Textbook Assignment 1

1. What do requirements mean in software development projects? What type of requirements defines what the program needs to do?
   In software development projects we have two requirements:
   a. Functional: what a program needs to do
   b. Non-functional: the way the functional requirements need to be achieved such as performance, usability, and maintainability
      Functional requirements define what a program needs to do

2. What do design constrains mean in software development projects?

   They are statements that constrain the ways in which the software can be designed and implemented. They are usually made by the customer and the user instead of by architects and designers.

3. Which decisions are those taken by the software engineer about the best ways (processes, techniques, and technologies) to achieve the requirements?

   Those related to nonfunctional requirements.

4. What is the difference between software validation and verification? In a software development project, who will do validation? Who will do verification?

   a. Software verification tests are tests run by the developers internally to determine if a program works, is correct, and behaves the way it should. Verification tests determine whether the system build is correct.
   b. Software validation test could be done by showing the clients a hand-drawn screens of the problem-solution. Validation tests determine whether the developers are building the correct system for the client. They are done by clients, or somebody on their behalf

5. List three of the typical kinds of non-functional requirements? Give a brief explanation about each non-functional requirement.
   a. Performance requirements because performance is always an issue. The program needs to finish most or all inputs within a certain amount of time.
   b. Modifiability requirements: if the program is to be used only once, modifiability won't be a big issue. However, if it will be used a lot of times, making it easy to maintain and modify the program will be something to worry about

c. Usability requirements: the users have different backgrounds, education, experiences, needs, and interaction styles that need to be considered in the development of the software.

6. Using one of programs that you have developed as example, describe the sequence of activities during development of a software program.
    a. Compute average program. The first thing that was done was reading what is needed and how should I go on developing it. It said that there should be three classes: 1- AverageApp 2- AverageCalculator 3-AverageUI. I read the requirements for each class, what functions it is supposed to be able to perform and their relationships with each other. It said to write the program using Java. So, I did what was asked.

7. Discuss whether you think that a programming language constraint may be viewed as a requirement. Explain why you think so.
    a. It should be viewed as a requirement. Although I agree that the client should never get involved with how the program works as long as it works, but if the client wishes to have things done in a certain way, then they should be done in that way. Meaning if the client asked for a program to be written in Python and the developers decided to do it in Java then the program should not be accepted by the client.

8. Describe the design of "compute average" program that you developed for lab 1 in the following way:
    a. Name of each class, and responsibility of each class
    AverageCalculator: is the class that is responsible for computing and calculating the average of given numbers.
    AverageUI: is the class that is responsible for creating an AverageCalculator object and getting the inputs needed from the user by using a scanner and passing these inputs into an array of a given size also given by the user. After that, sends the input in their respective types and variables. Then, it passes those variables to the AverageCalculator object.
    AverageApp: is the main (driver) class for this program. It is supposed to create an AverageUI object and call a method that exists in AverageUI

    b. Class relationships in terms that a class depends on or uses another class
    AverageApp has an instantiation relationship with AverageUI. (AverageUI depends on AverageApp)
    AverageUI has a direct association relationship with AverageCalculator, therefore, we did not need to draw an instantiation relationship.

c. Method names of each class, and responsibility/functionality of each method in each class

computeAverage method in the AverageCalculator class takes an array or int and calculates and returns the average of the numbers present in the array.
handleIO method in the AverageUI class creates a scanner object and asks the user for the size of the array that will store the ints that the user will also be asked later to input. Then the method will print the average of the given array with the given size using a previously created private AverageCalculator object.
main method in AverageAPP: this method is the driver class for these three classes. The main method is present in the AverageApp class. It creates an AverageUI object which is going to be used to call the handleIO method present in AverageUI.

d. Full signature (method name and parameters) of each method in each class

AverageCalculator(numbers : int[*]) : double
HandleIO()
Main(args : String[*])