

Programming Assignment 0: Warm Up and Review

Specification

Overview

This assignment is intended to be a review and warm up for CSE 123. It will require you to use the skills and concepts that you should be familiar with from your prior programming experience. It will also serve as an introduction to your first IDE, Visual Studio Code. This is designed to help everyone review and practice the programming skills that will be necessary to succeed in CSE 123. While we don't necessarily expect everyone to find this assignment *easy*, if you find yourself having major difficulties with any of the content, please contact the course staff to get support!

□□ Learning Objectives

By completing this assignment, students will demonstrate their ability to:

- Use the Visual Studio debugger to go through a program line-by-line
- Predict the behavior and results of executing a Java program that includes complex and/or compound data
- Identify errors in a Java program's state or behavior, and implement fixes for identified errors
- Write functionally correct Java programs that meet a provided specification using compound data types
- Write functionally correct Java classes to represent new, compound data types

□ Assignment Structure

Unlike most future assignments in CSE 123, this assignment will consist of a series of individual questions and problems. By focusing on a few separate and slightly smaller programming problems, we can help you target your practice on the programming skills that will set you up for success in our course.

Don't worry if you don't find this assignment particularly exciting since we are focusing on review here. We will have many very exciting applications of programming in our future assignments!

To complete this assignment, you should go to each slide and complete the task(s). For quiz slides (indicated by a blue clipboard icon), provide an answer to each question. For coding challenge slides

(indicated by a yellow angle brackets icon), upload your code to the workspace. When you have successfully completed each slide, you will see the dot next to the slide title fill in. The assignment is complete when you have a filled-in dot for every slide, including the "Final Submission" slide. The problems can be worked on in any order.



Exception: you do not need to write any code in the "Sample Submission" slide, and it will not give you a filled-in dot!

□ Feeling Stuck?

While we expect this assignment to be review, it's still OK if you find this assignment a bit challenging! Remember that learning is a challenging process, and you don't have to do it alone!

- You can visit the [Introductory Programming Lab \(IPL\)](#) to talk with a TA about programming concepts or get help on assignments.
- You can stop by [instructor office hours](#) to discuss course concepts or get help on assignments or discuss the course in general.
- You can post questions on the [discussion board](#)! You can make questions public (anyone can see them) or private (only course staff can see them). This is a great way to asynchronously get help on an assignment or ask questions about the course.

It is OK to get stuck and feel challenged by this assignment. However, note that this is intended to be a warm-up for the type of programming we will be doing for the rest of the quarter, and the tasks we will be solving in future weeks will be more complex than these problems and rely on a solid grasp of the skills practiced in this assignment. If you feel like you cannot do this assignment at all, we recommend reaching out to the course instructor(s) (cse123-instructors@cs.washington.edu) or the CSE undergrad advisors (ugrad-adviser@cs.washington.edu) to discuss more about academic planning and which programming course might be a good fit for your goals.

□ Submission

When you are ready to submit, go to the "□ Final Submission □" slide, read the statement and fill in the box, then click "Submit" in the upper-right corner. You may submit as many times as you want until the due date.

You can see your previous submissions by clicking the three dots icon in the upper-right and selecting "Submissions and Grades." By default, we will grade your latest submission from before the deadline. However, if you would like us to grade a different submission, you can select that submission on the left side of the window and click "Set final." Note that we will not grade any submission made after the deadline-- if you mark a submission after the deadline as final, we will grade your most-recent on-time submission instead.

