File: MainMenu.java

Components

* btnNewEmployeeActionPerformed,
* btnViewEmployeeActionPerformed
* btnViewTimetableActionPerformed
* btnSaveActionPerformed
* btnExitActionPerformed

Expected results

This is a simple test as each of the components are primarily used to produce a readable and functional GUI. When each component is ran, another class is executed and should show the correct user interface. Each component, when tested with null pointers parameters should not produce any error but no interfaces should execute. Components cannot be moved around, code should not be modified and is generated by the Form Editor.

Results

On running, the GUI is functional and all components work as expected. Using null pointers does not make any faults in the code. All components are compatible and interact correctly. Varying the components in the order they were activated in shared memory does not produce an error.  
  
File: AddEmployee.java

Components

* btnAddActionPerformed
* btnDiscardActionPerformed

Expected results

On executing this code, the GUI should appear asking for input of the new employee to be added. All should have a text box for users to input information, then add should update the program and add the new input into the application. Pressing ‘Add’ should successfully upload the information and exit the current window, bringing the user back to the main menu. The same should happen with ‘Discard’ but not add any information.

Results

When shown, the interface shows the correct format with functional ‘Add’ and Discard’ buttons. When no text is inserted and ‘Add’ is pressed, the program shows faults in the code, but doesn’t produce any error message in the application. When adding information and adding it to the application, the correct data transfers the correct data across interfaces which is evident in further testing.   
  
File:ViewEmployee.java

Components

* btnPrevActionPerformed
* btnNextActionPerformed
* jButtonActionPerformed
* showEmployee

Expected Results

Both ‘btn-‘ components should be functional, with the user being able to view the previous and next employee. ‘jButtonActionPerforned’ should close the application. Finally showEmployee will give a list of employees added to the program.

Results

Both ‘Next’ and ‘Previous’ buttons behave as specified. The previous interface communicated the correct data at the right time and showed the correct data. The code indicated a ‘show employee’ but there is nothing to show on the GUI for this, therefore **should be modified** to include an extra button.

File: ViewTimetable.java

Components

* btnCloseActionPerformed
* btnViewActionPerformed
* jButton3ActionPerformed
* public static void main

Expected Results

‘btnCloseActionPerformed’ will close the current window to return to the main menu and ‘btnViewActionPerformed’ will generate the timetable created from the current employees. ‘main’ will execute the form.

Results

Both btnCloseActionPerformed and btnViewActionPerformed function correctly. No null pointers to test code with.  
  
File: Person.java

Components

* public class Person
* public Person
* public String getFullName (getMaxHours, getIdNumber)
* public Boolean
* public int getCurrentHours
* public void setFullName (setShortName, setMaxHours, setIdNumber, resetHours, incrementHours)

Expected Results

This class is responsible for the creation of a new person. This should enable the user to create an employee and add it to the applications database. Public String / int / boolean methods used to return the values and set methods to change the value. Without this, the employees cannot be added.

Results

Each component has a purpose and outputs the correct data requested. Null pointer parameters do produce an error in the output but does not show on GUI. Varying the components in the order they were activated in shared memory does not produce an error.

File: Timetable.java

Components

* public static String[][] makeTimetable
* private static String printLine
* public static void printTable
* private static String printBoolLine
* public static void printBoolTable
* public static void printIntTable
* private static String catLine
* public static String catTable

Expected Results

This class is used to create a simple table and generate the data to make a suitable work rota. ‘static string‘ and ‘void print’ methods are used to print the data and generate a table layout.

Results  
  
The code can be seen working when viewing the timetable and using the buttons onscreen to interact. No errors can be produced.  
  
