

CS62

DATA STRUCTURES AND ADVANCED PROGRAMMING

1: Introduction and Java Basics



Alexandra Papoutsaki
she/her/hers

Welcome to a new semester! Hopefully you are all here for CS62: “Data Structures and Advanced Programming”. My name is Alexandra Papoutsaki and most students call me Professor Papoutsaki. My pronouns are she/her/hers. I will be your instructor for the lectures and labs. If you have been registered for the class, you need to also register for the Friday lab for which attendance is mandatory.

Lecture 1: Introduction and Java Basics

- ▶ About this course
- ▶ Getting started
- ▶ Variables
- ▶ Print statements
- ▶ Operators

Some slides adopted from Princeton COS226 course, Algorithms, 4th Edition, Oracle tutorials

Just to familiarize you with how lectures will typically flow, the second slide of each lecture will have an overview of the things we will cover today. So, for example, for today's meeting, the first half will be spent on logistics and then we will have a brief introduction to Java. I want to start with About this course.



Neil Chulani
he/him/his



Evelyn Hasama
she/her/hers



Stevie Kim
he/him/his



Maxim Koretsky
he/him/his



Camden Le
he/him/his



Gloria Lee
she/her/hers



Michele Tang
she/her/hers



Chau Vu
she/her/hers



Emily Wang
she/her/hers



Earn Wonghirundacha
she/her/hers



Alex Wood
he/him/his

Our class has a lot of support both from our class staff and the Department in general. First of all, we have an incredible team of 11(!) TAs that will be assisting with the grading of the assignments, the Friday labs, and mentor sessions. If you have taken a CS class before, you know how invaluable TAs are to our learning environment.

ABOUT THIS COURSE

Slack Channels

- ▶ If registered, already invited to cs62-sp2024 channel in Pomona College Students Slack workspace.
 - ▶ You will find the invitation in the Pomona Students workspace <http://slack.pomona.edu>
- ▶ Department-wide slack workspace: <https://tinyurl.com/PomonaCSSlack>

For the class, we will use the cs62-sp2024 slack channel to answer questions off-class and to send announcements. If you are registered in CS62, I have already invited you to this channel. Please let me know if you are facing any issues with it after class. You can post questions with your name known to everyone or anonymously.

There's also a departmental slack workspace with multiple channels that can help you socialize with other people in the department.

Who are you?

- ▶ Name
- ▶ Year
- ▶ Programming Experience
 - ▶ CS51P
 - ▶ CS51A
 - ▶ Skipped CS51 because of AP or something else

Now I want us all to get a better sense of who's in the room. Can you please state your name and pronouns if you prefer, the year you are in, and what's your programming experience? Did you take CS51A? CS51P? Or did you skip CS51 by taking AP in High School or something else?

ABOUT THIS COURSE

Sakai Survey Due this Friday at midnight

- ▶ “Getting to know you”

To know you a bit better, I have released a Sakai survey named “getting to know you” on Sakai. Please fill it by this Friday midnight.

What is CS62?

- ▶ Beginner to intermediate-level course
- ▶ **Data structures:** Emphasis on how to organize data in a computer based on problem needs
- ▶ **Advanced Programming:** Emphasis on how to write efficient algorithms for modern applications following the Object-Oriented Programming (OOP) paradigm

CS62 is at the cusp of a beginner to intermediate level course. You have already taken some form of programming course either here at Pomona or before and CS54 has exposed you to functional programming and discrete math. It might be already evident from the title: CS62 has multiple missions with the primary being to introduce you to data structures but also to build your programming skills so that you learn how to write large, reliable programs composed from reusable pieces. Throughout this semester, we will be emphasizing the development of clear, modular programs that are easy to read, debug, verify, analyze, and modify

By the end of this course you will...

- ▶ Be familiar with the most commonly used data structures and the time complexity of common operations they support.
- ▶ Be able to determine which data structure is appropriate to use based on the time and memory needs of your application.
- ▶ Be a more confident programmer, comfortable in reading and writing Java code, and familiar with basic Object Oriented Programming principles.

