```
/*

 Nand2Tetris launcher. Provides a simple interface to quickly compile the programs for
the Hack Computer.

*/

import gui.Gui;
import assembler.*;
import vmtranslator.*;

import javafx.application.*;
import javafx.stage.*;

import java.io.*;

public class nand2tetrisLauncher extends Application {

  private Gui gui;

  private void setActions() { // Maps all supported events to subsequent handling
reaction.
    gui.getLoad1Button().setOnAction( e -> load1ButtonHandle() );
    gui.getLoad2Button().setOnAction( e -> load2ButtonHandle() );
    gui.getLoad3Button().setOnAction( e -> load3ButtonHandle() );

    gui.getAssembleButton().setOnAction( e -> assembleButtonHandle() );
    gui.getTranslateButton().setOnAction( e -> translateButtonHandle() );
    gui.getCompileButton().setOnAction(e -> compileButtonHandle());

    gui.getTextField1().textProperty().addListener((obs, oldText, newText) ->
resetOuputFields1()); // All reset the TextFields UNDER them.
    gui.getTextField2().textProperty().addListener((obs, oldText, newText) ->
resetOuputFields2());
    gui.getTextField3().textProperty().addListener((obs, oldText, newText) ->
resetOuputFields3());
  }

  private void load1ButtonHandle() {
    gui.getTextField1().setText(getJackFile());
  }

  private void load2ButtonHandle() {
    gui.getTextField2().setText(getVMDirectory());
  }

  private void load3ButtonHandle() {
    gui.getTextField3().setText(getAsmFile());
  }

  private String getJackFile() {
    FileChooser fileChooser = new
FileChooser();                                                                    //
Creates FileChooser.

fileChooser.setInitialDirectory(retrieveSavedDirectory());
Sets last visited directory as starting point.
    FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("Assembly
files (*.jack)", "*.jack"); // Only show .asm files.
    fileChooser.getExtensionFilters().add(extFilter);
    File file = fileChooser.showOpenDialog(new
Stage());                                                          // Retrieves
chosen file.
    return file.getAbsolutePath();
  }
```

```
  private String getVMDirectory() {
    DirectoryChooser directoryChooser = new DirectoryChooser();     // Creates a
DirectoryChooser.
    directoryChooser.setInitialDirectory(retrieveSavedDirectory()); // Sets it to be
opened on the last opened directory.
    File directory = directoryChooser.showDialog(new Stage());      // Retrieves the
chosen directory.
    if (!containsVMFile(directory)) {                               // If there are
no .vm files gives out an error.
      gui.getErrorLabel().setText("ERROR: Chosen directory does not contain any .vm
file.");
      return "";
    }
    return directory.getAbsolutePath();
  }

  private File retrieveSavedDirectory() {
    String path = "";
    try {
      String basePath = new File("").getAbsolutePath();
      BufferedReader r = new BufferedReader(new FileReader(new File(basePath +
File.separator + "lastPath.txt"))); // Creates BufferedReader to the file that memorizes
the last opened folder (lastPath.txt)
      path = r.readLine();
      r.close();
    } catch (Exception e)  {
      e.printStackTrace();
    }
    return new File(path);
  }

  private boolean containsVMFile(File directory) {
    String [] files = directory.list();
    if (files != null) {
      for(String file : files) {
        if (file.contains(".vm")) return true;
      }
    }
    return false;
  }

  private String getAsmFile() {
    FileChooser fileChooser = new
FileChooser();                                                                  //
Creates FileChooser.

fileChooser.setInitialDirectory(retrieveSavedDirectory());
Sets last visited directory as starting point.
    FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("Assembly
files (*.asm)", "*.asm"); // Only show .asm files.
    fileChooser.getExtensionFilters().add(extFilter);
    File file = fileChooser.showOpenDialog(new
Stage());                                                                       // Retrieves
chosen file.
    return file.getAbsolutePath();
  }

  private void assembleButtonHandle() {
    String assemblyFile =
gui.getTextField3().getText();                                      // Retrieves .asm file
to assemble.
    HackAssembler assembler = new
HackAssembler(assemblyFile);                                    // ASSEMBLER
    gui.getTextField4().setText(assemblyFile.replaceAll(".asm", ".hack")); // Sets output
textbox with output file path.
```

```
saveLastDirectory();                                                                    //
Saves output folder for next FileChooser instance.
  }

  private void translateButtonHandle() {
    File VMDirectory = new
File(gui.getTextField2().getText());                                    // Retrieves
folder with .vm files.
    CodeWriter translator = new
CodeWriter(VMDirectory);                                                // TRANSLATOR
    String filePath = VMDirectory.getAbsolutePath() + File.separator +
VMDirectory.getName() + ".asm"; // Sets assembler textbox with .asm file path.
    gui.getTextField3().setText(filePath);
    assembleButtonHandle();
  }

  private void saveLastDirectory() {// Saves the folder to which the last output .hack
file has been saved.
    try {
      String basePath = new File("").getAbsolutePath();
      PrintWriter w = new PrintWriter(new BufferedWriter(new FileWriter(new File(basePath
+ File.separator + "lastPath.txt"))));
      w.println(new File(gui.getTextField4().getText()).getParent());
      w.close();
    } catch (Exception e)  {
      e.printStackTrace();
    }
  }



  private void compileButtonHandle() {
  }

  private void resetOuputFields1() {
    gui.getTextField2().setText("");
    resetOuputFields2();
  }

  private void resetOuputFields2() {
    gui.getTextField3().setText("");
    resetOuputFields3();
  }

  private void resetOuputFields3() {
    gui.getTextField4().setText("");
    gui.getErrorLabel().setText("");
  }

  @Override
  public void init() { System.out.println("Nand2Tetris launcher initializing..."); }

  @Override
  public void start(Stage window) {
    gui = new Gui(); // Builds and launches GUI.
    setActions();    // Handles the events.
  }

  @Override
  public void stop() { System.out.println("Exiting."); }

  public static void main(String [] args) {
    launch();
  }

}
```