Report for the bonus assignment of the semester project “Javass” (May 2019)

By Erik Alessandro Wengle (297099) and Hannah Laureen Casey (300981)

For the bonus of our semester project, we added a graphical interface that allows to configure and launch a game of *Javass*. It can be launched by running the class *Main* in the package *ch.epfl.javass*. It will launch a graphical interface where the user can decide if to host a game or to join a game, where explanations are given upon hovering over the buttons ‘host’ and ‘join’.

If the user, by clicking on the button ‘host’, decides to host a game, a new pane showing the games’ configurations will become visible. Here the user can change the attributes for the different players. Every player has to have a name and a player type. For a remote player, the IP address must be entered too, where the default IP address is the localhost *(127.0.0.1)*. For a simulated player the number of iterations must be entered, where the default number is 10’000. Optionally, a seed for the random generator can be entered in the field ‘Seed’.

The game can be started by clicking the button ‘start’, as soon as all arguments have been entered.

If any of the mandatory arguments is missing or any entry is invalid, a red error box will let the user know where exactly the mistake has been made, and the game will not start. When clicking on the return button, the pane showing these options becomes invisible again and the user can again decide between hosting or joining a game.

As soon as the game starts, the Javass Launcher window is closed, while the game opens in another window running on another thread.

If the user decides to join a game, they will reach a different pane by clicking on the button ‘join’. Clicking this button will also create an instance of the class *RemotePlayerServer* and invoke its method *run()*. The new scene will show the users IP address and will not change until the server has been connected to. If a game is successfully joined, the window generated by the *GraphicalPlayerAdapter* is shown and the Javass launcher window will close. This is achieved by giving said adapter a *BooleanProperty* whose value is based on whether the *GraphicalPlayer* being adapted has been initialized or not.

The Launcher itself has been created using the JavaFX library as well as a separate CSS sheet containing the styling for the launcher. This was done as to reduce the amount of code written in the class *GraphicalLauncher* to ensure a better readability. Overall, this class works similar to the class *GraphicalPlayer*; it is started by a separate JavaFX application, in our case the class *Main*, where the stage will become visible upon running the main method of the latter.

The overall improvement achieved hereby is first and foremost a quality-of-life improvement, making the program more user-friendly.