There are three main learning goals of this course: 1) You will become familiar with the most commonly used data structures and the time complexity of common operations they support. 2) You will be able to determine which data structure is appropriate to use based on the time and memory needs of your application. 3) You will be a more confident programmer, comfortable in reading and writing high quality Java code, and familiar with basic object oriented programming principles.

The advanced programming side of CS62

- ▶ In contrast to CS51, labs and assignments will typically be different.
- ▶ Labs are shorter and deliverables are due Friday midnight.
- ▶ Assignments are week- or two weeks-long, due on Thursday midnight.
- ▶ Some assignments will be partner assignments.
- ▶ Labs will mostly teach you tools:
 - ▶ VS Code, Debugger, Unit testing, git, CLI.
- ▶ Assignments will be deliberately vague and will be **using** appropriate data structures to solve interesting problems.
 - ▶ Realistically, no one will hire you and give you the steps to solve a problem.
 - ▶ But we are here to help you understand how to approach problems!

The second half of the course title refers to advanced programming which will be achieved through labs and assignments. In contrast to what happened in CS51, labs and assignments will be typically different. The lab this Friday is the exception. Labs are shorter and any deliverables are typically due the same day, by midnight. Assignments in contrast are opportunities to explore bigger problems and will be typically due every week, on Thursday midnight. In the lab, we will learn useful tools that will make you stronger programmers, tools that are universally used by software engineers regardless of the language: command line, IDEs, debugger, unit testing, git. In contrast, in the assignments you will be using data structures to solve interesting problems. I have to tell you that the assignments are often deliberately vague. In CS51 you might have been used to be given a lot of guidance. We will still guide you but we will also encourage you to explore your own approaches when writing your programs. It's a good idea to remember that we are at a place of growth. Realistically, no one will hire you and give you the steps you need to solve a problem. Having said that, we are here to help you understand how to approach problems and build transferrable skills.

How can I succeed in CS62?

- ▶ Sleep well the night before, eat, come to class, be on time
- ▶ Take notes, participate, fill the worksheet, ask questions, don't stay confused
- ▶ Review slides and do the assigned reading/problems after each lecture
- ▶ Start the assignments early
- ▶ Use the tools we learn in the lab (e.g., Debugger)
- ▶ Practice writing code on paper
- ▶ Learn how to read and write documentation
- ▶ Come to office hours/mentor sessions
 - ▶ But ask for help *after* you have tried solving a problem by yourself
- ▶ Did I say start early?

Here's a list of things that former students, TAs, and faculty came up with on how you can succeed in CS62.

Sleep well the night before, have something to eat, come to class, be on time

Take notes, participate, fill the worksheet, ask questions, don't stay confused

Review slides and do the assigned reading/problems after each lecture

Start the assignments early

Use the tools we learn in the lab (e.g., Debugger)

Practice writing code on paper

Learn how to read and write documentation

Come to office hours/mentor sessions

But ask for help *after* you have tried solving a problem by yourself

Did I say start early?

How can I be a good citizen in CS062?

- ▶ Use laptops/tablets/phones/other fancy electronics only for note taking.
- ▶ Be mindful when in office hours/mentor sessions of other students waiting for help.
 - ▶ Come with specific questions!
- ▶ TAs are students, too. Respect their time outside mentor sessions.
- ▶ We encourage collaboration but we want you to submit your own code.
- ▶ We monitor assignments for plagiarism. This includes using code from other students, websites, or tools like chatGPT.
 - ▶ Academic dishonesty cases are reported to the Dean of Students. Assignments will receive a zero. Exams will receive a zero and half a grade is reduced. Second infraction leads to failure of the course.
 - ▶ If unsure about what's allowed, talk to me.

