

Introduction to Programming

with Java, for Beginners

Arrays of Primitives

What if our Frog (from lab) could say 10 different things?

```
public class Frog{
    private boolean formerPrince;
    private String phrase1;
    private String phrase2;
    private String phrase3;
    private String phrase4;
    private String phrase5;
    private String phrase6;
    private String phrase7;
    private String phrase8;
    private String phrase9;
    private String phrase10;
    . . .
}
```

ESE112

1/17

What a Person could adopt lots of Pets ?

```
public class Person{
    private String name;
    private Pet pet1;
    private Pet pet2;
    private Pet pet3;
    private Pet pet4;
    private Pet pet5;
    private Pet pet5;
    private Pet pet6;
    private Pet pet6;
    private Pet pet8;
    private Pet pet9;
    private Pet pet10;
    private Pet pet11;
    private Pet pet12;
    private Pet pet13;
    private Pet pet14;
    private Pet pet15;
    . . .
}
```

ESE112

2/17

What if we wanted to store 500 Itoons?

```
public class IToonList{
    private IToon toon1;
    private IToon toon2;
    private IToon toon3;
    private IToon toon4;
    private IToon toon5;
    private IToon toon6;
    private IToon toon7;
    private IToon toon8;
    private IToon toon9;
    private IToon toon10;
    private IToon toon11;
    private IToon toon12;
    private IToon toon13;
    . . .
    private IToon toon500;
}
```

ESE112

3/17

What if we want to store lots of things...

- But we don't want to declare a separate variable for each one?
- That's what *arrays* are good for

ESE112

4/17

What is an Array ?

- It's an easy way to declare lots of variables that all have the *same type*

```
//E.g. declare an array of integers  
int[]data = new int[5]; //total ints = 5
```

- When an array of particular primitive type is created, Java initializes the elements to the types default value
 - E.g. Array of ints – default value is zero

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|

ESE112

5/17

Array Elements and Indices

- //initializing integer array

```
data[0] = 6;  
data[1]= 10;  
data[2] = 12;
```

| | | | | |
|---|----|----|---|---|
| 6 | 10 | 12 | 0 | 0 |
|---|----|----|---|---|

- The number within square brackets is called an *index*
- The valid *indices* are 0 thru (array length - 1)
- NOTE: Whenever you see square brackets [] in a Java program, it means you're dealing with an array

ESE112

6/17

An Array is an Object

```
int[] data;
```

data is a reference variable whose *type* is *int[]*, meaning "array of ints". At this point its value is null.

```
data = new int[5];
```

The *new* operator causes a chunk of memory big enough for 5 ints to be allocated on the heap. Here, *data* is assigned a reference to the heap address.

```
data[0] = 6;  
data[1]= 10;  
data[2] = 12;
```

Initially, all five ints are 0. Here, three of them are assigned other values.

```
int[] info = {6, 10, 12, 0, 0};
```

```
info = new int[]{6, 10, 12, 0, 0};
```

ESE112

7/17

Using Array Elements in Expressions

- An *element* of an array of ints can be used virtually anywhere an expression of type int is valid.
- Likewise for arrays of other types

```
int[] data = new int[] {6, 10, 12, 0, 0};
int x = data[0];
data[3] = data[2];
data[4] = data[3] + data[2] * 2;
System.out.println("data[0] is" + data[0]);

// note order of operations below:
data[4] = Math.pow(2, data[4]);
```

ESE112

8/17

Accessing an Array's Length

- `ArrayName.length` gives size of the array

```
int[] data;
data = new int[5];    // data.length is 5
data[0] = 6;
data[1] = 10;
data[2] = 12;

//Summing the contents of an array
int result = 0;
for (int i =0; i < data.length; i++){
    result = result + data[i];
}
```

ESE112

9/17

Complete the sum(..) method

```
public class ArrayTool{

    /**
     * Takes an array of ints as an argument.
     * returns the sum of all the integers in the array.
     */
    public static int sum ( ?? ) {

    }

}
```

```
Welcome to DrJava
> int[] data = new int[] {6, 10, 12, 0, 0};
> ArrayTool.sum(data)
28
```

ESE112

10/17

Array Out of Bounds Exceptions

```
public class ArrayTool{

    public static int sum(int[] data){
        int sum = 0;
        for (int i = 0; i < data.length; i++){
            sum = sum + data[i]; // sum += data[i];
        }
        return sum;
    }

    public static int sum2(int[] data){
        int sum = 0;
        for (int i = 0; i <= data.length; i++){ // error!
            sum = sum + data[i]; // sum += data[i];
        }
        return sum;
    }
}

> ArrayTool tool = new ArrayTool();
> int[] data = new int[] {6, 10, 12, 0, 0};
> tool.sum(data)
28
> tool.sum2(data)
ArrayIndexOutOfBoundsException:
```

ESE112

11/17

Examples of Arrays of Other Primitive Types

```
double[] temps;  
temps = new double[24];  
temps[0] = 78.5;  
temps[1] = 84.2;  
  
boolean[] answers = new boolean[6];  
. . .  
if (answers[0]){ //do something }  
  
char[] buffer = new char[50];
```

ESE112

12/17

Alternative way Declaring & Initializing Arrays

```
int[] info1 = { 2000, 100, 40, 60};  
int[] info2 = new int[] { 2000, 100, 40, 60};  
  
char[] choices1 = { 'p', 's', 'q'};  
char[] choices2 = new char[] { 'p', 's', 'q'};  
  
double[] temps1 = {75.6, 99.4, 86.7};  
double[] temps2 = new double[] {75.6, 99.4, 86.7};
```

ESE112

13/17

Complete this method

```
public class ArrayTool{  
  
    /* Returns true if all integers in the  
    data array are positive, false otherwise.  
    */  
    public static boolean allPositive(int[] data){  
  
    }  
}
```

ESE112

14/17

One Solution

```
public class ArrayTool{  
  
    /* Returns true if all integers in the data array  
    * are positive, false otherwise.  
    */  
    public static boolean allPositive(int[] data){  
        for (int i = 0; i < data.length; i++){  
            if (data[i] <= 0) {  
                return false;  
            }  
        }  
        return true;  
    }  
}
```

ESE112

15/17

Another Solution

```
public class ArrayTool{

    /* Returns true if all integers in the data
       array are positive, false otherwise.
    */
    public static boolean allPositive(int[] data){
        boolean result = true;
        for (int i = 0; i < data.length; i++){
            if (data[i] <= 0) {
                result = false;
                break; //break is java keyword
            }
        }
        return result;
    }
}
```

ESE112

16/17

break statement

- From example on the previous slide:
 - Terminates the for loop if the value found is negative
- Used only with loop structures
- Rule: Terminates the innermost for/while loop

ESE112

17/17