Homework 1

Chimzim Ogbondah

1. No silver Bullet
   1. Yes, I think what Fred Brooks said about software will hold true. In chapter 16 he discusses the complexity of software and how it is different from other things. With buildings or hardware when it is update or changed it tends to follow the already given structure but with software this isn’t the case. Software can have many dependencies that make adding on or upgrading not linear. I also believe while some issues about a silver bullet for software are addressed with higher level languages, I believe since there is no management or cohesive plan to develop/improve software will keep it very complex.
2. Code fragments
   1. GameRules
      1. It violates information hiding because the user of the gameRules doesn’t need to knows how the list of players is kept and the type of player
      2. It is giving insight into the data type and storage of the players in timeout
      3. String getPlayersInTimeOut(); String getNextPlayerTurn();
   2. DocumentContainer
      1. It violates bad abstraction because the user is creating a container class just for document elements (data class)
      2. It is suggesting that data/information is being used utilized else where.
      3. **Create an abstract container interface (generic) adding observers to view what has been added to the “children” vector**
   3. EmailMessage
      1. It violates information hiding by getHeaders() method
      2. It is giving insight on how the headers are stored in the class
      3. addEmailObserver(emailObserver listner); removeEmailObserver(emailObserver listner);
   4. Timer
      1. It violates bad abstraction because it is giving insight into how the timer operates
      2. The user doesn’t need to know that the object type is Time which allows for dependency issues since other classes or objects will then begin to assume the type of the timer class. This would cause problems if the Timer class were to change its methods
      3. String getCurrentTime(); int MinutesToNextAlarm();
3. Refactoring
   1. The refactored code exhibits examples of abstraction from DataLogger 🡪 DataLogger2
   2. Code Refactoring
      1. +Abstraction: in the storeSmaples function it had the same section of code that was used to store temperature and humidity but once it was refactored it was turned into two functions where one took in the sensor and the other called that method passing in the sensors. This is good because it adds modularity. If the code needed to be changed (more sensors) all that would need to be updated is storeSamples with it calling the storeSamples method with the new sensor
      2. -Information Hiding: The database and the sensors aren’t private variables and so anyone can access the attributes of the variables outside of their intended method manipulation.