CS 471: Software Engineering
Spring 2018
Homework 4 – UML

Due date: Wednesday, April 18, 2018 (at the end of day)

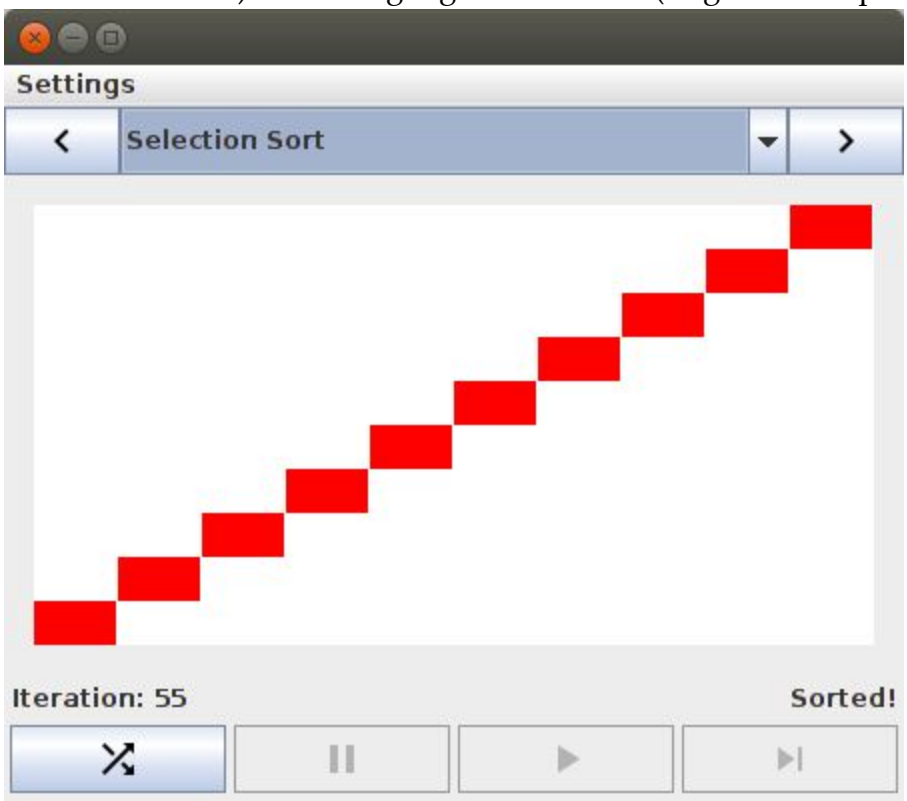
1. **Brief Description**

This assignment involves creating UML class diagrams and sequence diagrams in a scenario you may encounter as a software engineer. You will need figure out how a system works and document your findings using a UML class/sequence diagram. Additionally, you will use these documentation types to propose code changes based on a given feature request.

This assignment does not involve writing any code. The goal is to expand your skills in analyzing unfamiliar code, documenting the code using UML diagrams and designing proposed changes to the code by modifying the created UML class diagrams and sequence diagrams.

2. **Scenario**

You were assigned to maintain a small (~1KLOC) Java project (see image below) that visualizes different sorting algorithms. The software currently supports 3 sorting algorithms (Bubble Sort, Insertion Sort and Selection Sort) and 2 language translations (English and Spanish).



The project was initially created by Leeroy, a novice developer that made some questionable (i.e., not-optimal) design decisions. The resulting architecture requires a substantial amount of work to implement most types of features.

Leeroy was moved to another project and is not able to assist you with any knowledge transfer activity to get you up to speed with the current status of the project. The project README lists the existing documentation, which is clearly insufficient. Moreover, Leeroy never documented the structure of the program (e.g, UML class diagrams). Your first task as the maintainer is to understand the current class structure (by generating a UML class diagram) and object interactions (by generating a UML sequence diagram).

Your second step will be to update these UML diagrams with an improved, extensible and flexible design that would easily scale to adding multiple sorting algorithms and language

translations.

3. **Source Code**

You can clone the source code of the project from GitHub (https://github.com/BoiseState/CS471-Assignments-UMLSorting). The repository includes a README that explains how to import, build and run the project in Eclipse. However, you are free to use any other IDE.

A quick `git log` in the repository will illustrate what features were added to the repository and in which order.

The current implementation is full of hardcoded `if-else` statements to return the appropriate translation of a given piece of text. These `if-else` sections are distributed across the code, making each additional translation more difficult to implement and more error prone, contributing to the already existing technical debt of the project.

