Report:

I had gotten satisfying results for all my labs and assignments that had included space shapes, so I was confident going into this Assignment 4. When I started the assignment I realised that I had not revised over some of the GUI lectures so I was a bit lost when trying to start task1. However, after I went over the notes I made in class and re-watched the lecture recordings, I felt like Task1 was relatively straight forward.

Task1:

My approach for task one was to follow the general solution for an adapter pattern, where our adapter task1 implements Tree model and stores a reference copy of our adaptee; in this case it is shape model.

Task2:

I added the observer model pattern to our application so that when shape model changed, the tree model would change as well. Shape model listener is our observer which is implemented by our task2.

Task3:

I attempted Task3 before we had gone over the lectures about Swing worker and MultiThreading. So, my first attempt at the task was incorrect (as I found out after the multithreading lecture). In my first attempt, I created a new runnable object and placed the blocking code within the run(), and then after called object.start(). This passes the tests and the space shapes application did not freeze when I tired to add an image.

However, when I realised I did it the improper way, I changed it to use swing worker. My Image rectangle now extends SwingWorker, and doInBackground() contains all blocking code. I did not need to override done(). Runtimes between the two methods I found had the same speed.