I first wrote tests for GraphEdge first, then GraphNode, then Graph. For all classes, I first tested their constructors and observer functions if they have any. Because I wanted to optimize lexicographically printing node names and outgoing edges in my design, I wrote a lot of tests, including helper assertion functions, that make sure their override comparison functions work correctly, which are important for keeping TreeSet representation structures sorted. These functions include compareTo, equals, and hashcode. For GraphNode, I wrote tests for adding edges. For Graph, I wrote tests for adding both nodes and edges. Finally, I wrote tests for GraphWrapper's listNodes and listChildren functions, where I make sure the strings are returned in lexicographical order by the iterator.

In terms of black box testing heuristics, I mainly used equivalence partitioning to compare GraphEdge and GraphNode objects. Two GraphEdges and GraphNodes are either equal or one comes before the other in their lexicographic ordering.

After finishing my implementation, I felt that my original tests are sufficient, since I didn't leave any function untested.