

## Assignment Report for Canvas

Course and Section	CSC 340-01, TIC
Assignment Name	Assignment 2
Due Date and Time	06-23-2023 at 11:55 PM
First Name and Last Name	Juan Segura Rico
SFSU Email Account	jsegurarico@sfsu.edu
First Name and Last Name of Teammate	N/A
SFSU Email Account of Teammate	N/A

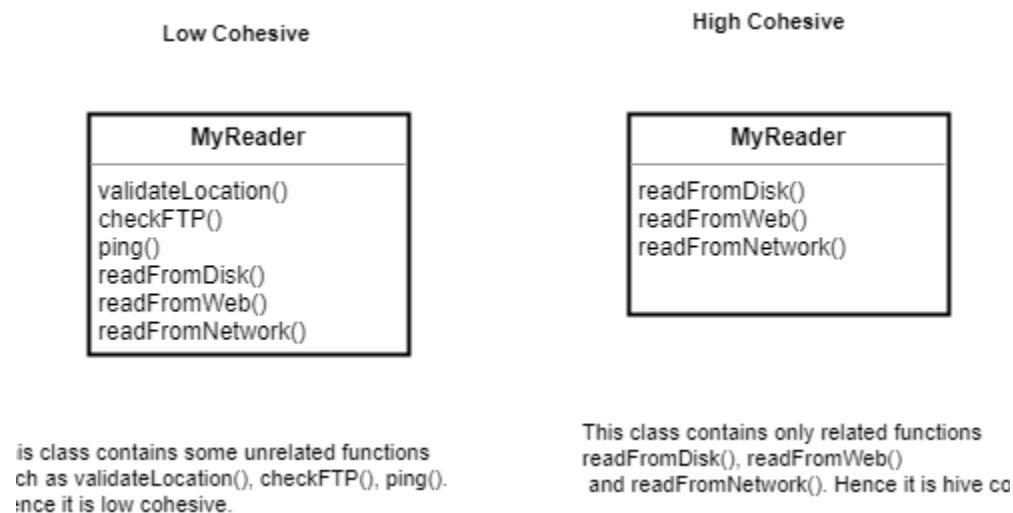
### PART A

#### Question Description and Analysis:

- Please choose 5 guidelines and discuss them in depth. For each guideline, use at least one page for your discussion. It is OK to use code to help demonstrate your points. The code portion, if any, should not take up more than 1/3 of each guideline's discussion.

**Answer:**

**Cohesion:** This refers to the measure of how closely related the elements within a module or class are to each other. It is important for various reasons. Cohesive code is easier to read and understand. When the elements within a module or class are closely related and share a common purpose, it becomes clearer for developers to comprehend the code's logic and behavior. Cohesion promotes code readability, reducing complexity and cognitive load, which ultimately leads to faster comprehension and better maintainability. Cohesion is closely related to encapsulation, another key principle of object-oriented programming. Cohesive modules or classes encapsulate related data and behavior, ensuring that internal details are hidden from external entities. Encapsulation provides a clear boundary and well-defined interface for interacting with a module or class, promoting better code organization. By focusing on well-defined responsibilities and closely related elements, cohesiveness leads to more manageable and clear code.



(Found from a JAVA Guide website:

<https://www.javaguides.net/2018/08/cohesion-in-java-with-example.html>)

**Consistency:**

It's pretty obvious that one needs to be consistent when coding. It's important for many reasons. To achieve consistency in coding, developers often follow established guidelines and style conventions specific to the programming language or framework being used. These guidelines typically cover aspects such as indentation, naming conventions, commenting, code organization, variable usage, and formatting. By adhering to these conventions, developers can create a codebase that is uniform and easy to understand. Some of the most important things your code must have to be consistent include readability, maintainability, and functionality. When code follows a consistent style and structure, it becomes more predictable and less prone to confusion (easier to read and understand). By following consistent coding practices, it becomes simpler to identify and fix bugs, make enhancements, and add new features. It reduces the learning curve for developers who join a project and promotes collaboration among team members. According to Liang's Guidelines, the order in which elements are placed go like this: data declaration (1st), constructors (2nd), and finally methods (3rd).