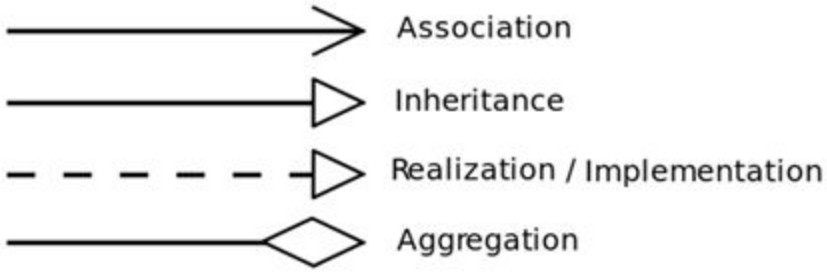
4. **Understanding the high-level structure of the project (Generating a UML Class Diagram)**

You need to create a UML Class Diagram of the project that uses CS471_S18_HW4_UMLClassDiagramTemplate.pdf as a starting point. This diagram was created using Lucidchart and if you prefer to **not hand draw** the relations on the PDF template:

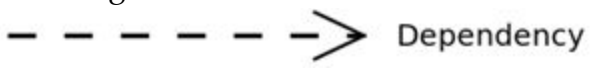
- you can download the Visio format CS471_S18_HW4_UMLClassDiagramTemplate.vdx of the diagram on your local computer,
- you can login with your (BSU) Google Account to Lucidchart and
- import the Visio `vdx` file into Lucidchart, which will allow you to edit the relations online.
    - NOTE: This (ahem, stupid) "Visio" workaround is needed because Lucidchart does not allow you to make copies of a shared lucidchart diagram (i.e., it only allows you to view it).

The generated UML Class Diagram:

- should not modify the layout of the classes/interfaces (i.e., do not rearrange their position)
- should not add/remove any new/existing classes or interfaces
- should not add any attributes or methods to the classes/interfaces
- should emphasize only the following relations (in other words, you only need to capture in the diagram Association, Inheritance, Realization and Aggregation relations between classes and interfaces):



    - NOTE: The template includes a Dependency relation to show the relationship between the class `Main` and the classes `Model`, `View` and `Controller`. Dependency relation is a weak, temporal relation that arises from a local parameter or variable. You do not need to include additional Dependency relations for this assignment.

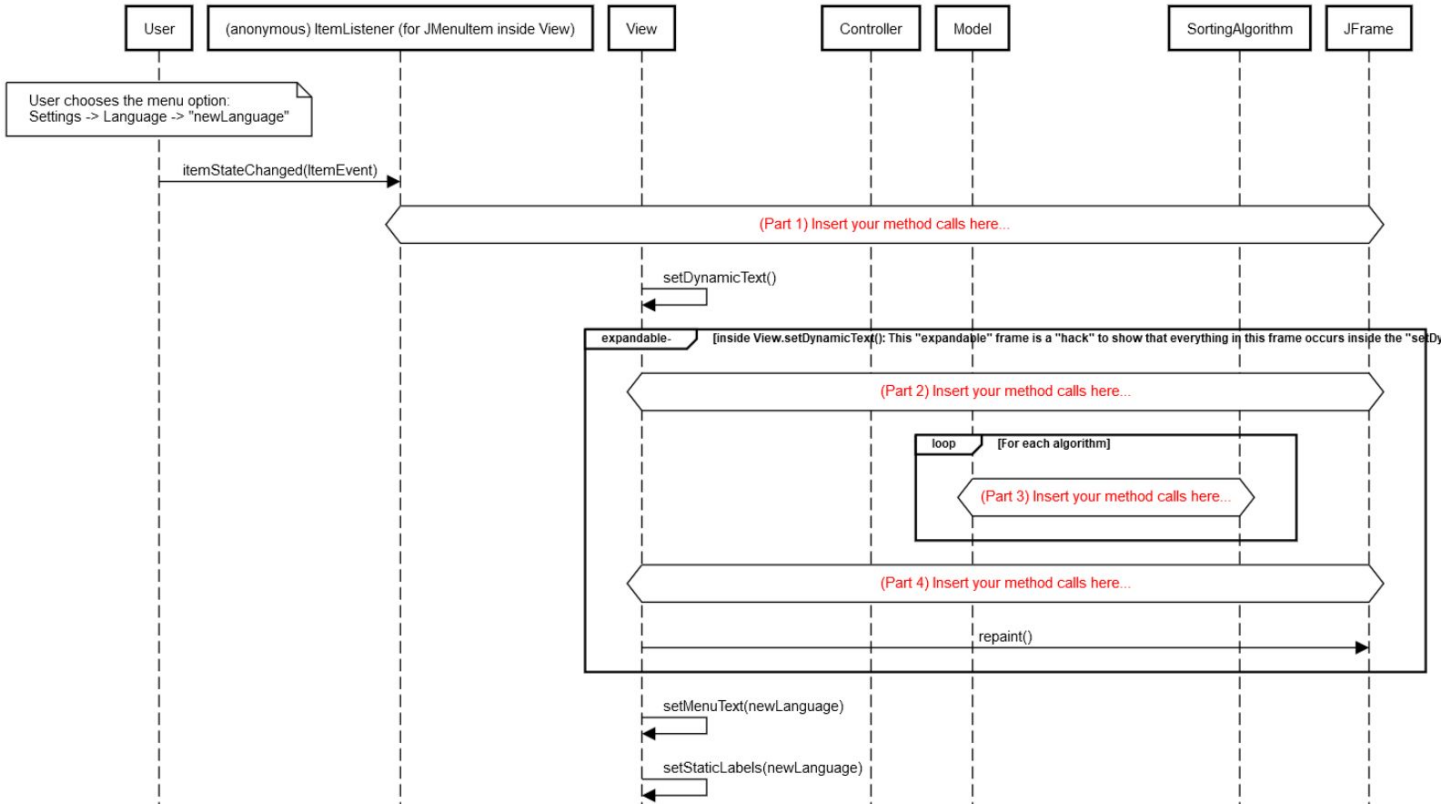


- can be hand drawn and scanned, or generated using an online tool (e.g., Lucidchart)
    - You can even print the CS471_S18_HW4_UMLClassDiagramTemplate.pdf, hand draw the relations and scan (or take photo) of the diagram
- should have in total around 17 Association, Inheritance, Realization and Aggregation

relations and only 3 Dependency relations (which are already in the template)

