

Objectives

- Last of Java fundamentals
 - Arrays wrap up
 - Indefinite loops
 - For-each loops
 - Switch statements
- Static methods and fields

Sept 20, 2021

Sprenkle - CSCI209

1

1

Review

- What is the difference between **declaring**, **initializing**, and **defining** a variable?
- If you need to use another Java class in your code (and it's not part of java.lang), what do you need to do? What is the syntax to do that?
- What is the syntax of a **for** loop?
- What is the scope of a variable?
- What are the logical operators (and, or, not) in Java?
- How do we access command-line arguments from a Java program execution?
- Arrays:
 - How do we declare an array of elements?
 - How do we access elements of an array?
 - How can we find out the size of an array?

Sept 20, 2021

Sprenkle - CSCI209

2

2

Terminology Review

- Declaration: `int x;`
- Definition: `x = 3;`
- Initializing: typically the first time the variable is given a value
 - `int x = 3;`
 - Or could be first assignment, e.g., `x = 3;`

Sept 20, 2021

Sprenkle - CSCI209

3

3

Review: Array Length

- All array variables have a *field* called `length`
 - Note: no parentheses because not a method

```
int[] array = new int[10];
for (int i = 0; i < array.length; i++) {
    array[i] = i * 2;
}

for (int i = array.length-1; i >= 0; i--) {
    System.out.println(array[i]);
}
```

I'm declaring `i` twice in this code. Why is that not a compiler error?

Sept 20, 2021

Sprenkle - CSCI209 `ArrayLength.java`

4

4

Review: Variable Scope

- All array variables have a **field** called `length`

➤ Note: no parentheses because not a method

```
int[] array = new int[10];
for (int i = 0; i < array.length; i++) {
    array[i] = i * 2;
}

for (int i = array.length-1; i >= 0; i--) {
    System.out.println(array[i]);
}
```

Scope of `i` is within each for loop; other loop doesn't see it.
Declaring counter variable in for loop is common practice.

Sept 20, 2021

5

5

Example FileExtensionFinder

```
/**
 * This Java program (FileExtensionFinder) takes a file name (a
 * String) as user input and displays the file extension, lowercased.
 *
 * @author Redacted McRedacted
 */
public class FileExtensionFinder {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your filename: ");
        String filename = sc.nextLine();
        sc.close();

        int periodIndex = filename.lastIndexOf('.');
        String extension = filename.substring(periodIndex + 1);
        String lcExtension = extension.toLowerCase();

        System.out.println("Your file is a(n) " + lcExtension + " file.");
    }
}
```

- Good variable names
- Good chunks – not doing too much in one line
- Good high-level comment

Sept 17, 2021

6

6

Arrays

- Assigning one array variable to another → both variables refer to the same array

➤ Similar to Python

- Draw picture of below code:

```
int [] fibNums = {1, 1, 2, 3, 5, 8, 13};
int [] otherFibNums;

otherFibNums = fibNums;
otherFibNums[2] = 99;

System.out.println(otherFibNums[2]);
System.out.println(fibNums[2]);
```

Sept 20, 202

7

7

Arrays

- Assigning one array variable to another → both variables refer to the same array

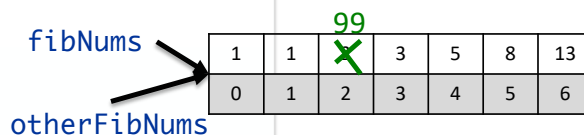
➤ Similar to Python

- Draw picture of below code:

```
int [] fibNums = {1, 1, 2, 3, 5, 8, 13};
int [] otherFibNums;

otherFibNums = fibNums;
otherFibNums[2] = 99;

System.out.println(otherFibNums[2]);
System.out.println(fibNums[2]);
```



Displays:
99
99

Sept 20, 2021

Sprenkle - CSCI209

8

8

java.util.Arrays

- `Arrays` is a class in `java.util`
- Static methods for sorting, searching, `deepEquals`, fill arrays
- To use class, need `import` statement
 - Goes at top of program, before class definition

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

`ArraysExample.java`

Sept 20, 2021

Sprenkle - CSCI209

9

9

Danger of a Large Library

- Be careful when searching for classes
- Lots of classes seem like they're what we want but aren't, e.g.,

```
java.lang.reflect.Array
javax.sql.rowset.serial.Array
```

An array (e.g., `int[]` array) is **not** an instance of a class so we cannot call methods on it.

