

Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM. T 2:00 – 3:00 PM

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email: See syllabus

Supplemental Instruction: TBA

MODULE 1 - ARITHMETIC AND PRINTING

COMPUTER SCIENCE I

OBJECTIVES

- ▶ Review
- ▶ Commenting
- ▶ Arithmetic in Java: Addition, Subtraction, Multiplication
- ▶ Printing



QUESTIONS:

- ▶ How do you create a Java program that prints out, "Hi there!"?
- ▶ How do you compile a Java program?
- ▶ What is created when you compile a Java program?
- ▶ How do you run a Java program?
- ▶ Do you need to re-compile every time you make changes and save a Java program?

COMMENTING

- ▶ Commenting is for you and other programmers that are looking at your code to read! The computer will ignore comments.
- ▶ Commenting is good to explain what a program is doing.
- ▶ Can use `//` for one line comment
- ▶ Can use `/* */` for multiple lines of comments

```
public class Main {  
    public static void main(String[] args) {  
        //this is a comment!!!  
  
        System.out.println("Hi, there!!!");  
  
        /*  
        This is another comment!  
        I can write as much as I want in here!  
        */  
    }  
}
```

| Symbol | Meaning |
|---------------|-------------------|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Remainder/modulus |
| - | negative |

JAVA IS A BIG CALCULATOR!

- ▶ Can use Java to do equations. Follows **PEMDAS**.
- ▶ For now, we'll use it using the print statement. HOWEVER, we'll see a much better way later on.

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.print("1 + 1 = "); //print keeps everything on same line  
        System.out.println(1 + 1); //println jumps to next line  
  
        System.out.print("3 * 8 = ");  
        System.out.println(3 * 8);  
  
        System.out.print("4 * (1+2) = ");  
        System.out.println(4 * (1+2));  
  
        System.out.print("2 + 2 * 3 - (9 + 1) = ");  
        System.out.println(2 + 2 * 3 - (9 + 1));  
  
    }  
}
```

Our Code

```
/Library/Java/JavaVirtualMachines/
```

```
1 + 1 = 2
```

```
3 * 8 = 24
```

```
4 * (1+2) = 12
```

```
2 + 2 * 3 - (9 + 1) = -2
```

Output

```
Process finished with exit code 0
```

LABELING EQUATIONS WITH TEXT USING ONE PRINT LINE

- ▶ So the **+** sign means addition. What if we want to use one print line to label our equations? Let's see what happens if we use the **+** sign...

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("1 + 1 = " + 1 + 1);  
    }  
}
```

```
/Library/Java/JavaVi  
1 + 1 = 11  
  
Process finished with
```

- ▶ It should be 2 but looks like it added the 1 as a **String** (a String represents text in Java)! To fix this, be sure to use **parenthesis** around your equation:

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("1 + 1 = " + (1 + 1));  
    }  
}
```

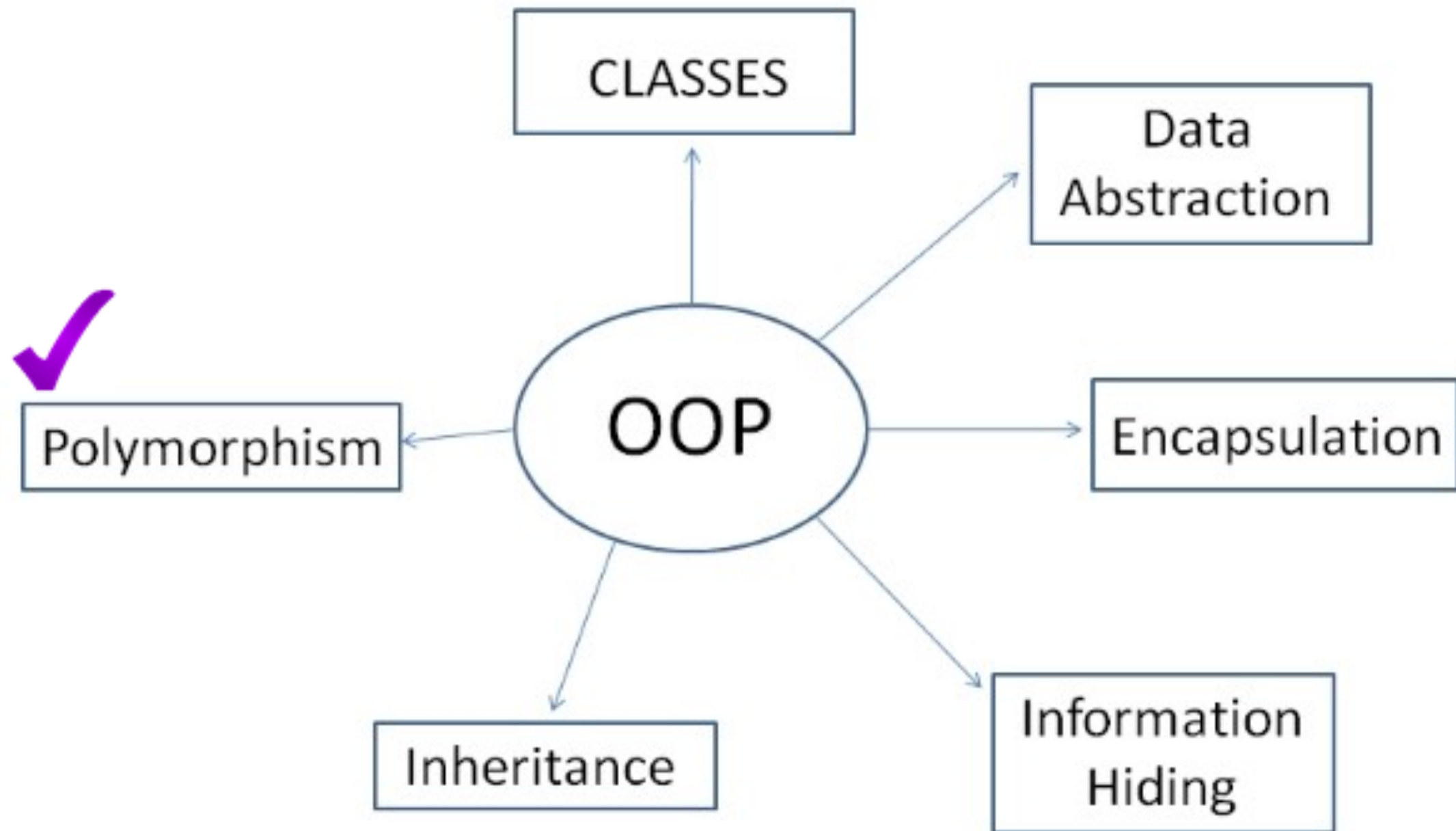
```
/Library/Java/JavaV  
1 + 1 = 2
```

It seems that the **+** symbol has **two** different meanings...

THE + SYMBOL HAS TWO MEANINGS

- ▶ **Polymorphism:** a single construct can have more than one meaning depending on how it is used.
- ▶ "+" is polymorphic in Java because it can mean **concatenation** or **addition** (concatenation means joining two strings together).
 - ❖ **number + number** will always result in "+" being addition.
Example: 5+2 will result in 7
 - ❖ **String + number** or **number + String** or **String + String** will result in "+" being concatenation
Example: "hello"+ 5 + 5 + 1 will result in: hello551
Example: 3 + 4 + "bye" will result in: 7bye (this is because the computer reads from **left to right** resulting in the 3 + 4 being added first.)

What will this produce? System.out.println(3 + 4 + "bye" + 1 + 2 + 8 + 9);



TRY IT! (DON'T LOOK ON THE NEXT SLIDE, THINK ABOUT IT!)

- ▶ What will the following produce in Java?

