

Course: CSC340.05

Student: Rajdeep Riar

Assignment #: 01

Assignment Due Date & Time: 09/18/18 @ 11:59PM

Part A

The first topic I would like to discuss is #2 on the guidelines, which is Consistency. Consistency means something being done a same way over time. This also refers to following a list of rules, guidelines, and following standard Java style and naming conventions. For example, giving class names and method names accordingly so it makes sense with the program. If the program is about fruits, your class names would be Apple or Orange, and not something like Honda or Acura.

Another important universal rule is to follow the standard style for coding. For instance, placing data declaration before the constructors and place constructors before the methods. This all is useful when other users try and read your program. If every programmer follows the same general guidelines, it is much easier to follow the flow of a program; otherwise only the coder who wrote it would know where everything is.

In general, providing a public no-arg constructor is typically used. Also preventing users from creating objects for a class or even preventing variables being used in another class, the coder should be using the keyword private to make everything private and accessible to that specific class. This way the variables are only accessible in that class and data is protected.

The second guideline I am choosing is #4 Clarity. What clarity is referring to is format, and how your program is written. Cohesion, consistency, and encapsulation are great guidelines to achieve design clarity. Each class and method should be so well written where it is easy to read and explain by any user.

Every class and variable should be independent to itself, and should not depend on other variables or methods. There should be no restrictions on how or when the user can use it. The code should be written where the user can use them in any order or combination.

Methods should be defined clearly so it gives the user an idea on what it does. For example, if the purpose of a method is to get the area of a circle, the name of that method should be something like `getAreaCircle()`;

Indentation is also a very major aspect in clarity. The best recommendation I can give for this is to use the shortcut Alt-Shift-F. This formats the program to where the entire code lines up very neatly and is very clear to read. The main focus of this guideline is to make the program user friendly for any user.

The third guideline I would like to discuss is #6 Instance Vs. Static. A variable or method that is dependent on a specific instance of the class must be an instance variable or method. However, if the variable is shared by all the instances in that class it should be declared static. The keyword static means it is shared among each other. It is common practice to reference static variables and methods from the class name, rather than a reference variable. This ties in with clarity, which improves readability.

Static data should not be passed by a parameter from a constructor for the data to be initialized. The best way to do this is by using setter methods to change the static data fields. Setters and getters are used for this sole purpose of changing to the static data field or simply returning what the data is.

Instance and static are integral parts of OOP and they often get used in place of the other. It is a common design error for the programmer to define an instance method that should have been static. One way to eliminate these silly errors is to truly research the meanings behind these words to really understand where one should be used.

Lastly, it talked about a constructor is always an instance because its sole purpose to create a specific instance which makes the program function properly. A static variable or method can be invoked from an instance method, but it is not true vice-versa.

The final topic in the guidelines I would like to discuss is #8 Interfaces Vs. Abstract Classes. An interface and an abstract class are both used to specify common behaviors for objects. Sometimes it can be a little tricky to know where to use which one. The guidelines states that if there is a strong IS-A relationship, it's best that to use classes. If there is a weak IS-A relationship, that indicates that an object has a certain specific property. In the case of a weak IS-A relationship, its best to use interfaces.

In an interface, the methods will have no bodies in it. When implementing the interface in another class, that is when you add the details of the method for it to function properly. In an interface, all the data fields should be private final static. All methods in an interface should be public abstract methods. An interface is commonly used when you want to specify the behavior of a particular data type, but not who implements its behavior. Take advantage of multiple inheritance of type.

An abstract class uses the keyword extends. If there mixture of unique and generalized methods the subclasses will inherit, then use abstract class as the superclass/parent class. An abstract class is commonly used when you want to share code among related classes, expect classes to extend abstract class have many common methods or fields, or require protected and private access modifiers.

Part B

Please refer to attached java file for Part B.

I used netbeans 8.2 to write the program. I had emailed you and asked you if I can make multiple enum classes to store the Data and you said it is fine as long as I store Raw Data inside ENUM classes only. Then from there I put the values in arraylists, and ultimately used a hashmap to store my enum data.

