At first students read the rules of the game. This stage can be called a requirements gathering. The main focus for them is understand rules and all possible variants. Then they can generate requirements. I assume students don’t have an extensive experience in programming. Therefore, it is best for them to generate more requirements. This approach is justified in Kent Beck’s book. In his book, Beck states the less steps you take, the less mistakes you can make while doing refactoring. If you take longer steps, the refactoring becomes tedious because you need to change bigger lines of code. Another risk is that students can make mistakes while they are refactoring. The requirements should follow certain order

* **Basic functionality** should be covered first. For example, Students should initialize basic structures. Students should make a domain model and visualize relationships among classes. Each class have a certain behavior in the game. For example, game consists of many frames. It should be shown in domain model. The tests should confirm the initialization of classes and their behavior.
  + 1. Each turn of a bowling game is called a frame. 10 pins are arranged in each frame. The goal of the player is knock down as many pins as possible in each frame. The player has two chances or **throws**, to do so. The number of pins knocked down in that throw gives the value of a throw.
    2. An ordinary frame’s score is the sum of its throws.
    3. A single game consists of 10 frames
    4. The score of a bowling game is the sum of the individual scores of its frames
* **Common rules.** Students should implement the tests related to common rules of the game. Each rule should be written in the form of test. For example, spare can be written as the test. The behavior of the class should be adapted to the test. The adaptation may include refactoring. Students should use extract method technique in order to avoid code duplication and simplify complex segments of code.
  + 1. A frame is called a strike if all 10 pins are knocked down in the first throw.
    2. A frame is called a spare when all 10 pins are knocked down in two throws.
* **Combination or Edge cases.** While game is played, various extreme situations might happen. It is important to cover them as well. For example, there can be multiple spares or multiple strikes. It is important to analyze all such cases.
  + 1. A strike can be followed by a spare.
    2. Two strikes in a row are possible.
    3. Two spares in a row are possible.
    4. When a game’s last frame is a spare, the player will be given a bonus throw.
    5. When a game’s last frame is a strike, the player will be given two bonus throws.
    6. Further bonus throws are not granted when a game’s last frame is a spare and the bonus throw is a strike
* **Real-life situations**. After three points, the game is almost completed. It should be tested in real-life situations to check if whole code structure was correct.
  + 1. A perfect game consists of all strikes (a total of 12 of them including the bonus throws), and has a score of 300.