **2D Arrays**

```java
int x[][]; // Declare a 2D array
int[][] y; // Alternate way to declare 2D array

x = new int[5][10]; // Create 50 spaces
y = new int[4][4]; // Create 16 spaces
```

09-7: **2D Arrays**

```java
int x[][]; // Declare a 2D array
x = new int[5][5]; // Create 25 spaces

x[2][3] = 11;
x[3][3] = 2;
x[4][5] = 7; // ERROR! Index out of bounds
```

09-8: **2D Arrays**

- How would we create a 9x9 array, and set every value in it to be 3?

```java
int board[][];
board = new int[9][9];
for (int i = 0; i < 9; i++)
    for (int j = 0; j < 9; j++)
        board[i][j] = 3;
```

- Show memory!

09-9: **2D Arrays**

- How would we create a 9x9 array, and set every value in it to be 3?

```java
int intMatrix[][];
intMatrix = new int[5][];
for (int i = 0; i < 5; i++)
{
    intMatrix[i] = new int[5];
}
```

09-10: **2D Arrays**

```java
int intMatrix[][];
intMatrix = new int[5][];
for (int i = 0; i < 5; i++)
{
    intMatrix[i] = new int[5];
}
```
int intMatrix[][];
intMatrix = new int[5];
for (int i = 0; i < 5; i++)
{
    intMatrix[i] = new int[i];
}

09-12: Using Arrays

- Need to declare array size before using them
- Don’t always know ahead of time how big our array needs to be
- Allocate more space than we need at first
- Maintain a second size variable, that has the number of elements in the array we actually care about
- Classes that use arrays often will have an array instance variable, and a size instance variable (how much of the array is used)

09-13: Using Arrays

```java
public class StringArrayList
{
    String data[];
    int listSize;

    public StringArrayList()
    {
        data = new String[10];
        listSize = 0;
    }

    // other methods
}
```

09-14: Using Arrays

```java
public class StringArrayList
{
    String data[];
    int listSize;

    int size()
    {
        // Fill me in!
    }

    // other methods
}
```

09-15: Using Arrays

```java
public class StringArrayList
{
    String data[];
    int listSize;

    int size()
    {
        return listSize;
    }

    // other methods
}
```

09-16: Using Arrays

```java
public class StringArrayList
{
    String data[];
    int listSize;

    void add(String newString)
    {
        // Fill me in!
    }

    // other methods
}
```

09-17: Using Arrays
public class StringArrayList
{
    String data[];
    int listSize;
    void add(String newString)
    {
        data[listSize] = newString;
        listSize++;
    }
    // other methods
}

• What happens when someone tries to add 11 strings to this StringArrayList?

09-18: Using Arrays

public class StringArrayList
{
    String data[];
    int listSize;
    void add(int index, String newString)
    {
        // Fill me in!
    }
    // other methods
}

09-19: Using Arrays

public class StringArrayList
{
    String data[];
    int listSize;
    void add(int index, String newString)
    {
        for (int i = listSize; i > index; i--)
        {
            data[i] = data[i-1];
        }
        data[index] = newString;
    }
    // other methods
}

09-20: Static

• Normally, can only call methods on classes when we created an instance of the class
  • Methods can rely on instance variables to work properly
  • Need to create an instance of a class before there are any instance variables
  • What would the size() method return for an ArrayList if there was not an instance to check the size of?

09-21: Static

• Some methods don’t operate on an instance of the class – pure functions that don’t use instance variables at all
  • Math functions like min, or pow
  • parseInt – takes a string as an input parameter, and returns the integer value of the string parseInt("123") returns 123
  • Seems silly to have to instantiate an object to use these methods
  • static to the rescue!

09-22: Static

• If we declare a method as static, it does not rely on an instance of the class
  • Can call the method without creating an instance first
• Use the Class Name to invoke (call) the method

double x = Math.min(3.4, 6.2);
double z = Math.sqrt(x);

09-23: Constants

• Having 3.14159 appearing all over your code is considered bad style
  • Could end up using different values for pi in different places (3.14159 vs. 3.1415926)
  • If you want to change the value of pi (to add more digits, for instance), need to search through all of your code to find it
• In general, any time you have a “magic number” (that is, an arbitrary numeric literal) in your code, it should probably be a symbolic constant instead.
• The “final” modifier is used to prevent you from changing the value of a variable

09-24: Constants

class Calendar
{
    final int MONTHS_IN_YEAR = 12;
    final int DAYS_IN_WEEK = 12;
    final int DAYS_IN_YEAR = 365;

    // Methods that use the above constants
}

• Every instance of class Calendar will contain those 3 variables
• Somewhat wasteful
• Need to instantiate an object of type Calendar to access them

09-25: Constants: Static

• We can declare variables to be static as well as methods
• Typically used for constants (Math.pi, Math.e)
• Access them using class name, not instance name (just like static methods)
• You should only use static variables for constants
  • public static final float pi = 3.14159;