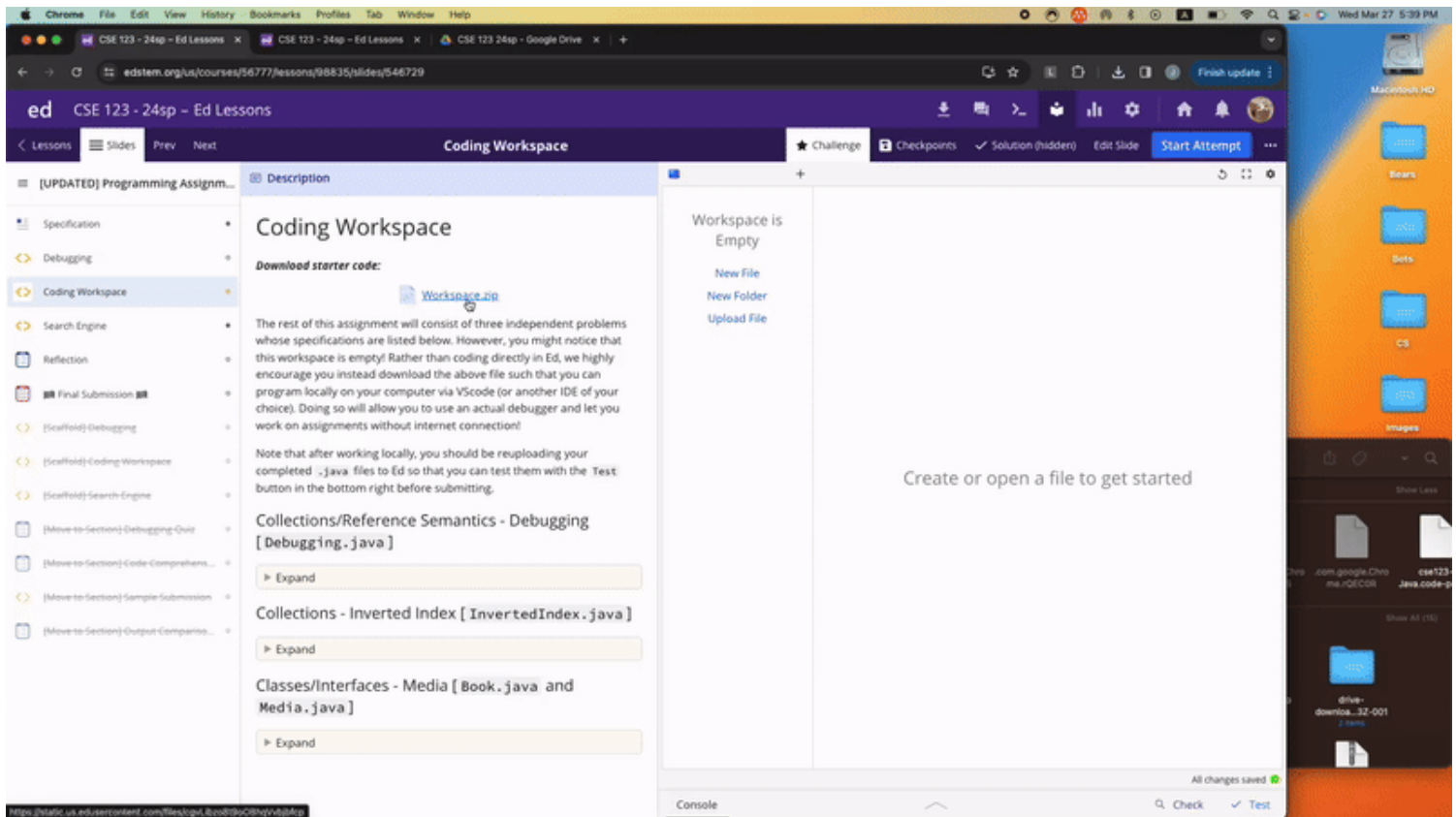
Please make sure you are familiar with the resources and policies outlined in the [syllabus](#) and the [programming assignments](#) page.