**Example:**

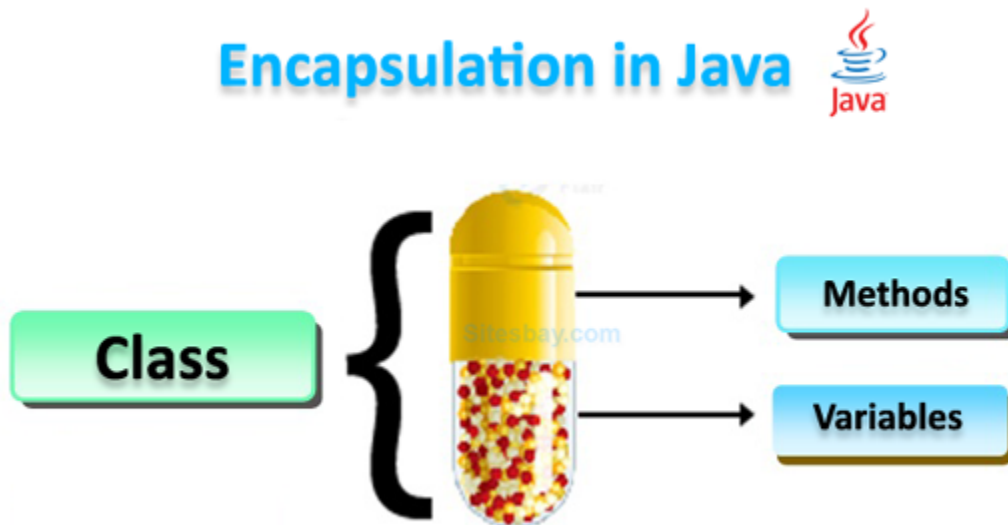
```
public class HelloWorld {  
    public static void main(String[] args) {  
        String message = "Hello, world!";  
        printMessage(message);  
    }  
}
```

```
public static void printMessage(String message) {  
    System.out.println(message);  
}  
}
```

By following consistent coding practices, even in a small example like this, the code becomes more readable and easier to understand,

**Encapsulation:**

This is a fundamental concept in object-oriented programming (OOP) that involves bundling data and the methods that operate on that data into a single unit, known as a class. It provides several benefits and is considered a crucial principle in software development. A key point to why it is important includes, *data protection* which includes our modifiers such as public and private to control how our code is accessed. Another key is *code reusability*, by defining well-designed classes with clear responsibilities and interfaces, you can reuse those classes in different parts of your codebase or in future projects. This saves development time and promotes code consistency (they both fall into each other's sections (consistency)). Examples of encapsulation include private variables and public methods.



(Image found to show depiction of my last sentence)

### **Clarity:**

This refers to the quality of code being easily understandable and comprehensible by anyone. It plays a crucial role in coding for several reasons. Code clarity helps code maintainability. When code is clear and easy to read, it becomes simpler to identify and fix bugs, make modifications, and add new features. Clear code reduces the time and effort required for maintenance tasks, making it more efficient for us to complete our task at hand. Clear code is essential for effective collaboration among team members. When multiple people are working on the same project, code clarity ensures that everyone can understand each other's code. This helps ensure we are all on the same page and understand what we're doing. Writing clear and readable code is a best practice that ultimately leads to more efficient development, easier maintenance, and better overall software quality.

### **Good Clarity:**

```
public class person {
    int age = 21;

    public boolean isOfAge(){
        return 2023 - currentAge >= 21
    }
}
```

