Design justification (including a brief discussion of at least one alternative you considered)

The SurgicalRobot adventure game was borne out of a desire to apply computer programming to the medical sector. The game was structured to give its users a cursory feel of what a typical robot-assisted surgery looks like while teaching them basic steps in surgical operations like administering anesthesia, stitching organs, and much more.

Our final game design is a product of a series of design revisions done mostly for two major reasons:

- To make the frontend interface more interactive
- To narrow down the scope of work given the time constraint of the project

The first idea we came up with involved making a single surgical robot class that implemented the contract interface outlined in A7. It also largely involved a strictly defined process, such that the user had little to no say in the process.

However, we realized that this model won't allow a lot of interaction between the user and the program. As such, we refined the design to ensure more back-and-forth conversations by including more choices the users could choose from while ensuring to ask for the users' approval before carrying out certain decisions. In this scenario, we tried to model a real-life surgery in which the user's actions greatly affect the result. For example, if a surgeon administers the wrong anesthesia in real life, the patient will be affected as surgeries aren't reversible. This was a tricky one for us because an important part of the rubric says "Game supports reversible moves." To make clear, reversible moves in the context of our surgical game entails being able to use the lessons learned from your previous choice to make informed decisions in the next play/game level. This was done to allow the user to learn more effectively and make careful decisions as they progress through the game. Also, the user is allowed to quit the game if they make a woeful decision and would like to start again. By implementing choices like "click 1 to continue and 2 to quit," we are ensuring reversible moves in a way that would the user to learn while not costing more patients' lives due to wrong decisions.

After deciding on the interactive setting of the game, we were faced with how to ensure it followed an object-oriented approach. While having a single big surgical robot class might be easy to implement, we realized that extending it to three subclasses, each corresponding to the type of robot we are interested in, will inevitably improve the program's reusability, maintenance, and encapsulation features. We thus, had the main robot class serve as a superclass and extended its features to develop three more classes:

- Brain.java
- HeartRobot.java
- LiverRobot.java

One more alternative design we considered involved incorporating a map/location feature into the game, such that certain equipment would be placed in certain locations and would require the

user to move between different locations during the surgery. However, given the time constraint of the project, we weren't able to implement that. We, however, were able to fulfill our end goal of ensuring a user-friendly and super-educative game.