Program Analysis to Program Design

The provided information stated that there was no hard coding allowed. This meant that I couldn't force the program to produce certain outputs. For example, I wasn't allowed to make a bunch of if else statements and produce identical outputs. Another way of hard coding is by using switch statements and using case by case scenarios.

The prompt also said that we had to store the Data inside a set of enum objects. We were not allowed to combine them as storing three parts into one string. Once the program started, it loads all the raw data into our dictionary data structure. The user then uses a search interface where a value is output on the screen.

The problem we were trying to solve is to store data in enum objects, then use our experience with data structures, and choosing the best data structure for this specific problem. I stored my data inside separate enum classes. For example, made an enum class for CSC210, CSC220, CSC340, etc. In each class I had multiple enum objects which stored the parts of speech and descriptions of the objects, those were known as the values. I then put the values into separate arrayLists.

I then used the enum objects as my "key" and their values which were in arrayLists and stored all that data inside of a hashmap. I chose to use a hashmap as my data structure for my dictionary because it seemed the best fit since I had a key value pair. My parameters for hashmap were <String, List<String>>. From there I got user input and made that as a String array. I used the .split() function to split up the array for different amount to arguments. For example, the user can enter placeholder noun, and it would split up the two strings that way I can display the proper output.

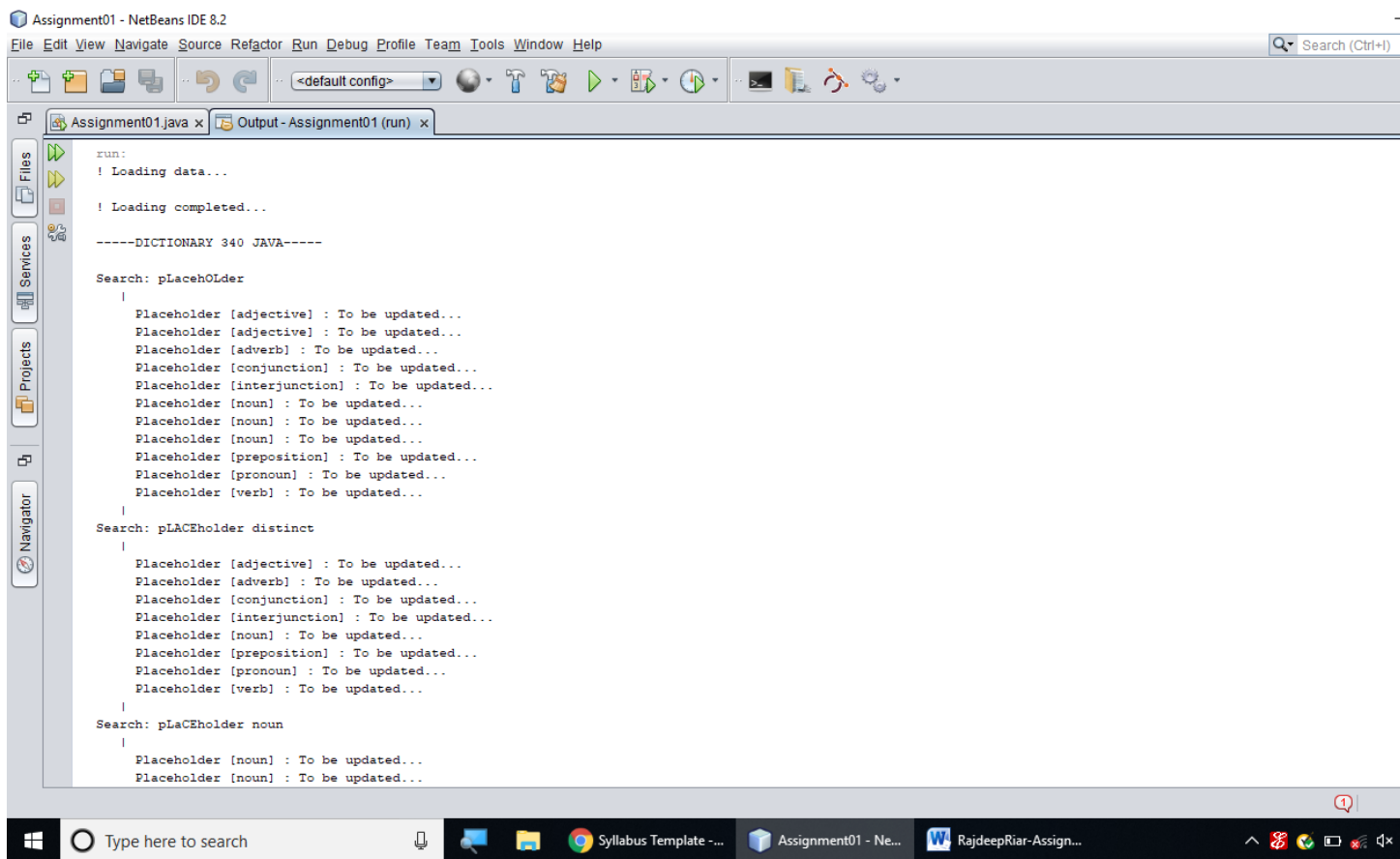
Finally, I used 3 arguments as cases on which the output should come out to be and everything worked out. My program works properly, and I am getting the exact same output as the sample output.