Here is declared and calculated

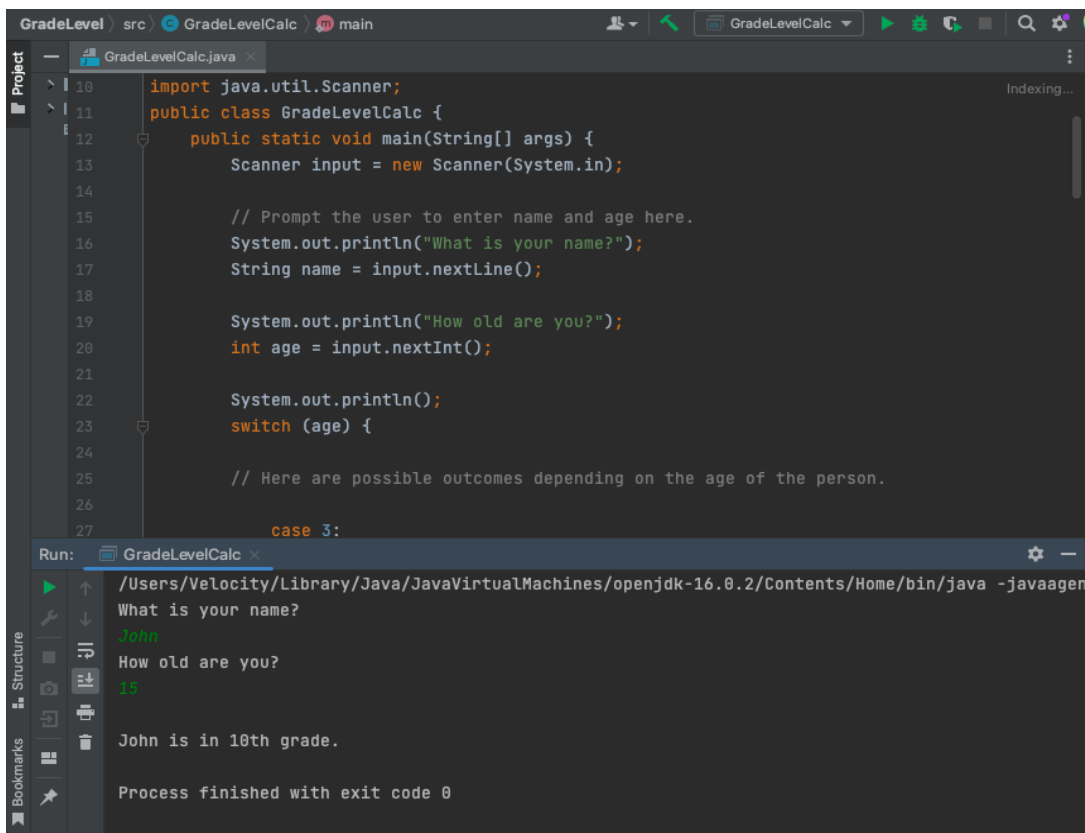
### **Bad Clarity:**

```
public class person {
    int age = 21;
    boolean OfAge = true;
}
```

Here it isn't.

**Completeness:**

This is needed in order for our code to function properly and correctly. Completeness ensures that the code covers all the required functionality and features specified by the project requirements or user needs. It means that the code accomplishes its intended purpose without missing any critical elements. Incomplete code may result in bugs, unexpected behavior, or the inability to perform certain tasks, leading to a poor user experience (will lead to a bad grade too). When code is complete, it reflects a thorough understanding of the problem domain and demonstrates good software engineering practices. Complete code is typically well-structured, modular, and organized, making it easier to understand, test, and maintain. Writing complete code leads to more maintainable and user-friendly programs.



The screenshot shows an IDE with a Java file named `GradeLevelCalc.java`. The code is as follows:

```

10 import java.util.Scanner;
11 public class GradeLevelCalc {
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14
15         // Prompt the user to enter name and age here.
16         System.out.println("What is your name?");
17         String name = input.nextLine();
18
19         System.out.println("How old are you?");
20         int age = input.nextInt();
21
22         System.out.println();
23         switch (age) {
24
25             // Here are possible outcomes depending on the age of the person.
26
27             case 3:

```

The bottom panel shows the program's execution output:

```

Run: GradeLevelCalc x
/Users/Velocity/Library/Java/JavaVirtualMachines/openjdk-16.0.2/Contents/Home/bin/java -javaagen
What is your name?
John
How old are you?
13

John is in 10th grade.

Process finished with exit code 0

```

(Example of a grade level calculator program I did in my first year... it runs how its supposed to and has comments to show what things are for)

---

**PART B****Question Description and Analysis:**

We are hired to implement an interactive dictionary. Our dictionary takes input from users and uses the input as a search key to look up values associated with the key.