Sept 20, 2021

Sprenkle - CSCI209

10

10

MORE CONTROL STRUCTURES

Sept 20, 2021

Sprenkle - CSCI209

11

11

Control Flow: foreach Loop

- Sun called “enhanced for” loop
- Iterate over all elements in an array (or Collection)

➤ Similar to Python’s `for` loop

```
int[] a;
int result = 0;
. . .
for (int i : a) {
    result += i;
}
```

<https://docs.oracle.com/javase/8/docs/technotes/guides/language/foreach.html>

for each int element `i` in the array `a`,
the loop body is executed

Sept 20, 2021

Sprenkle - CSCI209

12

12

Control Flow: **while** Loops

- **while** loop

- Condition must be enclosed in parentheses
- Body of loop must be enclosed in `{ }` if multiple statements

```
int counter = 0;
while (counter < 5) {
    System.out.println(counter);
    counter++; ← shortcut
}
System.out.println("Done: " + counter);
```

Sept 20, 2021

Sprenkle - CSCI209

Counter.java


13

13

Changing control flow: **break**

- Exits the current loop

```
while ( <condition> ) {
    ...
    if( <something> ) { // now we're done!
        break;
    }
}
```



Sept 20, 2021

Sprenkle - CSCI209

14

14

Control Flow: **switch** statement

- Like a big **if/else if** statement
- Works with variables with datatypes **byte**, **short**, **char**, **int**, and **String**

```
int x = 3;
switch(x) {
    case 1:
        System.out.println("It's a 1.");
        break;
    case 2:
        System.out.println("It's a 2.");
        break;
    default:
        System.out.println("Not a 1 or 2.");
}
```

Sept 20, 202

15

15

Control Flow: **switch** statement

```
switch(grade) {
    case 'a':
    case 'A':
        System.out.println("Congrats!");
        break;
    case 'b':
    case 'B':
        System.out.println("Not too shabby!");
        break;
    ... // Handle c, d, and f ...
    default:
        System.out.println("Error: not a grade");
}
```

Sept 20, 2021

Sprenkle - CSCI209

Grades.java

16

16

Summary: Python to Java Gotchas

- Every variable needs to be declared with its data type before it is used
- Scope of variables
- Need to use equals method to do more than just a “same object” check
- Syntax
 - Semicolons at the end of **statements**
 - Braces around blocks of code
 - Keywords
- Need to (1) compile and then (2) execute program

Sept 20, 2021

Sprenkle - CSCI209

17

17

Benefits of Static Typing

- Easier to keep track of type of variable
 - Know operations that can be executed on a variable of a certain type
- Compiler can check that you're only using valid operations for this type
- More benefits later this semester

Sept 20, 2021

Sprenkle - CSCI209

18

18

STATIC METHODS AND FIELDS

Sept 20, 2021

Sprenkle - CSCI209

19

19

static Methods/Fields

- For functionality/data that is specific to a *class*
 - And is **not** specific to a particular object

Sept 20, 2021

Sprenkle - CSCI209

20

20

static Methods/Fields Case Study: java.lang.Math

- No constructor (what does that mean?)
- Static fields: PI, E
 - To refer to field: `ClassName.field`
 - Example: `Math.PI`
- Static methods:
 - `static double sin(double a)`
 - To call method: `ClassName.methodName(...)`
 - Example: `Math.sin(number);`


Sept 20, 2021

Sprenkle - CSCI209

21

21

Static Methods `ClassName.method(...)`

- Do **not** operate on objects 
 - i.e., you do **not** call `object.staticMethod()`;
 - **Cannot** access *instance* fields of their class
- Can access *static fields* of their class
 - Example: `Math` class could have a static method that uses `PI`
- Similar to Python *functions* that are associated with the class

Sept 20, 2021

Sprenkle - CSCI209

22

22

Analyzing java.lang.String API

- Consider a “typical” (non-static) method:
`String toUpperCase()`
 - Converts all of the characters in *this String* to upper case
 - Example use: 1) create a string
2) call `myString.toUpperCase()`

Sept 20, 2021

Sprenkle - CSCI209

23

23

Analyzing java.lang.String API

- `String toUpperCase()`
 - Converts all of the characters in *this String* to upper case
 - Example: create a string, call `myString.toUpperCase()`
- `static String valueOf(boolean b)`
 - Returns the string representation of the `boolean` argument
 - Example use: `String.valueOf(false);`

Sept 20, 2021

Why can/should the second method be `static`?

