SuperSweeper Incremental & Regression Testing

9/20/13
Team #8:
Offir Golan
Adam Rea
Austin Drefke
Chris Wendt

Incremental Testing

- **Defined Components:** See "/documents/SuperSweeper Design.pdf" file to see component descriptions and their relationships.

- **Type of Incremental Testing Used:** We used bottom up incremental testing because it's much easier to test to make sure that the simpler, base components work before testing the components that need other modules to be created. Therefore we tested the GameFrame and it's sub panels first, followed by the game logic and mouse listeners that layed on top of it. The higher modules, such as GameState, GridUnit, and Powerup, provided actual game logic and game flow, which needed the base modules to be displayed and interacted with correctly.

Product	SuperSweeper
Date	9/20/13
Authors	Team 8

Defect #	Description	Severity	How To Correct
1	Game logic was being kept in GridUnit, when it should have been within the scope of GameState.	3	Move functions into GameState so that GameLogic is in one universal spot.
2	A Utility method for generating bombs via flattening array was not compatible with grid definitions that were not perfect squares.	1	The flatten method was changed to accept varying height and width.
3	GridPanel: After changing the image of a grid unit, the panel would call paint twice. Once from the super() call, and the other from the method being called. This causes the screen to flash every time the user pressed a unit.	3	Refactor the existing code to call the repaint method instead of the paint method. This

			removes the override of the predefined paint method and allows us to only call paint once per click
4	LevelSelectionPanel fed the bonus level property file to GamePanel incorrectly because it wasn't referenced as an integer. This issue is due to the application using the level number to locate the appropriate properties file.	1	To fix this, we added some code to see if we receive "Bonus Level" as input and load the correct properties files.
5	GameState couldn't see the ScoreLabel in GameFrame due to scope.	2	We passed a reference the the label needed by GameState down from GameFrame in its constructor
6	GridPanel couldn't access game logic from GameState due to scope.	2	We passed a reference of GameState, as well as it's parent panel GamePanel so that it could make all appropriate calls
7	LevelSelectPanel could not inform GameFrame of what GamePanels to spawn from the user's selection due to scope.	2	We passed a reference of the parent GameFrame to LevelSelectPanel so that it could communicate back the user's input appropriately

Regression Testing

Product	SuperSweeper
Date	9/20/13
Author	Team 8

Defect #	**Description**	**Severit y**	**How To Correct**

1	When implementing PowerUp logic, it exposed mines which was a count used by endGame() logic, causing premature game endings	1	Instead of exposing mines, it just flagged the mines for the user instead.
2	When multiple sub panels were added to GamePanel, the drawing method would cause the grid to be drawn on top of the labels and buttons	3	We repositioned the sub-panels so that they would account for each other’s heights
3	The rearranged sub panels caused mouse events y coordinate to be off by a factor of how big the label panel was.	2	We accommodated for the difference inside of the event by subtracting the label panels height.
4	When larger grids were added to the GameFrame, the screen would not always be large enough to draw the entire thing.	3	We just added a pack() after rendering a new grid so that the GameFrame would be at least large enough to show the entire grid
5	When user properties were added to the project, the classpath was disrupted in such a way that the application couldn’t find any of the properties files and wouldn’t load	1	We changed the directory structure/classpath of the project to accommodate for the additional properties file