Testing Document

Connor Ahearn, Micah Arndt and Kevin Chan

Table of Contents

1. Outline

At the time this document is being published, the focus of the project has been making a GUI that can create Scenarios that can be interpreted by the BrailleBox. Our group focus before the midterm is excluding all provided code for testing scenarios until we have a working product to produce them.

All text file creation is currently based on the assumption that the device used has only one Braille Cell and 4 Buttons. In the future different devices will be accounted for, but for now this is our scope.

2. Document Format

The format of the document is structured as follows:

- The class / function being tested
- Summary of the class / function's purpose or task
- Test cases implemented for said class / function
    - Also listing why these cases were created
- Why the cases listed above are sufficient
- EcLemma Summary

3. Individual Classes

a. Printer.java

The printer class is the portion of the application tasked with printing the information from the GUI onto a text file. It does this by receiving information in the form of *Block* objects (**Block.java** listed above). It then stores the information in a collection until the **print()** method is called, which is when the information is printed to the text file. The text file it produces is based on the examples given for the simulator to run.

The class itself doesn't serve a complex purpose. It has one specific task, and as such only has a public constructor and 2 public methods.

Printer() (Original Constructor)

- Creates a new Printer object
    - Takes a filename for the new file, as well as how many cells and buttons the new scenario will use
    - Places the information required for the first 3 lines of the text file into the **lines** collection

The constructor was tested with the **testInitial1()** test in **testPrinter.java**. It printed the input to the text file in the correct format. The same test was used throughout the rest of the tests to assure the initial block is correct before any other information is printed.

addBlock() and *print()* (Method)

- Adds a block object to the collection of lines to be printed
- Formats the information in a way the BrailleBox can read it
- Prints the information in the collection to the text file

These methods were tested in the **test1Block()** and **test2Block()** methods. The main portion being checked for the addBlock() method is that the information is readable for the BrailleBox, and no information is lost. These tests checked the correctness of the text file printed for 1 and 2 blocks of input.

More tests will be required for these two methods in the future when more freedom of input from the user are accepted.

These tests provide 94.3% test coverage to the Printer Class.

Block.java

Datatype used for collecting user input together for each section of the scenario.

Block() (Original Constructor)

- Stores all information provided to Constructor parameters in fields

This constructor is tested with the **testConstructor()** method, which runs the constructor with some test input and checks the fields of the new Block.

The Block class has 70% test coverage.

BrailleInterpreter.java

Object that converts characters to braille pin binary equivalent. Braille pin equivalents are stored and retrieved from a HashMap, and the pin equivalents were based on the BrailleBox simulator code.

BrailleInterpreter() (Constructor)

- Adds all Braille Possibilities to the HashMap, initializing its field

This class is tested by running the constructor in a JUnit test **testConstructor()**.

This class has 100% coverage.

ScenarioCreator.java

GUI class that allows the user to create Scenarios. At the time of publishing, this class was not capable of actually using its input, so **testInit()** simply ran the constructor and used its **.launch()** method.

This class has 97.9% coverage.