Documentation

for Java fundamentals group project

Game application: "Mouse Мaze"

**

Developed by:

Todor Ilchev

Stanislav Nedyalkov

Ivelin Stamov

Htistian Kanev

Team "Athame"

Evaluated by:

Sofia – 2015

**1. Description:**

Our application is a game in which the main character is the mouse cursor. The players objective is to go through the labyrinth without touching the different obstacles (red and black objects). So far there are two levels. If the player is fast enough with solving both of them, he could save his high score in the top 10 scoreboard.

For developing this game we chose to use JavaFX with Scene Builder (for the design of the windows and levels) and Net Beans IDE (for the logic).

The game is mainly based on an "onMouseExited" event which is set on a rectangle. That rectangle is sent on the bottom layer of an anchor pane. This was a solution for the base game functionality but it caused problems when the player stepped on an item which shouldn’t cause a loss (button or label). We followed the Javadoc of labels and buttons and we found that they share a common abstract parent class "Labeled". So we made a boolean method, named "contains". With it we can mace exclusions from the "onMouseExited" event by checking coordinates plus a tolerance of 2.0d.

We found several ways the player could cheat the game. One is to traverse through the button with the Tab key and click the finish button without moving the mouse at all. This was an easy fix - we removed the "focus traversable" property of the buttons. Another cheat we found was to go hold the stop button when the game is started and drag it to the finish. This way the "onMouseExited" event isn't triggered and the player can't lose. This we fixed by setting an "onDragDetected" event on the start buttons and calling the stopGame method whenever the event is triggered.

The Scoreboard implements the Serializable interface and has an event listener for new submits. A score is only added if its time is smaller than the 10th entry or there are less than 10 entrees. We use an iterative binary search algorithm to find at which position each entry should be placed.

**2. Contributions**

Stanislav Nedyalkov - LevelOne and other.

Todor Ilchev - LevelTwo, Scoreboard and other.

See <https://github.com/clueless92/JavaGroupProject/commits/master> for more info.