24

24

Discussion

Why is main static?

Sept 20, 2021

Sprenkle - CSCI209

25

25

main()

- Most common **static** method
- `main()` does not get called on an object
 - Runs when a program starts
 - There are no objects yet but there is the class
- `main()` executes and constructs the objects the program needs and will use
 - Like the **driver function** for the program

Sept 20, 2021

Sprenkle - CSCI209

26

26

JavaDocs for Methods

```
/**
 * Returns the string representation of the boolean argument.
 *
 * @param b - a boolean
 * @return if the argument is true, a string equal to "true" is
 *         returned; otherwise, a string equal to "false" is
 *         returned.
 */
public static boolean valueOf(boolean b) {
```

- Use format similar to class comments
- Use **@param** tag(s) to describe what method takes as parameter(s)
- Use **@return** tag to describe what method returns

Sept 20, 2021

Sprenkle - CSC1209

27

27

JavaDocs for Methods

```
/**
 * Returns the string representation of the boolean argument.
 *
 * @param b - a boolean
 * @return if the argument is true, a string equal to "true" is
 *         returned; otherwise, a string equal to "false" is
 *         returned.
 */
public static boolean valueOf(boolean b) {
```

Generated on Web Page: [https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/lang/String.html#valueOf\(boolean\)](https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/lang/String.html#valueOf(boolean))

valueOf

```
public static String valueOf(boolean b)
```

Returns the string representation of the boolean argument.

Parameters:

b - a boolean.

Returns:

if the argument is true, a string equal to "true" is returned; otherwise, a string equal to "false" is returned.

Sept 20, 2021

Sprenkle - CSC1209

28

28

JavaDocs for Methods

```
/**
 * Returns the string representation of the boolean argument.
 *
 * @param b - a boolean
 * @return if the argument is true, a string equal to "true" is
 *         returned; otherwise, a string equal to "false" is
 *         returned.
 */
public static boolean valueOf(boolean b) {
```

- Expectation in CSCI209
 - **All** methods will have JavaDoc comments

Static Summary

- Static fields and methods are **part of a class** and **not** an object
 - Do not require an object of their class to be created to use them
- When would we make a method **static**?
 - When a method does not have to access an object's state (fields) because all needed data are passed into the method
 - When a method only needs to access static fields in the class

Practice Problem Overview

- Want to write a program that prompts the user for two numbers and then displays the average of those two numbers
- Breaking it down
 1. Main class (the driver program) will handle user input/display
 2. Calculator class will implement method to find average

Sept 20, 2021

Sprenkle - CSCI209

31

31

Static Method Practice

- Implement a **static** method called **average** that
 - Takes as parameters 2 integers
 - Returns the average of those 2 numbers
- What should the signature of this method look like?
 - Use the main method's signature to guide you
 - Discussion: what is the method's return type?
- Discussion: Why should this be a static method?

Sept 20, 2021

Sprenkle - CSCI209

Calculator.java

32

32

Test the Method

- In `Calculator.java`, main method will test the method
 - Call the method and check that output is what you expected
- If there is more than one method in the class, `main` should either be the first or the last method in the class
 - i.e., not somewhere in the middle
- How do we call the method?

Sept 20, 2021

Sprenkle - CSCI209

33

33

Putting it Together

- `main` will prompt user for two integers and display the result of getting the average
 - `main` is the driver
- Development process
 1. Hardcode the numbers
 2. Switch to user input
- How do we call the method?
 - Note that it is different because method is being called from another class

Sept 20, 2021

Sprenkle - CSCI209

`Main.java`

34

34

Looking Ahead

- Assignment 3 – due Tuesday at midnight
- Textbook: Read through Loops and Iteration