I could have improved my program by instead of making multiple enum classes, I could have stored everything inside one enum class and somehow manage the data that way. Of course that would have been more ideal, but in this specific case I knew how to implement it correctly with multiple enum classes so I chose this route. Things did confusing because I had so many class names to work with, but writing them down on paper and tracing which goes to where was helpful. However, I understand

putting them inside one enum class would be better and easier. I have 6 classes this semester and I work 30 hours a week, so I just wanted to complete this assignment and move on.

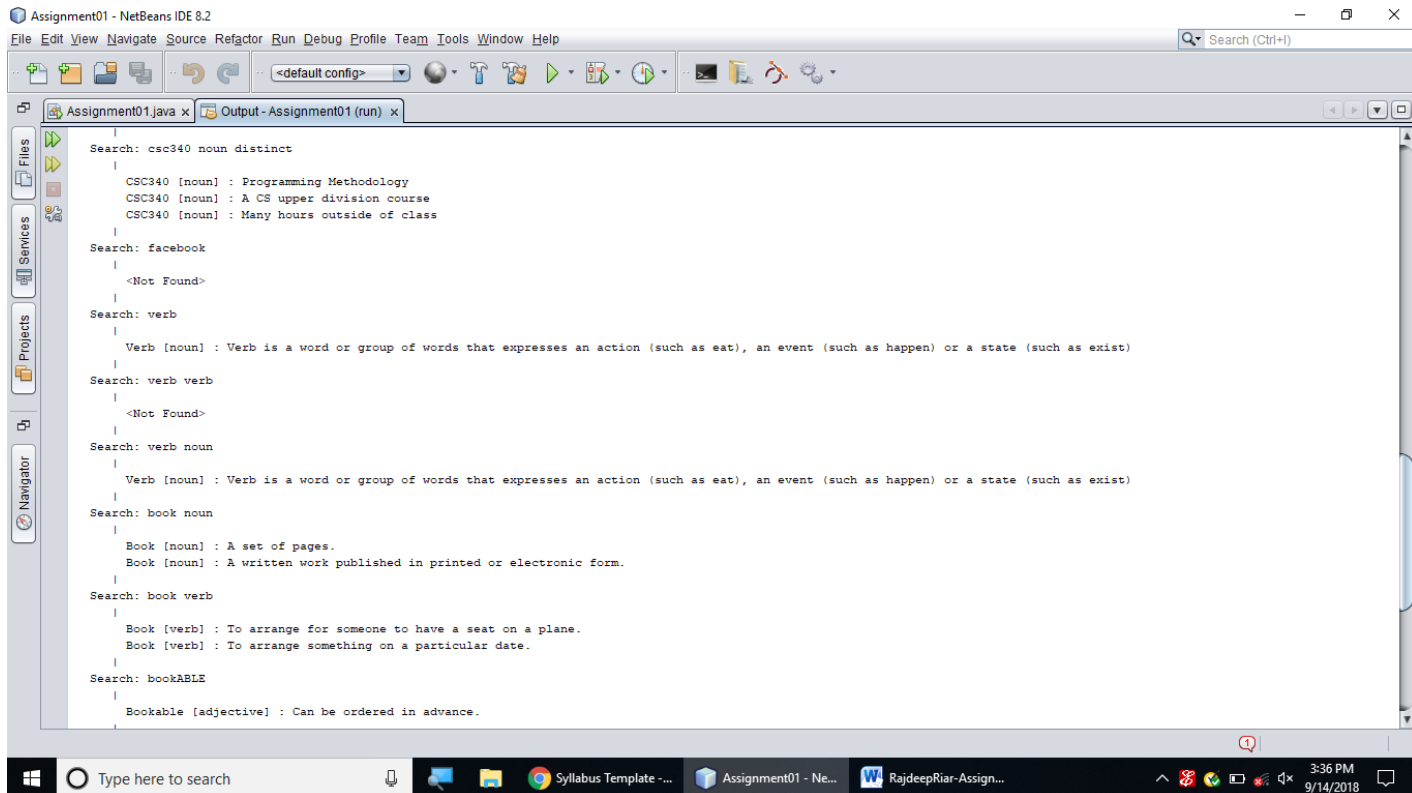
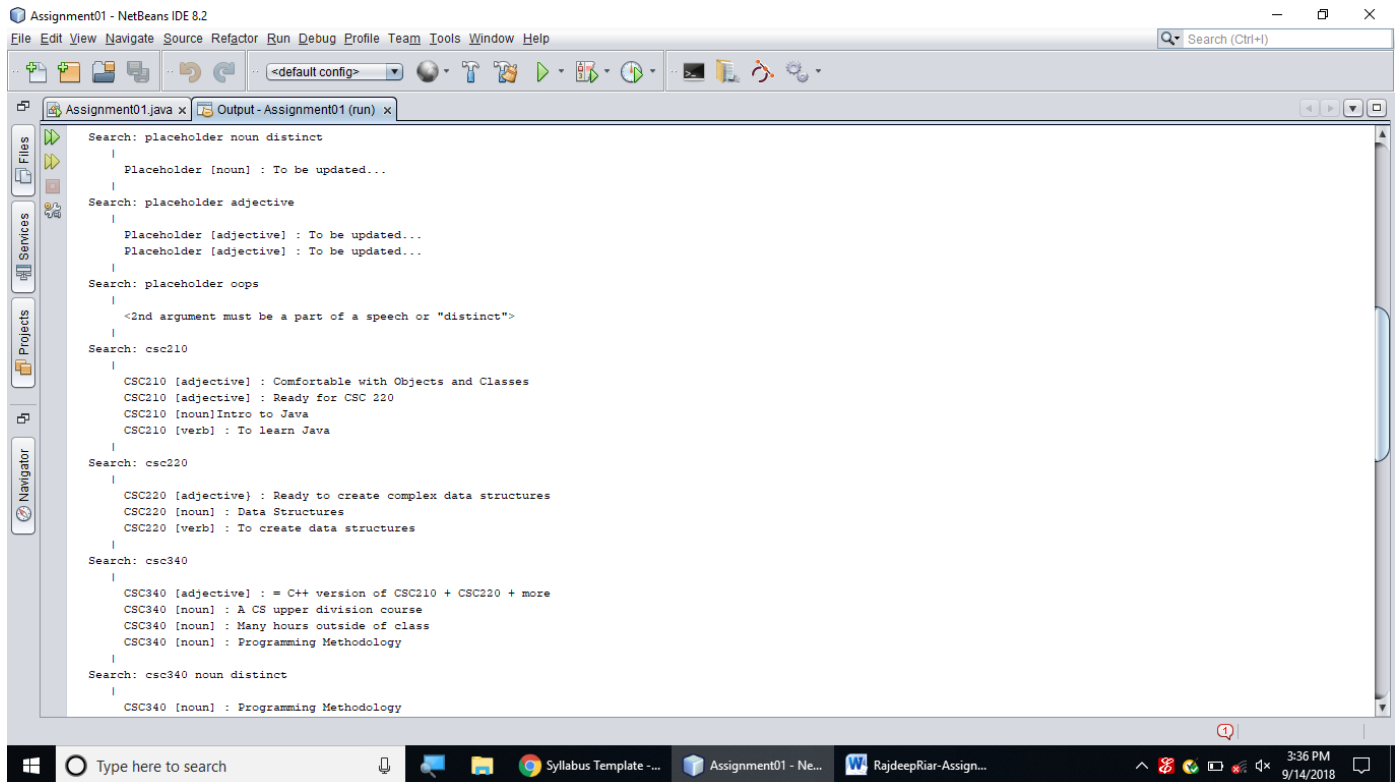
I took professor Ta over the summer and we made a dictionary last semester, so I am a bit experienced in this topic. However, last semester it was fine if it was hard coded..