Beyond how you can succeed as an individual, I hope we can all agree that it's often the entire class dynamic that can make a class unforgettable in a good way. Please keep these pieces of advice in mind throughout the course. Try to avoid using laptops/tablets/phones/other fancy electronics unless you use them for note taking

Be mindful when in office hours/mentor sessions of other students waiting for help.

Come with specific questions

TAs are students too. Respect their time outside mentor sessions.

We encourage collaboration but we want you to submit your own code. This is something that we will be very strict. We will be monitoring your submissions and exams and academic violations can lead to quite strict penalties. Please be careful and avoid copying code from classmates, websites, or even tools like chatGPT.

What will my average week look like?

- ▶ MW lectures.
- ▶ Monday quizzes, starting next week.
- ▶ Weekly assignments due on Thursday midnight.
- ▶ Friday labs (mandatory) due on Friday midnight.

BUDGET AT LEAST 8 HOURS OUTSIDE THE CLASSROOM

CS62 can be a demanding class but it's tons of fun and it will make you great programmers and expose you to the foundation of our science. Please budget your time accordingly and consider your physical and mental health when thinking of taking this class along with balancing other demanding classes, jobs, etc. An average week will look like:

MW lectures.

Monday quizzes starting next week.

Friday labs (mandatory) due on Friday midnight.

Weekly assignments due on Thursday midnight.

Grading summary

- ▶ Weekly Programming Assignments: 30%
 - ▶ Three free days - can use on one assignment or across different assignments. Let me know before the deadline if you will take a late day pass.
 - ▶ If group assignment, both partners have to use a free day
- ▶ Midterm I: 15%
- ▶ Midterm II: 15%
- ▶ Final Exam: 25%
- ▶ Quizzes: 10%
- ▶ Labs: 5%
 - ▶ No late submissions

More information: <http://www.cs.pomona.edu/classes/cs62/>

In case you're wondering, here's the breakout of the grading.

Weekly Programming Assignments: 30%

You have three free days to use either on one assignment or spread them across different assignments - use wisely

Midterm I: 15% (in lab)

Midterm II: 15% (in lab)

Final Exam: 25%

Quizzes: 10%

Labs: 5%

Lecture 1: Introduction and Java Basics

- ▶ About this course
- ▶ Getting started
- ▶ Variables
- ▶ Print statements
- ▶ Operators

Are there any questions? If not, let's learn (or remember) Java!

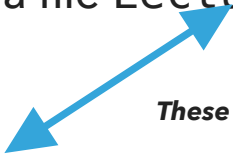
Java

- ▶ One of the most popular general-purpose programming languages.
- ▶ Java code is written in .java files. Each Java file has one Java class which matches the name of the file.
 - ▶ e.g., `Lecture1.java` will have a `Lecture1` class where we'll write all of our code.
- ▶ In order to run a Java program, we will need a special `main` method.
 - ▶ We will ignore both classes and the `main` method for the next two lectures.
- ▶ We will use VS Code as an IDE (Integrated Development Environment).
- ▶ In contrast to Python, we will use curly braces (`{}`) instead of tabs to create logical blocks of code.
- ▶ Single-line comments follow `//` and multi-line are enclosed within `/**/`.

Java is one of the most popular general-purpose programming languages. When we write Java code, we will need to store it in files that end in the extension `.java`. Each Java file has one Java class inside it that matches the name of the file. For example, if I have a `Lecture1.java` file, I am expected to have a `Lecture1` class inside it where all the code will be included. Within our class, we will need to have a special method called `main`, if we want to be able to run our code. We will ignore both classes and the `main` method for the next two lectures, but we will need to have them both to run code even today. In this class, we will write, run, and debug our Java code in the VS Code IDE, although there are a lot of different integrated development environments out there that support Java.

Some basic things you need to read our first Java file: We will be using curly braces (instead of tabs that we used in Python) to create logical blocks of code. And we will comment our code using `//` for single-line comments and `/**/` for multi-line comments.