File: INSEBase.java

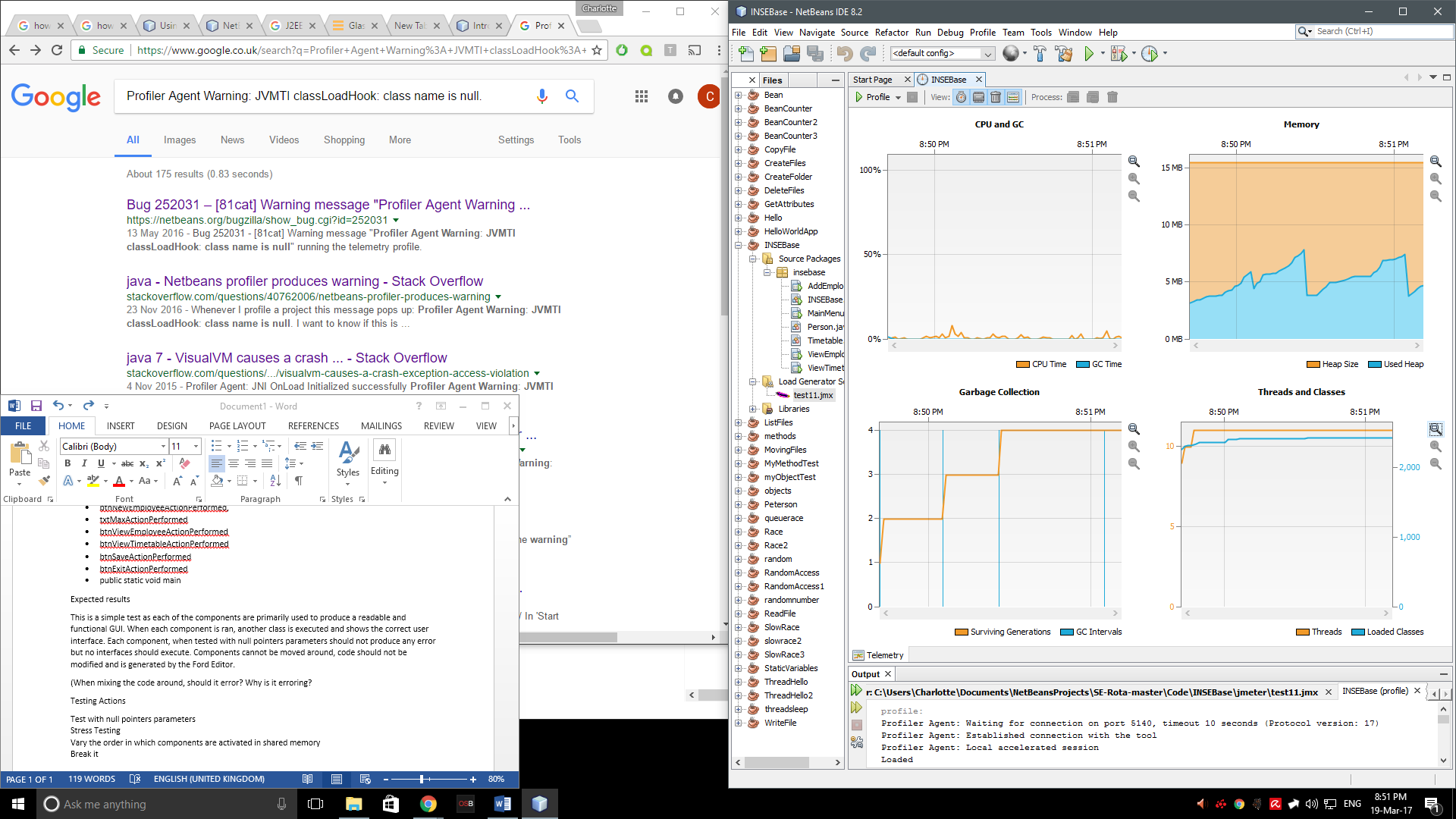
Expected result

The whole application could not be pieced together without this file. Evident of working code should be seen on first use of the application. Each component should be working together to create a working, functional application.

Conclusion

Data passes from one component to another using methods correctly with shared memory between components accurate and does not produce any false data. It is evident that components can be successfully called by other components. Each call to an external component work well, null parameters work with most of the system, but should be revised. Application is quite stable but can be improved.

Stress Test / Profile Testing



Testing the program we can see that at the beginning of running the application there is a spike within the CPU and used heap memory. The same happens with Threads and Classes but does stay constant as we can see also with the Garbage Collection table.