After tests are generated, The code should be done as well. Each time the code passes the test, if it is necessary students should refactor the code. After each test, students should commit the code to the repository. Once code base becomes larger, they should choose refactoring technique and justify it. It can be done in the form of comments. The main focus is how students learn refactoring techniques.

Refactoring guidelines:

* **Isolate changes.**  How is it possible to modify one part of the method or object, which consists of a several parts? At first, you should change variable part. You might notice that after you isolated change and made change to the code the result became so trivial so you can cancel an isolation. For example, if you noticed that there is one action within findRate() method – the return of the field value. We can directly access the field instead of accessing the method. As the result, findRate() method can be removed However, such changes can not be implemented automatically. Try to find a balance between related to the cost of usage of additional method and benefit which is brought by new concept.
* **Extract method.** How is it possible to make a long and complicated code easy to read? Extract a tiny part of long method into separate one and access that part from long method
  + Outline the fragment of the code, which can be put into separate method. Good candidates are the bodies of loops, loops and the branches of conditional operators.
  + Make sure that inside the fragment there is no assignment of values to the temporary values, which are declared outside the scope of visibility that match to that fragment
  + Copy the code from old method to the new one. Compile it.
  + For each temporary variable or parameter of initial method used in new method add the parameter to the new method
  + Make sure that at necessary place the old method accessed the new one

This method is used to understand a complicated code because you simplify it. It is also used to get rid of code duplication when two methods have similar pieces of code

* **Inline method.** How can you simplify a code in case when it becomes hard to observe the sequence of transfer control from method to method? Replace the access to the method with the code of that method
  + Copy the code the method to the clipboard
  + Insert the code of the method instead of access to the method
  + Replace all formal parameters with factual parameters. If, for example, you transfer reader.getNext() which is the expression that has a side effect, be careful and assign the received value to the temporary variable.
* **Move method.** How can you relocate the method to a new place where it should belong. Add it to the class where it should belong and then access it
  + Copy the method into clipboard
  + Insert method into the target class. Assign it a necessary name. Compile it
  + If within the method there is an access to the initial object. Add the parameter which will pass the object inside the method. If within the method there is an access to member-variables of initial object, pass them as the parameters. If inside the method member-variables are assigned values, you should refuse from the idea of transferring to new object
  + Replace the body of the initial method with the access to new method
* **Method Object.** How can you implement a complex method which uses several parameters and local variables? Convert method into a separate object
  + Create class with the same number of parameters as the original method
  + Convert local variable into instance variables of new class
  + Define new method run() inside new class. The body of that method will be same as body of the original method.
  + In original method create new object and access to the method run() of that object