Example Java file Lecture1.java



These need to match

```
public class Lecture1 {  
    public static void main(String[] args) {  
        //This is a comment  
  
        /*  
        * This is a multi-line comment.  
        * So much easier than Python  
        */  
    }  
}
```

This is an example of a basic Java program that right now doesn't do anything. You will notice that I have saved it in a Lecture1.java file and because of that, I need to have a class Lecture1 inside it. I also have the special method main. Let's ignore the class and main method for now and look at what happens within the main method. You will see that I have added single-line and multi-line comments that are for my own sake and will be ignored if I were to run my code.

Lecture 1: Introduction and Java Basics

- ▶ About this course
- ▶ Getting started
- ▶ **Variables**
- ▶ Print statements
- ▶ Operations

Let's now learn how to work with variables in Java. Variables are labeled containers that hold data. They are used to store information to be referenced by name and manipulated in a computer program. [QUESTION] What types of your variables have you seen before?

Declaring and initializing variables

- ▶ Java is statically-typed: all variables must first be **declared** before use:
 - ▶ `dataType variableName = value;`
- ▶ For example:
 - ▶ `int numberOfCS62Students = 38;`
 - ▶ `int` means it can hold integers, that is positive and negative whole numbers.
 - ▶ The name of the variable is `numberOfCS62Students`.
 - ▶ `=` **assigns** the value on the right to the variable on the left.
 - ▶ The variable is **initialized** to 38.
 - ▶ I always need to finish a statement in Java using a semi-colon (`;`).

The Java programming language is statically-typed, which means that all variables must first be declared before they can be used. This involves stating the variable's type and name. The syntax almost always follows the same pattern `data_type variable_name = value;`

For example, let's say I have a variable that holds the number of student in this class. I would write:

```
int numberOfStudents = 38;
```

`int` means that the variable can hold integers, that is positive and negative whole numbers.

The name of the variable is `numberOfStudents`.

`=` assigns the value on the right to the variable on the left.

The initial value is 38.

I always need to finish a statement in Java using a semi-colon (`;`).

Assigning new values

- ▶ Once a variable is declared, I can reference it elsewhere in the program and **assign** to it a new value.
 - ▶ `variableName = newValue;`
- ▶ For example:
 - ▶ I could change the number of students to 39, if a new student were to join:
 - ▶ `numberOfCS62Students = 39;`
 - ▶ Note that once a variable has been declared, we do **not** declare again its type. But don't forget the semi-colon.

Now that we have declared a variable, we can reference it elsewhere in our code and assign to it a new value. For example, I could change the number of students to 39. Note that I do not declare again its type, I just write the name of the variable and assign to it a new value (don't forget the semi-colon).

Naming conventions

- ▶ Naming variables is very hard. They should be accurately descriptive and understandable to another reader (and to you, days later).
- ▶ It should start with a lowercase letter such as `id`, `name`.
- ▶ It should not start with the special characters like `&`, `$`, `_`.
- ▶ They should be one word. If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as `firstName`, `lastName`.
 - ▶ This is known as camel-case.
- ▶ Avoid using one-character variables such as `x`, `y`, `z`.

<https://www.javatpoint.com/java-naming-conventions>

Naming variables is very hard. They should be accurately descriptive and understandable to another reader (and to you, days later). There are some naming conventions that all Java developers follow when they name their variables:

It should start with a lowercase letter such as `id`, `name`.

It should not start with the special characters like `&`, `$`, `_`.

If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as `firstName`, `lastName`.

This is known as camel-case.

Avoid using one-character variables such as `x`, `y`, `z`.

PRACTICE TIME - Worksheet Problems 1a-b.

- ▶ Declare a variable that stores the number of CS classes you have taken before CS62 at Pomona and initialize it to the appropriate number.
- ▶ Now assume you access this variable at the end of this semester. Assign to it the new value that corresponds to the total number of CS classes you will have taken, including CS62 (and potentially CS101).