Requirements:

- **Coding:** No hard coding, [https://en.wikipedia.org/wiki/Hard\\_coding](https://en.wikipedia.org/wiki/Hard_coding). Please think about Dynamic and Scalable.
- **Data Source:** Store the original data in a set of enum objects. Each keyword, each part of speech, and each definition must be stored in a separate data field. Do not combine them such as storing three parts in one String.
- **Data Structure:** Use existing data structures or create new data structures to store our dictionary's data.
- **Data Loading:** When our program starts, it loads all the original data from the Data Source into our dictionary's data structure. Data Loading must finish before our program starts interacting with users.
- **User Interface:** A program interface allows users to input search keys. This interface then displays returned results. Our program searches the dictionary's data (not the Data Source) for values associated with the search keys.
- **Identical Output:** Our program's output must be identical to the complete sample run's output.



1. In at least 1 full page, please explain the following in detail:

- Your analysis of the provided information and the provided complete sample output. Please think about Clients and Sales.
- What problem are you solving? Please explain it clearly then define it concisely. Please think about Problem Solving and Interviews.
- How do you store data in enum objects? And why? Please think about Data Structures and Data Design.
- Which data structures do you use/create for your dictionary? And why? Please think about Data Structures and Data Design.

2.

- Implement your program to meet all the requirements.
- In your assignment report, demonstrate your program to your grader/client.
- Does your program work properly?
- How will you improve your program?

**Answer:**

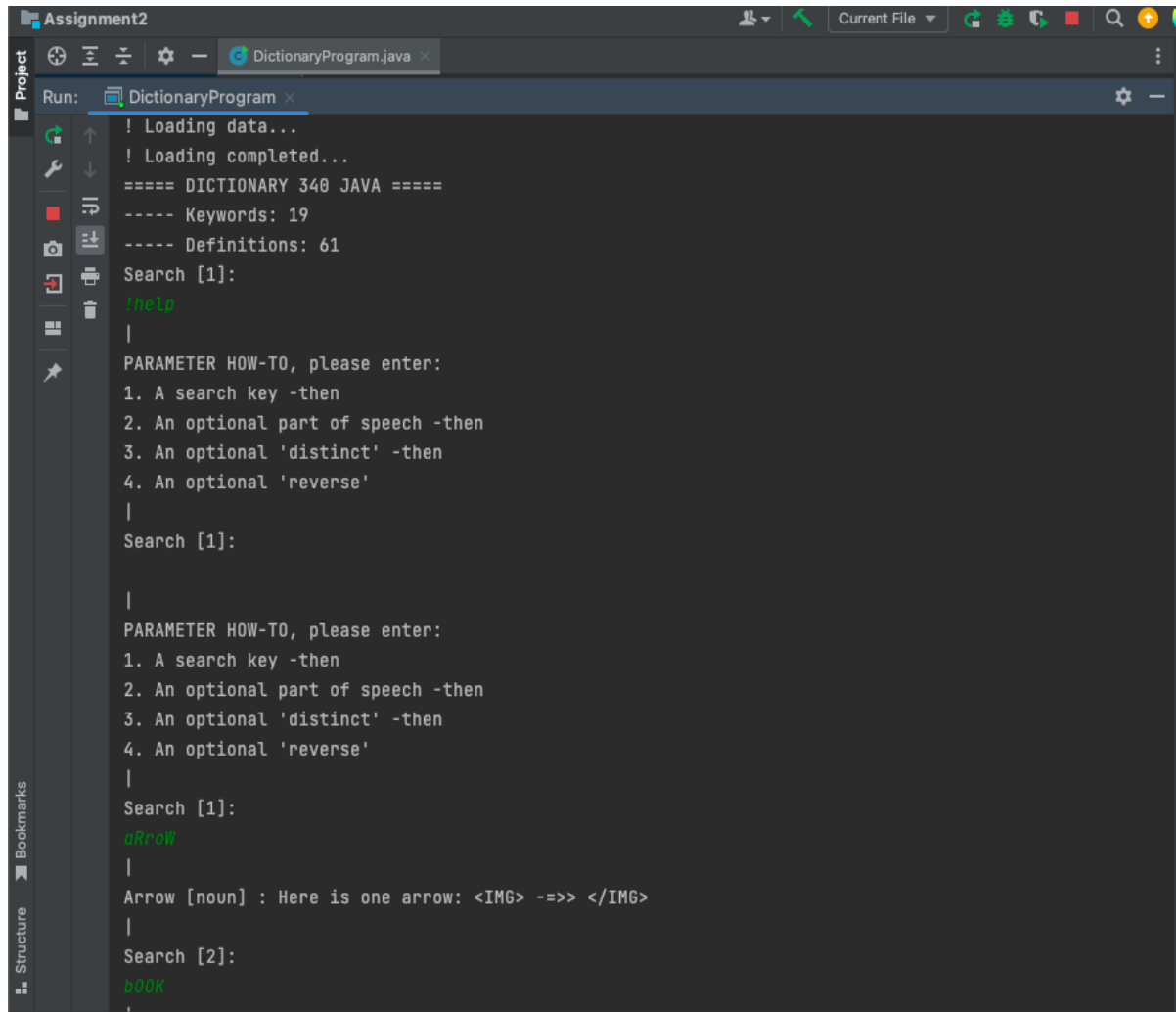
1. In this assignment, our program uses enums to represent dictionary entries and their corresponding parts of speech. It utilizes a multimap data structure from the Google Guava library to store the dictionary entries. The provided sample output demonstrates the functionality of the program. It showcases the loading of data, displays the dictionary title, and prompts the user for search input. The program allows users to search for specific keywords, optionally filtering by part of speech, distinct entries, or reversing the search results. The program then

