EE364 Advanced Programming

Electrical and Computer Engineering Department
Engineering College
King Abdulaziz University

Final project
Fall 2022

Simulating Real-World Problem

Objective:

- Apply what you have learned in this course into a real-world problem.
- Practice on designing an OOP program from scratch.

Introduction:

Writing a simulation program is one of best ways to apply what you have learned in one big project. The simulation you are going to code has to simulate a real-world problem. In other words, you have to represent all the aspects in the problem you are simulating in Java objects and variables. The random changing in the simulation is what makes results from your simulation close to reality.

Therefore, for the 1st phase in your simulation problem you have to make the problem easier to be noticed by whoever is looking to your simulation output. The company manager, the warehouse manager or factory manager should be convinced that there is a problem if he saw the output of your simulation that simulate a problem in his organization. In general, you are trying to show the problem and the source of the problem.

In the 2nd phase, you will try to solve the problem. In other words, you will change the code to make the results better. The company manager or whoever is looking at your 2nd phase output should realize which of the factors that has the most impact in solving the problem.

Finally, you should show the improvement percentage. It should show how much your simulation results have improved from phase1 to phase2.

Phase#0 Project design:

In order to start working on your project, you have to find a real-life problem that you can simulate in an OOP programming language. The real-life problem should have many factors that affect it's results and it should be interesting to you to know the sources of this problem. As you know, every real-life problem has many factors that affect it.

After you have found the problem, you want to simulate, write the problem definition. In the problem definition, you should focus on presenting the problem and making the problem clear for the reader to understand. A clear problem definition that has no solving steps is a good starting point to write a simulation program.

In general, a real-life problem consists of two main parts. The first part is the real-world data. In your program, you do not have to look for a real-world data, but you should make this data a random variable that can change for a period of time. Doing this, will ensure that you run all the scenarios of the problem. The second part of the real-life problem is the data processing. Processing data in the wrong way leads to a problem that can be prevented if the process was right. Therefore, the source of the problem can be from either the data or the process of running the data.

Since you have to start with end in mind, you have to think how this problem is going to be solved. solving the problem can be achieved by adding more information or changing the process that handled the information.

After you are done writing your problem definition and the solution of the problem, you should start brainstorming all the variables needed in the program. Those variables should be related to the problem and they are not additional information that doesn't affect the problem. Of course, you can use some organizing variables but do not add too much of them. Whenever you think that you are out of new variables, go back and read the problem definition. this will help you stay focused to the problem you are simulating.

Since you have all the variables that should be in your program, you should start thinking about the classes that can hold these variables. As a Java programmer, you know that classes are a data type That holds variables related to each other. Therefore, you should think about the variables that can be joined together in one group and give them a name. This step is your program infrastructure, so try your best to do it right from the first time.

Since missing up the variable distribution and the class choosing is critical in the writing of your program, make sure that you have a good object-oriented programming structure. In other words, the classes and their data fields should be similar to the real-world problem you are trying to simulate. All the objects from those classes should match all the objects in the real-world problem you are simulating. Therefore, you should always ask yourself if you are missing some important variables that can be a key factor in changing the output.

Having a list of classes, you should start drawing the UML diagram. There are some important concepts that you have to make sure your program has at this point. one of them, is object-oriented programming where each object represents and other objects in real life problem. In addition, inheritance and polymorphism are another two concepts that are important to have in your program. These two concepts the play a big concept in writing an efficient code. Finally, your project must also have the abstract and interface concepts since they improve the quality of your project design.

Notice please use a good UML generator like **lucidchart.com**

Now you should be almost done with the phase zero. submit your proposal and prepare yourself to discuss your project design. You should not start writing codes before you get the proposal approval.

Phase#1 Problem simulation

In this phase of the project, you will start writing the code. The first step, is to convert the UML diagram into Java classes. You should fill all the classes with the pseudocode starting from the main method. The main method has the main loop which is the engine of the program. This loop should continue to work from the beginning of the simulation until the end. In general, the main loop has all the main steps that your program goes through. all the detailed steps and sub tasks are handled and methods in other classes.

After creating all the classes needed, the work can be distributed among all team members equally every team member knows that his code needs to call other methods in another classes. A team member does not have to know the implementation of the method. it is enough to know the input, output, and the name of the method. This is possible because all the team member agreed on how each method should work while writing the pseudocode.

Notice every team member should make sure that he has equal load like other team members since this will affect his project grade.

Each team member has some classes that he is going to work on and the pseudocode is clear for each method in all classes. Therefore, it should be easy for each team member to convert the comments into Java code. as a team member, it is necessary after writing each class to test your code before sharing it with other team members. You can make your own main method and test your class separately.

After each member of the team is done writing his own code, all the classes should be gathered together in one Java project. Gathering the classes together is tricky sometime and can cause some problems, so you need to pay attention for all the assumptions you have made, including packages name. If there is no compilation error then you can step into Testing the output. You should make sure that the output is showing the problem and the source of the problem. if the problem is not clear then you need more analysis. Also, if the source of the problem is not clear then you might need a table that shows all the output scenarios. The output should make it easy for the decision maker to see where the problem is coming from by showing all the changes in the factors that affect the results. you might consider having a general percentage for some of the major analysis data and the output.

Notice at this point you might start working as a group. you should have a daily meeting where you test and debug the code together.

If the whole team is satisfied with the output and the code is running flawlessly, then you're ready to submit the first phase of your project. in this submission I'm going to look for the problem in the output. if your project output is well designed and the data Your project showing has been chosen carefully, then it is easier for me to say that your project is simulating the problem. Moreover, I will be looking at the code and ask you about some command line you wrote, so be prepared.

Phase#2 Problem solving

After you wrote a code that simulate a real-world problem and you pass phase one presentation, then you can start modifying your code to solve the problem. as we said before, you can solve the problem and improve the results by adding or changing some of the factors that affect the output.

If you are able to develop a solution for the problem, then you should work on perfecting the output. Since this output is the final output of the program, it shouldn't look different than the output from phase#1. However, you can add some analysis results that can show the improvement you have made in solving the problem. In fact, you should have an improvement percentage that shows the improvement between the situation of the problem in phase #1 and phase #2. Finally, try to run your code many times and record at least the output of 10 program running. All the 10 output files should show that the percentage of your improvement is increasing from phase one to phase two. If you are satisfied with what you have achieved, then you can submit your final project

Notice there will be bonus grades for GUI development.

What to turn in

Project design submission:

- Project proposal form (in .pdf)
- UML diagram (use this website lucidchart.com)

Problem simulation submission:

- The whole java project file
- The updated UML diagram
- Javadoc for the whole project (the author's name should hold the team member who wrote the class)

Problem solving submission:

- 10 text files of the program output showing the improvement percentage each time.
- The whole java project file
- The updated UML diagram
- Javadoc for the whole project (the author's name should hold the team member who wrote the class)