Now let's practice declaring, initializing, and assigning values to variables. On your worksheet:

Declare a variable that stores the number of CS classes you have taken before CS62 at Pomona and initialize it to the appropriate number.

Now assume you access this variable at the end of this semester. Assign to it the new value that corresponds to the total number of CS classes you will have taken, including CS62 (and potentially CS101).

ANSWER - Worksheet Problems 1a-b.

- ▶ You should end up with something like:
 - ▶ `int numberOfCSClasses = 2;`
 - ▶ `numberOfCSClasses = 3;`

You should end up with something like:

```
int numberOfCSClasses = 2;  
numberOfCSClasses = 3;
```

Primitive data types

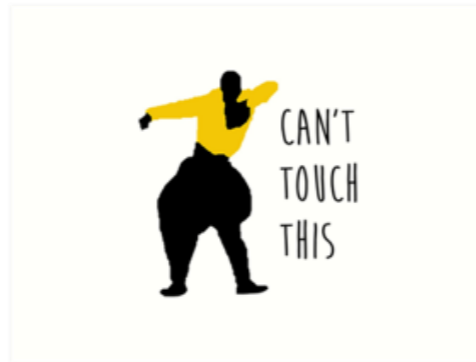
- ▶ In addition to int, Java supports in total eight **primitive data types**. A primitive type is predefined by Java and is named by a **reserved keyword** (that means they have a special meaning. e.g., I can't have a variable named int).
- ▶ The eight primitive data types are:
 - ▶ byte, short, int, long, float, double, boolean, char.

In addition to int, Java supports in total eight primitive data types. A primitive type is predefined by Java and is named by a reserved keyword (that means they have a special meaning. e.g., I can't have a variable named int).

The eight primitive data types are:

byte, short, int, long, float, double, boolean, char.

Reserved words



Reserved Words				
abstract	default	goto	package	synchronized
assert	do	if	private	this
boolean	double	implements	protected	throw
break	else	import	public	throws
byte	enum	instanceof	return	transient
case	extends	int	short	true
catch	false	interface	static	try
char	final	long	strictfp	void
class	finally	native	super	volatile
const	float	new	switch	while
continue	for	null		

In case you are wondering, here is a list of the reserved words that have a special meaning.

Primitive Data Types

Type	Bits	Default	Example
byte	8	0	<code>byte yearsOld = 10;</code>
short	16	0	<code>short pixels = 2;</code>
int	32	0	<code>int luckyNumber = 47;</code>
long	64	0L	<code>long bigNumber = 4747L;</code>
float	32	0.0f	<code>float smallDex = 47.0f;</code>
double	64	0.0	<code>double largeDec = 47.0;</code>
char	16	<code>\u0000'</code>	<code>char initial = 'a';</code>
boolean	1	false	<code>boolean fun = true;</code>

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

Here is a more detailed breakdown of the eight primitive types

byte: The byte data type is an 8-bit integer. It has a minimum value of -128 and a maximum value of 127 (inclusive). The byte data type can be useful for saving memory; the fact that a variable's range is limited can serve as a form of documentation.

short: The short data type is a 16-bit integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive). As with byte, the same guidelines apply: you can use a short to save memory in situations where the memory savings actually matters.

int: The int data type is a 32-bit integer, which has a minimum value of -2³¹ and a maximum value of 2³¹-1.

long: The long data type is a 64-bit integer. long has a minimum value of -2⁶³ and a maximum value of 2⁶³-1. Use this data type when you need a range of values wider than those provided by int.

float: The float data type is a single-precision 32-bit IEEE 754 floating point. As with the recommendations for byte and short, use a float (instead of double) if you need to save memory.

double: The double data type is a double-precision 64-bit IEEE 754 floating point. For decimal values, this data type is generally the default choice.

boolean: The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.

char: The char data type is a single 16-bit Unicode character. It has a minimum value of `\u0000'` (or 0) and a maximum value of `\uffff'` (or 65,535 inclusive).