outputs the matching dictionary entries along with their definitions and parts of speech. The problem being solved by this program is the efficient storage and retrieval of dictionary entries. It provides a convenient way for users to search for keywords and obtain their corresponding definitions and parts of speech. Enums are a suitable choice for storing data in this scenario because they provide a concise and structured way to represent the dictionary entries. The enum objects can be easily accessed using their names and provide a clear mapping between the keyword and its definition. The choice of a multimap data structure is appropriate for a dictionary because it allows efficient storage and retrieval of entries based on their keywords. It handles cases where multiple entries may have the same keyword, as seen in the provided code. The multimap allows for quick access to all the entries associated with a specific keyword. In summary, the program solves the problem of efficient storage and retrieval of dictionary entries. It utilizes enums to represent the data, storing them in a multimap data structure. This design choice allows for easy access to the dictionary entries based on keywords and provides a well structured and organized program for one to test and/or use. (Screenshots are at the bottom)

**2.** After finishing the program, It worked pretty well. Some of the formatting is off, like the search count doesn't update until you actually search a keyword (I was unable to fix it which is something I can improve on). Another big thing that could've been improved was the keywords stored in the enum datatype... I was unable to fix it so that I wouldn't have multiple books stored in it and so on. One last thing that could be improved was my code for distinct and reverse. I noticed that it would work properly on most searches, but on a couple it didn't (I think around

the 14th-15th search). Overall, I'm content with how it came out, because I learned more things, but improvements can always be made.