Recommended Development Process

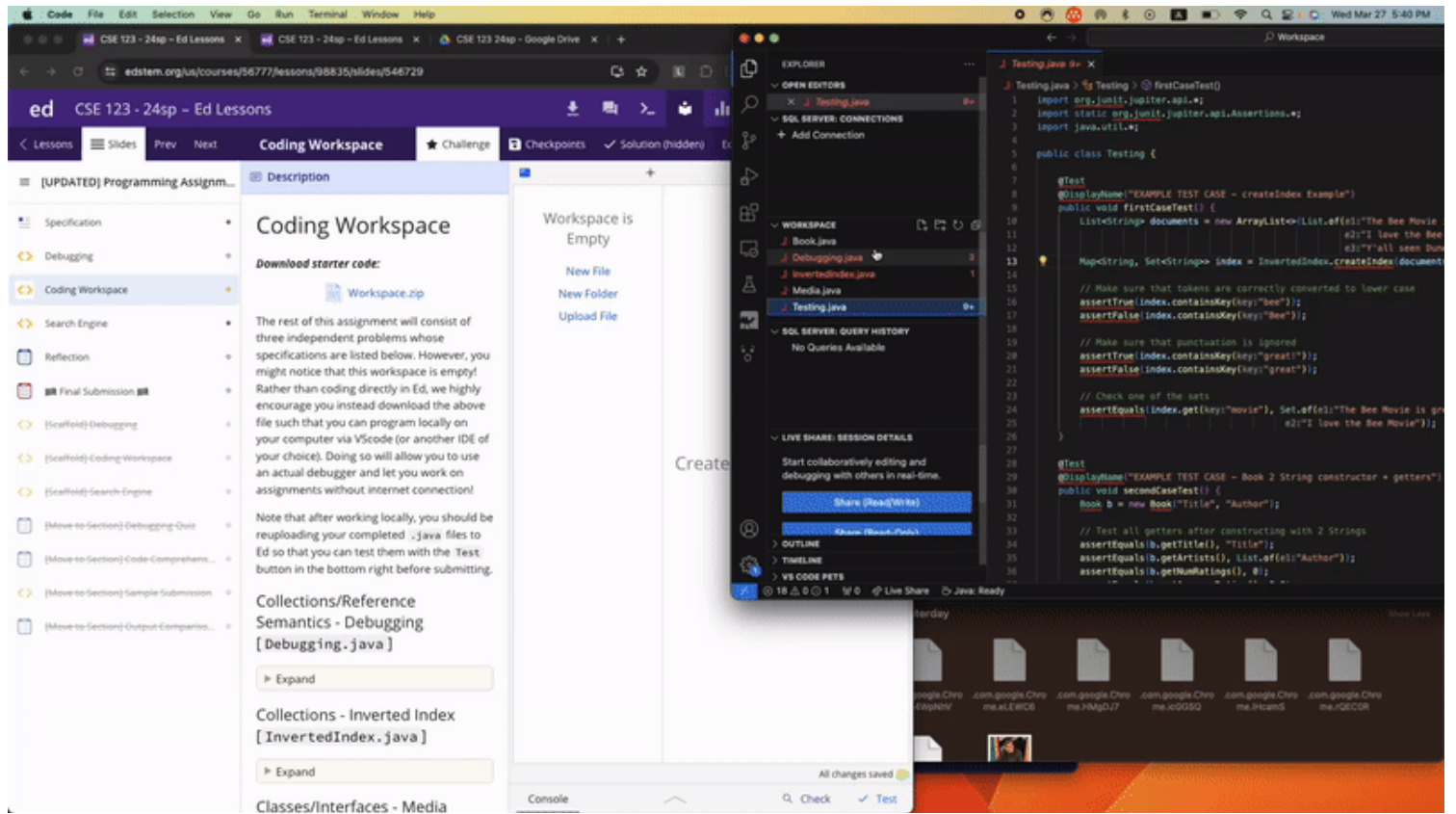
The general process we're expecting you to follow when working on and submitting HW assignments is slightly different from previous courses now that you have an IDE (VSCode)! You should

1. Download the provided `.zip` file
2. Find it in `Finder` / `File Explorer`
3. Unzip it to get the folder with relevant files.
4. In VSCode, click `File > Open Folder`, and select the recently unzipped folder to open it!

This process is outlined in the following `.gif` (mac shown, but the process would be the same for windows with File Explorer)



Then, once you've finished, you should re-upload your code by copy-pasting or drag-dropping the files back into Ed! At this point you can run tests via the `Test` button.