The most important primitive data types to know

- ▶ `int` - for integers.
 - ▶ e.g., `int numberOfCS62Students = 40;`
- ▶ `double` - for decimal-point numbers.
 - ▶ e.g., `double temperatureCelsius = 27.5;`
- ▶ `boolean` - for the set of values `{true, false}`.
 - ▶ `boolean lovingCS62 = true;`
 - ▶ Note that in contrast to Python, `true` and `false` are not capitalized.
- ▶ `char` - for alphanumeric characters and symbols.
 - ▶ `char firstLetter = 'a';`

But in general, we will mostly use these four:

`int` - for integers.

`double` - for decimal-point numbers.

`boolean` - for the set of values `{true, false}`.

Note that in contrast to Python, `true` and `false` are not capitalized.

`char` - for alphanumeric characters and symbols.

Strings

- ▶ Character strings are not primitive data types but are supported through the `String` class. Note that `String` is capitalized.
- ▶ We enclose strings in double quotes. For example:
 - ▶ `String name = "Alexandra";`
- ▶ Note that single quotes are reserved for the `char` data type.

You might be wondering what about character strings? Strings are not primitive data types but are supported by Java through the `java.lang.String` class. Enclosing your character string within double quotes will automatically create a new `String` object; for example, `String name = "Alexandra";` Note that single quotes are reserved for the `char` data type.

Lecture 1: Introduction and Java Basics

- ▶ About this course
- ▶ Getting started
- ▶ Variables
- ▶ **Print statements**
- ▶ Operators

So far we have seen how to declare variables and initialize and assign a new value to them. What if we want to print their contents to our screen? Or if we want to print a message to the screen in general?

Print statements

- ▶ The method `System.out.println()` is used to print an argument that is passed to. For example:
 - ▶ `System.out.println("Hello World");`
 - ▶ `System.out.println(name); //will print Alexandra`
 - ▶ `System.out.println(numberOfCS62Students); //will print 40`
 - ▶ Note that in contrast to Python, you do not need to convert non-string arguments to string, this is done automatically.

The special print method that we will use is `System.out.println()` and it prints to the screen any argument we pass to it. For example:

```
System.out.println("Hello World");
```

```
System.out.println(name); //will print Alexandra
```

```
System.out.println(numberOfCS62Students); //will print 40
```

Note that in contrast to Python, you do not need to convert a number to a string

String concatenation

- ▶ Strings are more commonly **concatenated** with the + operator, as in `"Hello," + " world" + "!"` which results in `"Hello, world!"`
- ▶ The + operator is widely used in print statements, e.g.,
 - ▶ `System.out.println("My name is " + name + " and I will be teaching " + numberOfCS62Students + " students this semester");`

You might be wondering what about character strings? Strings are not primitive data types but are supported by Java through the `java.lang.String` class. Enclosing your character string within double quotes will automatically create a new `String` object; for example, `String name = "Alexandra"`; Note that single quotes are reserved for the `char` data type.

PRACTICE TIME - Worksheet Problem 1c

- ▶ Declare and initialize a variable whose type is a primitive and pass it into a print statement, using string concatenation at least once.

Let's practice what we've learned so far. Turn into your worksheets and work on 1c. Declare and initialize a variable whose type is a primitive and use it into a print statement that uses string concatenation at least once.

Lecture 1: Introduction and Java Basics

- ▶ About this course
- ▶ Getting started
- ▶ Variables
- ▶ Print statements
- ▶ Operators

Now that we've learned how to declare and initialize variables, you probably want to know how to do something with them. Learning the operators of the Java programming language is a good place to start. Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result.