## Screenshots and Explanation:

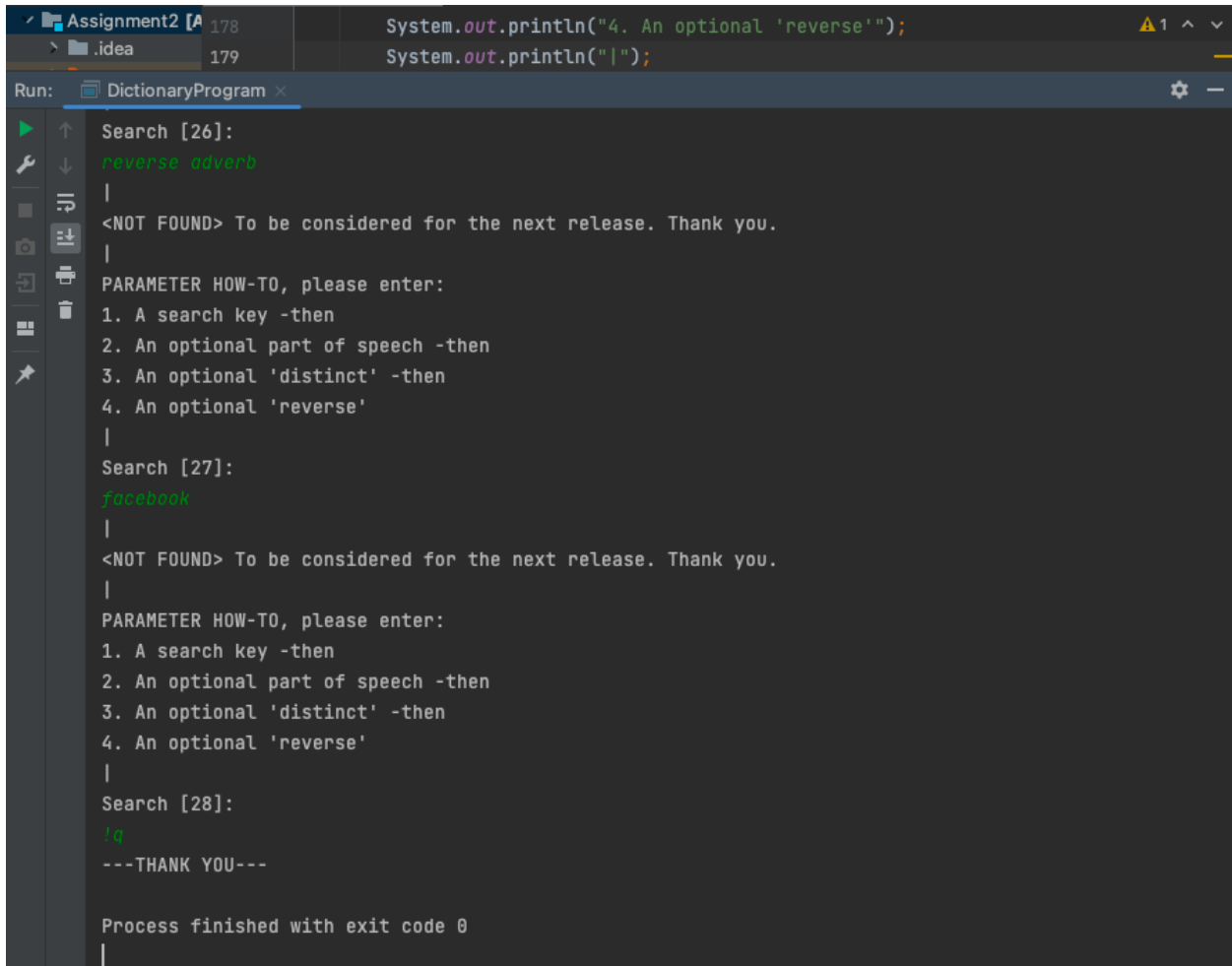


```

Assignment2
DictionaryProgram.java
Run: DictionaryProgram
! Loading data...
! Loading completed...
===== DICTIONARY 348 JAVA =====
----- Keywords: 19
----- Definitions: 61
Search [1]:
help
|
PARAMETER HOW-TO, please enter:
1. A search key -then
2. An optional part of speech -then
3. An optional 'distinct' -then
4. An optional 'reverse'
|
Search [1]:
|
PARAMETER HOW-TO, please enter:
1. A search key -then
2. An optional part of speech -then
3. An optional 'distinct' -then
4. An optional 'reverse'
|
Search [1]:
aRow
|
Arrow [noun] : Here is one arrow: <IMG> -=>> </IMG>
|
Search [2]:
b00K
|

```

Beginning of program



```

Assignment2 [A] 178
    System.out.println("4. An optional 'reverse'");
    System.out.println("|");
    DictionaryProgram x
Run:
Search [26]:
reverse adverb
|
<NOT FOUND> To be considered for the next release. Thank you.
|
PARAMETER HOW-TO, please enter:
1. A search key -then
2. An optional part of speech -then
3. An optional 'distinct' -then
4. An optional 'reverse'
|
Search [27]:
facebook
|
<NOT FOUND> To be considered for the next release. Thank you.
|
PARAMETER HOW-TO, please enter:
1. A search key -then
2. An optional part of speech -then
3. An optional 'distinct' -then
4. An optional 'reverse'
|
Search [28]:
tq
---THANK YOU---

Process finished with exit code 0
|

```

Towards end of program