5. **Understanding a dynamic interaction scenario between objects (Generating a UML Sequence Diagram)**

There are numerous interaction scenarios that can be captured using a UML Sequence Diagram for this project. However, for this assignment we will focus on creating a UML Sequence Diagram for the **use case of a user changing the language** (i.e., if a user changes the language from English to Spanish, what sequence of object interactions occur in the program?). You can use the following sequence diagram as a starting point.



This diagram was created using sequencediagram.org using the following input text:

```
    participant User
    participant (anonymous) ItemListener (for JMenuItem inside View)
    participant View
    participant Controller
    participant Model
    participant SortingAlgorithm
    participant JFrame

    note over User:User chooses the menu option:\nSettings -> Language ->
"newLanguage"
    User->(anonymous) ItemListener (for JMenuItem inside
View):itemStateChanged(ItemEvent)
    abox over (anonymous) ItemListener (for JMenuItem inside View),JFrame:<color
#red>(Part 1) Insert your method calls here...</color>
    View->View:setDynamicText()
    expandable- inside View.setDynamicText(): This "expandable" frame is a "hack" to
show that everything in this frame occurs inside the "setDynamicText()" method
    abox over View,JFrame:<color #red>(Part 2) Insert your method calls
here...</color>
    loop For each algorithm
    abox over Model,SortingAlgorithm:<color #red>(Part 3) Insert your method calls
here...</color>
    end
```

```
    abox over View,JFrame:<color #red>(Part 4) Insert your method calls
here...</color>
    View->JFrame:repaint()
    end
    View->View:setMenuText(newLanguage)
    View->View:setStaticLabels(newLanguage)
```

The generated UML Sequence Diagram:

- should not modify the layout of the objects (i.e., do not rearrange their position)
- should not add/remove any new/existing objects (i.e., should have in total 7 objects, including `User`)
- should have in total around 16 message calls (5 of these calls are already included in the template)
- should have in total around 1 loop element (already illustrated in the template)
- can be generated using any online tool as long as it is clean and readable. However, if the tool supports a 'hand drawn' style, do not use that option because it is harder to read.

6. **(EXTRA CREDIT) Designing Proposed Changes by Modifying the Created UML Class and Sequence Diagrams**

Assume that the following features will need to be added to the software in the near future:

- Add a description for each sorting algorithm
- Add translation support for another 10 languages

The current application structure makes it very difficult to add other language translations. Think about how the application should be re-structured/re-factored to make it scalable to adding new translations (e.g., French) and an additional text field (e.g., a description of each algorithm) to the GUI that must support each translation.

Update the previously generated UML Class and Sequence Diagrams to indicate how the code should be restructured to facilitate adding the new features.

If some aspect of your design requires additional clarification, add notes to your diagrams. However, a large number of notes containing textual explanation is an indication of a diagram containing a non self-explanatory design, or a "draft".

You are not required to implement these features, only to design them.

7. **Submission**

Submit via Blackboard (see HW4UML assignment):

- a pdf (or image file) named `CS471_S18_HW4_ClassDiagram_[LastName].pdf`, based on the CS471_S18_HW4_UMLClassDiagramTemplate.pdf template found under "Homework Assignments" in https://piazza.com/boisestate/spring2018/cs471/resources. See Section 4 for more details about the class diagram.
- a pdf (or image file) named `CS471_S18_HW4_SequenceDiagram_[LastName].pdf`, based on the template from Section 5. If you use the sequencediagram.org tool to generate the diagram, include the URL (use "create URL to share" feature on the left side of the sequencediagram.org tool)

- (EXTRA CREDIT): an updated UML Class Diagram (`CS471_S18_HW4_ClassDiagramExtraCredit_[LastName].pdf`) and Sequence Diagram (`CS471_S18_HW4_SequenceDiagramExtraCredit_[LastName].pdf`) using a similar format as the original diagrams

8. Grading Rubric

The maximum points for this homework (representing 8% of the final grade) is 100 (130 with Extra Credit), and the points are distributed as follows:

Item	Points
UML Class Diagram is clean, readable and adheres to the specifications in Section 4	70
UML Sequence Diagram is clean, readable and adheres to the specifications in Section 5	30
(EXTRA CREDIT) Updated UML Class Diagram (Section 6)	15
(EXTRA CREDIT) Updated UML Sequence Diagram (Section 6)	15