```
// 1  
System.out.println(":)" + (8 - 3));
```

```
// 2  
System.out.println(1 + 2 + 3 + "hi there");
```

```
// 3  
System.out.println(3 * 2 - (1+5));
```

```
// 4  
System.out.println("The sum is " + 23);
```

SOLUTIONS

► What will the following produce in Java?

► Result:

:)5

6hi there

0

The sum is 23

```
// 1
System.out.println(":)" + (8 - 3));

// 2
System.out.println(1 + 2 + 3 + "hi there");

// 3
System.out.println(3 * 2 - (1+5));

// 4
System.out.println("The sum is " + 23);
```

ESCAPE CHARACTERS

- ▶ Some symbols have special meanings in Java, so you must precede them with a backslash (\) to display them. The backslash is above the enter/return key usually.

| Escape Character | Action |
|------------------|------------------------|
| \t | Inserts a tab |
| \n | Inserts a newline |
| \' | Inserts a single quote |
| \" | Inserts a double quote |
| \\ | Inserts a backslash |

Note: The `\n` and `\t` escape characters will be the most useful in this class.

ESCAPE CHARACTERS – EXAMPLE

Our Code

```
public class Main {  
    public static void main(String[] args) {  
        System.out.print("- - - \n2\t" + (2 * 2));  
    }  
}
```

Output

```
- - -  
2    4
```

- Notice the 2 goes onto the next line due to `\n` and there is space between the 2 and 4 due to `\t`

FORMATTED PRINTING

| Info About Earl the Cat | | |
|-------------------------|-----|-------------|
| Name | Age | Weight(lbs) |
| Earl | 8 | 15.50 |

- ▶ You may need to create **tables** like the one above which can become tedious using only `/t` and `/n`
- ▶ In this case, **formatted printing** is much easier and cleaner
- ▶ General Format and Syntax without data:

```
System.out.printf(" ", , );
```



FORMATTED PRINTING

`System.out.printf(" ", , ,);`



- ▶ **printf** is for formatted printing
- ▶ How to Use:
 - ❖ **Inside " "**: Place parameters/specifications for what you are printing.
 - ❖ **After first comma**: Put the values or data that you are printing. **Each** piece of data **must** be separated by a comma and have a matching specification to what is inside the " ".
- ▶ The format for specifications is:
% [flags] [width] [.precision] conversion-character
 - The flags, width, and precision are optional. (So you need at least the % and **conversion-character**!)

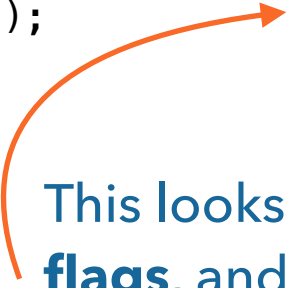
% AND CONVERSION-CHARACTER

- ▶ **ALWAYS** starts with % to indicate what follows after is a **format rule**.
- ▶ A **conversion-character** is **required** and needs to match the data type in the second half of the print statement.

| Symbol | Data Type |
|--------|-----------------------------|
| %d | int |
| %f | double |
| %c | char |
| %C | Capital of a char |
| %s | String |
| %S | Capital of a String |
| %n | For newline, similar to \n |
| %e | Doubles in exponential form |
| %b | boolean |
| %B | Capital of a boolean |

% AND CONVERSION-CHARACTER EXAMPLES:

| Code | | Result |
|---|---|---------------------------------|
| System.out.printf("%d", 15); | | 15 |
| System.out.printf("%f", 8.7); | There are a lot of decimal spaces! .precision may come in handy here. | 8.700000 |
| System.out.printf("%C", 'a'); | | A |
| System.out.printf("%s", "Hello World!"); | | Hello World! |
| System.out.printf("%S%S%n%s%d", "Employee", "ID", "John Smith", 1234567); | | EMPLOYEEID John Smith1234567 |



This looks squished! This is where **flags**, and **width** may come in handy.

.PRECISION

- ▶ Precision is specified starting with a **period** → .
- ▶ Specifies the maximum numbers to follow the decimal in double numbers.
Note: **THIS ROUNDS NUMBERS**
- ▶ Can also be used with Strings – It specifies how many **characters** of the String to display.

| | Code | Result |
|--------------------|--|--------------|
| With .precision | System.out.printf("%.2f", 8.7); | 8.70 |
| | System.out.printf("%.7s", "Hello World!"); | Hello W |
| Without .precision | System.out.printf("%f", 8.7); | 8.700000 |
| | System.out.printf("%s", "Hello World!"); | Hello World! |

WIDTH

- ▶ The width specifies the **minimum number of spaces** to allot for the output.
 - ▶ When deciding on the width, make sure to account for what the maximum lengths of the output may be.
 - ▶ Default is right aligned. Need - flag to left align.
- I only want 1 space between each word, so the width for ID (length of 2) is **3**, the width for 1234567 (length of 7) is **8**.

| | Code | Result |
|---------------|---|-----------------------------------|
| With width | System.out.printf("%7s%10f%8s", "Name", 35.234, "hi"); | Name 35.234000 hi |
| | System.out.printf("%S%3S%n%s%8d", "Employee", "ID", "John Smith", 1234567); | EMPLOYEE ID John Smith 1234567 |
| Without width | System.out.printf("%s%f%s", "Name", 35.234, "hi"); | Name35.234000hi |
| | System.out.printf("%S%S%n%s%d", "Employee", "ID", "John Smith", 1234567); | EMPLOYEEID John Smith1234567 |

FLAG

- ▶ - ➡ left aligns
- ▶ + ➡ displays + in front of numbers
- ▶ , ➡ displays commas in numbers greater than 1000

| Flag | Code | Result |
|------|---|-------------------|
| - | System.out.printf("%-7s%-10f%-8s", "Name", 35.234, "hi"); | Name 35.234000 hi |
| + | System.out.printf("%+d", 15); | +15 |
| , | System.out.printf("%,d", (400 + 2500)); | 2,900 |

SYSTEM.OUT.PRINTF();

- ▶ Let's try to replicate the following:

```
Some Random Numbers
```

8 2.45 90

← Just a title

← 3 spaces between all

```
Name      Job      ID
```

Sally Writer 12345

← 3 spaces between Name and Job,
5 spaces between Job and ID

```
Characters:    a    B    C    d
```

← 2 spaces between all

SYSTEM.OUT.PRINTF();

► Solutions

```
System.out.println("Some Random Numbers");
System.out.printf("%-4d%-7.2f%d", 8, 2.45, 90);
```

| | | |
|------|--------|---------|
| Some | Random | Numbers |
| 8 | 2.45 | 90 |

```
System.out.printf("%-7s%-8s%s", "Name", "Job", "ID\n");
System.out.printf("%-7s%-8s%d", "Sally", "Writer", 12345);
```

| | | |
|-------|--------|-------|
| Name | Job | ID |
| Sally | Writer | 12345 |

```
System.out.printf("%-13s%-3c%-3C%-3C%-3c", "Characters:", 'a', 'b', 'c', 'd');
```

Characters: a B C d

SYSTEM.OUT.PRINTF();

- ▶ Try to replicate the following table (use paper or your computer – doesn't matter) using `System.out.printf()`. *"Info About Earl the Cat"* is just the title using `System.out.println()`. I'll give you a few minutes.

```
Info About Earl the Cat
Name      Age      Weight(lbs)
Earl      8          15.50
```

***Note:** There are **4 spaces** between Name and Age, and **4 spaces** between Age and Weight(lbs)

SYSTEM.OUT.PRINTF(): ANSWER

```
Info About Earl the Cat
Name      Age      Weight(lbs)
Earl      8         15.50
```

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Info About Earl the Cat");
        System.out.printf("%-8s%-7s%s", "Name", "Age", "Weight(lbs)\n");
        System.out.printf("%-8s%-7d%.2f", "Earl", 8, 15.50);
    }
}
```