Debugging

Background

Before you begin this assignment, you need to download and set up Visual Studio Code with the CSE 123 Profile! If you have not done so already (either in section or by yourself), you can do this [here!](#)

Once you have set up the Visual Studio "IDE" (integrated development environment – an application that helps you code), download and open the file below. You will have to determine the correct code needed to defuse the bomb by using the debugger (more info the the following video / on the [course website](#)). Time is of the essence and the fate of the world rests in your hands!

Download starter code:

 [P0_Debugging.zip](#)



NOTE: You should only need to update `line 30` of the provided file: `defuse("00000");` to the defusal code you determine using the VSCode debugger!



WARNING: We're trying our best to encourage you to use the VSCode debugger here, so you may find that `println`s don't actually print anything! This is expected behavior :)

The following video will walk you through some of the useful features of the VSCode debugger. Watch the video, and then defuse the bomb!

An error occurred.

[Try watching this video on www.youtube.com](#), or enable JavaScript if it is disabled in your browser.

Coding Workspace

Download starter code:



The rest of this assignment will consist of three independent problems whose specifications are listed below. However, you might notice that this workspace is empty! Rather than coding directly in Ed, we highly encourage you instead download the above file such that you can program locally on your computer via VSCode (or another IDE of your choice). Doing so will allow you to use an actual debugger and let you work on assignments without internet connection!

Note that after working locally, you should be reuploading your completed `.java` files to Ed so that you can test them with the `Test` button in the bottom right before submitting.

Collections/Reference Semantics - Debugging [`Debugging.java`]

▼ Expand

One of the TAs has been programming in Python for too long, and forgot how to code in Java! They wrote a solution to the following problem, but accidentally included some bugs:

Write a method called `deepCopy` that takes as a parameter `map` whose keys are strings and whose values are sets of integers and that creates and returns a new map that is a copy of the `map` parameter. For example, given a variable called `map` that stores the following information:

```
{"cse121"=[17, 42], "cse122"=[10, 12, 14], "cse123"=[-97, 98, 99, 100]}
```

The call `deepCopy(map)` should return a new map whose structure and content are identical to `map`. Any later modifications to `map` or the sets in `map` following this call should not be reflected in the copy. The map you construct should store keys in alphabetical order, while the sets in the values should prefer fast lookup speed. Your method should not modify the contents of the map passed as a parameter.

There are 5 bugs in the following program. Find and fix them all!

Collections - Inverted Index [`InvertedIndex.java`]

▼ Expand

Write a method called `createIndex` that creates an inverted index for a list of documents. Your method should take one argument, a list of "documents" where each document is represented as a string. Your method should return a map where the keys are individual words that appear in the

parameter list of documents and the values are sets of documents in which those words appear.

For example, suppose the variable `titles` contains the following list:

```
[Raiders of the Lost Ark, The Temple of Doom, The Last Crusade]
```

In this case, the call `createIndex(titles)` would return the following map:

```
{ark=[Raiders of the Lost Ark], crusade=[The Last Crusade], doom=[The Temple of Doom], last=[The  
lost=[Raiders of the Lost Ark], of=[The Temple of Doom, Raiders of the Lost Ark], raiders=[Raid  
temple=[The Temple of Doom], the=[The Temple of Doom, The Last Crusade, Raiders of the Lost Ark
```

The keys of the returned map should be case-insensitive (i.e. treat "The" and "the" as the same word). The keys of the returned map should be in sorted order, while the sets in the values should prefer fast lookup speed.

You may assume that the parameter passed in non-null, that each element of the parameter is a non-null and non-empty string, and that words in each document are separated by a single space. We recommend that you approach solving this problem using the `Scanner` class, as it will make following steps of the assignment easier.



When writing your class, be sure to follow all guidelines in the [Code Quality Guide](#) and [Commenting Guide](#).

Any additional helper methods created, but not specified in the spec, should be declared *private*.



Note: This is essentially how many search engines work! They build up an index mapping "search terms" (which could be more than single words) to "documents" (which could be more than just strings). See [Wikipedia](#) for more information.

Classes/Interfaces - Media [`Book.java` and `Media.java`]

▼ Expand

Write a Java class called `Book` that implements the provided `Media` interface and represents a book. For books, the artists are considered to be the author(s).

