Report

Your report has significantly improved since the last practical. The single column layout with appropriate headings makes understanding what you did a lot easier. However, there are still too many spelling mistakes. Before submitting you should definitely proof read your report and make sure it reads well, spelling mistakes and awkward sentence structure breaks the flow of your report which is a shame. In the third section where you discuss the search strategy you do not go into any detail on how Prim's algorithm actually works. It is fine to use well established algorithms in your practicals to your advantage but you must show proof that you understand how they work. This is the point of your report. Use it to justify your design decisions and discuss the algorithms you used. You state that Prim's algorithm is faster than another algorithm, however is this really of importance in this practical? You are trying to find the optimal in terms of game time build strategy given a goal state and not trying to find a possible solution as quickly as possible.

Your report should contain all images of testing and not rely on external images.

In terms of testing you can test to ensure your program does not contain any bugs such as entering negative numbers or letters into the value fields. Whenever you allow the user to enter data you must ensure it is valid before dealing with it to ensure your program does not crash or behave unexpectedly (See Buffer Overflows on Wiki). When testing your program you should be as honest as possible and try to break your own code. By doing this you should demonstrate how your program works. For instance, stating that you get a time of X for a certain goal does not say a lot. You could have started with a simple goal and gradually made it more complex noting down the times your optimiser calculated. This could have allowed you to discuss the underlying functionality of your program and also demonstrated that it works correctly. Screenshots by themselves don't demonstrate any thought behind testing.

Code

Please try to structure your code using an object oriented approach. For the moment it probably doesn't really matter to you if you structure your program out elegantly or not, however when your practicals get bigger and more complex having a good structure is well worth the extra time it takes to design it. Try to make your code as flexible and as generic as possible. If you wanted to change some aspect of your code could it be done without much effort? Also think about fellow programmers who you might give the code to, could they easily follow your trail of though and extend it easily? A lot of your build methods are similar and could have been compressed into a generic method. You could also have split units and buildings into classes and so made your code more clear. You make use of the java.util.Timer class which makes it easy for actions to be triggered after a certain amount of time. It would have been interesting if you had made your own class that mapped events to times, allowing you to abstract over the idea of time. Thus you could have had the option to run the code in real time or not. In order to test you code I had to manually remove the timing values. For the next practical try to design your program loop better, the user should be able to run the program once but try it out with different configurations instead of having to quit the program to be able to see the main GUI screen again. When designing a GUI you need to make sure it is usable on different resolutions. On my resolution half of the GUI was missing and I had to change it by hand. Also, if I select a unit without any buildings no error message is shown, instead your program shows the output console but never prints to it.

Excellent submission but more attention needs to be paid towards the report and your program structure.