Below are screen shots of the sample run:



```
run:
! Loading data...
! Loading completed...
-----DICTIONARY 340 JAVA-----

Search: pLacehOlder
|
Placeholder [adjective] : To be updated...
Placeholder [adjective] : To be updated...
Placeholder [adverb] : To be updated...
Placeholder [conjunction] : To be updated...
Placeholder [interjunction] : To be updated...
Placeholder [noun] : To be updated...
Placeholder [noun] : To be updated...
Placeholder [noun] : To be updated...
Placeholder [preposition] : To be updated...
Placeholder [pronoun] : To be updated...
Placeholder [verb] : To be updated...
|
Search: pLACEholder distinct
|
Placeholder [adjective] : To be updated...
Placeholder [adverb] : To be updated...
Placeholder [conjunction] : To be updated...
Placeholder [interjunction] : To be updated...
Placeholder [noun] : To be updated...
Placeholder [preposition] : To be updated...
Placeholder [pronoun] : To be updated...
Placeholder [verb] : To be updated...
|
Search: pLaCEholder noun
|
Placeholder [noun] : To be updated...
Placeholder [noun] : To be updated...
```



Assignment01.java x Output-Assignment01 (run) x

Files
Services
Projects
Navigator

```
Search: book noun
|
|   Book [noun] : A set of pages.
|   Book [noun] : A written work published in printed or electronic form.
|
Search: book verb
|
|   Book [verb] : To arrange for someone to have a seat on a plane.
|   Book [verb] : To arrange something on a particular date.
|
Search: bookABLE
|
|   Bookable [adjective] : Can be ordered in advance.
|
Search: conJUNCTION
|
|   Conjunction [noun] : Conjunction is a word that joins words, phrases or sentences, for example 'and', 'but', 'or'.
|
Search: interjection noun
|
|   Interjection [noun] : Interjection is a short sound, word or phrase spoken suddenly to express an emotion. Oh!, Look out! and Ow! are interjections
|
Search: pronoun verb
|
|   <Not Found>
|
Search: adjective noun
|
|   Adjective [noun] : Adjective is a word that describes a person or thing, for example big, red and cleaver in a big house, red wine and a clever idea
|
Search: !Q
|
|   -----THANK YOU-----
|
BUILD SUCCESSFUL (total time: 2 minutes 2 seconds)
```

Type here to search



Syllabus Template -...

Assignment01 - Ne...

RajdeepRiar-Assign...

3:36 PM
9/14/2018