Your class should have two constructors:

```
public Book(String title, String author)
```

- Creates a book with the provided title and single author.

```
public Book(String title, List<String> authors)
```

- Creates a book with the provided title and multiple authors.

The title and author(s) should *not* be able to be modified by a client after creation.

In addition to the methods required by the interface, your `Book` class should include the following method:

```
public String toString()
```

- Produce a readable string representation. If the book has zero ratings, the string representation should be:

```
<name> by [<authors>]: No ratings yet!
```

- If the book has at least one review, the string representation should be:

```
<name> by [<authors>]: <average rating> (<num ratings> ratings)
```

- The average rating should be rounded to at most two decimal places *in the string representation only*. (The `getAverageRating` method should return the actual average without rounding.)



When writing your class, be sure to follow all guidelines in the [Code Quality Guide](#) and [Commenting Guide](#).

Any additional helper methods created, but not specified in the spec, should be declared ***private***.

Testing

Below are the testing requirements for this assignment:

- `Debugging.java`: None
- `InvertedIndex.java`: You must write at least one *additional* JUnit test for `createIndex` within `Testing.java`
- `Book.java`: You must write *additional* JUnit tests such that each constructor/instance method is tested at least once within `Testing.java`

Search Engine

Download starter code:

 [PO_SearchEngine.zip](#)

Once you've completed all aspects of the assignment, we can put all these pieces together into an application! Most of the work has been done for you here, but you'll have to integrate all 3 of your fixes/implementations from the workspace slide. This will involve 3 steps:

1. Paste your fixed version of `deepCopy @ line 47`
 1. This will require updates such that your solution uses `Media` instead of `Strings`. (To get an idea of what we mean, take look at what's currently written on `line 47`).
2. Paste your implementation of `createIndex @ line 58`
 1. This will require updates such that your solution uses `Media` instead of `Strings` as documents. You are guaranteed that text file of the provided `Media` item can be found via `new File(BOOK_DIRECTORY + "/" + m.getTitle() + ".txt")`. You should create a `Scanner` over this file to get all the words and create your index.
 2. These changes sound like a lot, but in our solution only 3 lines changed (including the method header).
3. Paste your implementation of `Book` within `Book.java`



NOTE: You don't need to worry about reuploading the books directory as it's already included in the scaffold!

Once you've done these 3 things hit the run button to see your hard work! You'll have a working search engine over all the books within the `books` directory!

Reflection

The following questions will ask that you practice **metacognition** to reflect on the topics covered on this assignment and your experience completing it. For each question, focus on your plan and/or process for working through the assignment along with the CS concepts. Think about things like how you organized your working time, what sorts of things tended to go wrong, and how you dealt with those errors or mistakes.

Please answer all questions.

Question 1

Describe your process using the VSCode debugger in the Debugging problem (`Bomb.java`). What skills have you learned or practiced to help you when debugging code? How will you use or adapt this process for future assignments?

Question 2

Choose either the Inverted Index or Media problem: describe how you went about testing that the code you wrote for that problem was correct and met the requirements. What specific test cases did you consider? Why were those cases important?

Question 3

What skills did you learn and/or practice with working on this assignment?

Question 4

What did you struggle with most on this assignment?

Question 5

What questions do you still have about the concepts and skills you used in this assignment?

Question 6

About how long (in hours) did you spend on this assignment? (Feel free to estimate, but try to be close.)

Question 7

Was any part of the specification or requirements unclear? If so, which part(s), how was it unclear, and how could it have been made more clear?

Question 8

[OPTIONAL] Do you have any other feedback, questions, or comments about this assignment?

(Note that we may not be able to respond to questions here, so please post on the message board if you would like a response!)

□ Final Submission □

□ Final Submission□

Fill out the box below and click "Submit" in the upper-right corner of the window to submit your work.

Question

I attest that the work I am about to submit is my own and was completed according to the course [Academic Honesty and Collaboration](#) policy. If I collaborated with any other students or utilized any outside resources, they are allowed and have been properly cited. If I have any concerns about this policy, I will reach out to the course staff to discuss *before* submitting.

(Type "yes" as your response.)