Operator precedence

Operators	Precedence
postfix	expr++ expr--
unary	++expr --expr +expr -expr !expr
multiplicative	* / %
additive	+ -
relational	< > <= >=
logical AND	&&
logical OR	
ternary	? :
assignment	= += -= *= /= %=

As we explore the operators of the Java programming language, it may be helpful for you to know ahead of time which operators have the highest precedence. The operators in the following table are listed according to precedence order. The closer to the top of the table an operator appears, the higher its precedence. Operators with higher precedence are evaluated before operators with relatively lower precedence. Operators on the same line have equal precedence. When operators of equal precedence appear in the same expression, a rule must govern which is evaluated first. All binary operators except for the assignment operators are evaluated from left to right; assignment operators are evaluated right to left. Please note that this is not an exhaustive list and there are more operators supported by Java

The Simple Assignment Operator

- ▶ One of the most common operators that we've already encountered is the simple assignment operator "="; it assigns the value on its right to the operand on its left:
 - ▶ `int age = 19;`
 - ▶ `int year = 2024;`

One of the most common operators that we've already encountered is the simple assignment operator "="; it assigns the value on its right to the operand on its left:

```
int age = 19;
```

```
int year = 2024;
```

Arithmetic Operators

- ▶ Java operators support addition, subtraction, multiplication, division, and remainder/modulo.

Operator	Description
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator

Java operators support addition, subtraction, multiplication, division, and remainder/modulo and they behave exactly as you'd expect based on your prior math experience.

PRACTICE TIME - Worksheet Problem 2

Assume you are given the following Java code. What would be printed on your screen?

```
int result = 1 + 2;
System.out.println("1 + 2 = " + result);
int original_result = result;

result = result - 1;
System.out.println(original_result + " - 1 = " + result);
original_result = result;

result = result * 2;
System.out.println(original_result + " * 2 = " + result);
original_result = result;

result = result / 2;
System.out.println(original_result + " / 2 = " + result);
original_result = result;
```

...

Let's look into our worksheets. Assume you are given the following statements. What do you think would be printed in your screen?

ANSWER - Worksheet Problem 2

$$1 + 2 = 3$$

$$3 - 1 = 2$$

$$2 * 2 = 4$$

$$4 / 2 = 2$$

$$2 + 8 = 10$$

$$10 \% 7 = 3$$

You should see:

$$1 + 2 = 3$$

$$3 - 1 = 2$$

$$2 * 2 = 4$$

$$4 / 2 = 2$$

$$2 + 8 = 10$$

$$10 \% 7 = 3$$

Other Assignment Operator

- ▶ The assignment operators `+=`, `-=`, `*=`, `/=`, and `%=` are a compound of arithmetic and assignment operators.
- ▶ They operate by adding/subtracting/multiplying/dividing/taking the remainder of the current value of the variable on the left to the value on the right and then assigning the result to the operand on the left. E.g.,
- ▶ `num1 += num2;` means `num1 = num1 + num2;`

You might also encounter the assignment operators `+=`, `-=`, `*=`, `/=`, and `%=` which are a compound of arithmetic and assignment operators. They operate by adding/subtracting/multiplying/dividing/taking the remainder of the current value of the variable on the left to the value on the right and then assigning the result to the operand on the left. E.g., `num1 += num2;` means `num1 = num1 + num2;`

Unary Operators

- ▶ Unary operators require only one operand.

Operator	Description
+	Unary plus operator; indicates positive value (not necessary to have)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

The unary operators require only one operand; they perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a boolean.

PRACTICE TIME - Worksheet Problem 3

Assume you are given the following Java code. What would be printed on your screen?

```
int result = +1;
System.out.println(result);

result--;
System.out.println(result);

result++;
System.out.println(result);

result = -result;
System.out.println(result);

boolean success = false;
System.out.println(success);

System.out.println(!success);
```

Let's look into our worksheets. Assume you are given the following statements. What do you think would be printed in your screen?

ANSWER - Worksheet Problem 3

1

0

1

-1

false

true

You should see

1

0

1

-1

false

true

Pre vs post-fix operators

- ▶ The increment/decrement operators can be applied before (prefix) or after (postfix) the operand.
- ▶ The code `result++;` and `++result;` will both end in `result` being incremented by one. The only difference is that the prefix version (`++result`) evaluates to the incremented value, whereas the postfix version (`result++`) evaluates to the original value.
- ▶ If you are just performing a simple increment/decrement, it doesn't really matter which version you choose. But if you use this operator in part of a larger expression, the one that you choose may make a significant difference

The increment/decrement operators can be applied before (prefix) or after (postfix) the operand.

The code `result++;` and `++result;` will both end in `result` being incremented by one. The only difference is that the prefix version (`++result`) evaluates to the incremented value, whereas the postfix version (`result++`) evaluates to the original value.

If you are just performing a simple increment/decrement, it doesn't really matter which version you choose. But if you use this operator in part of a larger expression, the one that you choose may make a significant difference

Pre vs post-fix operators example

```
int i = 3;
i++;
// prints i (4)
System.out.println(i);
++i;
// prints i (5)
System.out.println(i);
// first increments to 6 then prints it (6)
System.out.println(++i);
// first prints i (6) then increments i to 7
System.out.println(i++);
// prints i (7)
System.out.println(i);
```

Here is an example that showcases how pre and post-fix operators differ.

```
int i = 3;
i++;
// prints i (4)
System.out.println(i);
++i;
// prints i (5)
System.out.println(i);
// first increments to 6 then prints it (6)
System.out.println(++i);
// first prints i (6) then increments i to 7
System.out.println(i++);
// prints i (7)
System.out.println(i);
```


Equality and relational operators

- ▶ Determine if one operand is greater than, less than, equal to, or not equal to another operand

Operator	Description
==	equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

The equality and relational operators determine if one operand is greater than, less than, equal to, or not equal to another operand. The majority of these operators will probably look familiar to you as well. Keep in mind that you must use "==" , not "=", when testing if two primitive values are equal.

== equal to

!= not equal to

> greater than

>= greater than or equal to

< less than

<= less than or equal to

Conditional operators

- ▶ The && and || operators perform Conditional-AND and Conditional-OR operations on two boolean expressions

exp1	exp2	exp1 && exp2	exp1 exp2
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

The && and || operators perform Conditional-AND and Conditional-OR operations on two boolean expressions:

exp1	exp2	exp1 && exp2	exp1 exp2
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

PRACTICE TIME - Worksheet Problem 4

Consider the following code snippet:

```
int i = 10;  
int n = i++%5;
```

- a. What are the values of `i` and `n` after the code is executed?

- b. What are the final values of `i` and `n` if instead of using the postfix increment operator (`i++`), you use the prefix version (`++i`)?

Let's look into our worksheets. Assume you are given the following code snippet.

```
int i = 10;  
int n = i++%5;
```

- a. What are the values of `i` and `n` after the code is executed?

- b. What are the final values of `i` and `n` if instead of using the postfix increment operator (`i++`), you use the prefix version (`++i`)?

ANSWER - Worksheet Problem 4

- a. `i` is 11, and `n` is 0
- b. `i` is 11, and `n` is 1.

- a. `i` is 11, and `n` is 0
- b. `i` is 11, and `n` is 1.

Lecture 1: Introduction and Java Basics

- ▶ About this course
- ▶ Getting started
- ▶ Variables
- ▶ Print statements
- ▶ Operators

That will be it for today. As a recap, we discussed this course is about and some logistics about it. Then we saw how to declare, initialize, and change the values of variables. We saw the eight primitive data types and strings. We learned how to print string messages and the contents of variables. We learned the basic operators that Java supports. I will see you in lab on Friday where we will set up your computers and ensure than you can all run Java. Don't forget that next Monday we have our first quiz which will cover the topics we saw today.

Readings:

- ▶ Variables: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>
- ▶ Operators: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>

Code

- ▶ [Lecture 1 code](#)

Worksheet

- ▶ [Lecture 1 worksheet](#)

If you want to deepen your understanding of what we covered, at the end of each lecture, you will see links to readings and code we saw together in class, as well as a link to the worksheet of the day.