Report (Lab 2.2 to 2.7)

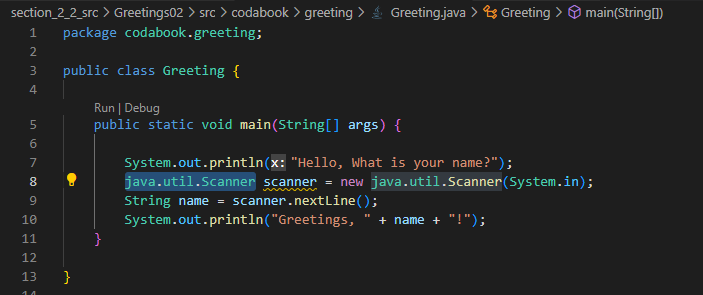
Lab 2.2

Text

Description automatically generated

This is the result I get after running execute the three programs by running from my Operating System console window provided Java Runtime Environment is installed and configured. In the OS console window command prompt, I first navigate to the folder location and type the relative commands to execute each of the above-listed programs accordingly.

In Greetings01 and Greetings02, it works the same. The difference is Greetings01 imported the java.util.Scanner. So that in the program no need to write java.util before using the Scanner. Since the Scanner is an external class from the java.util package that both of the programs need to refer to. And it is not imported implicitly by the Java compiler. Thus, we either need to explicitly define an import for the java.util.Scanner statement in our source code or using its fully qualified name the java.util.Scanner as Greetings02 source code shown below.



Lab 2.3

Text

Description automatically generated

Above show the result of executing AgeCalculatorApp01 in the command prompt. It will work fine because the AgeCalculator is imported into the AgeCalculatorApp.

Text

Description automatically generated

So, there will be a dependency relationship between the component and the application. Since we need to include AgeCalculator01.jar in our build path. And this violates the independent component fabrication.

Lab 2.4 to 2.6

Text

Description automatically generated

The above shows the result of executing AgeCalculatorApp with the glue code modification.

Text

Description automatically generated

The glue code instantiates the providing component object by using the AgeCalculator Interface and passes the object to the requiring component which is AgeCalculatorApp as the source code shown above. As a result, we do not need to construct the AgeCalculator class that will cause build time dependency.

The glue code act as a middleman component providing the respective interface to the application that required the interface by referring to the Application, Calculator Interface, and the Calculator program. However, we must specify the components that need to be glued which means there is custom glue needs to be written for each group of components that have to be glued which is a very tedious job to do.

Lab 2.7

Text

Description automatically generated

The above shows the result of executing the codabook.agecalculator.app.AgeCalculatorApp.

Text

Description automatically generated

From the figure above we can see the program uses ComponentRegistry to fetch the component interface to the component that requires the respective interface. And without instantiating the AgeCalculatorIfce in the 22 lines of code.

Text

Description automatically generated

So, the ComponentRegistry will return an object that is the interface class of the component. In this example, it will be AgeCalculatorIfce.

Text

Description automatically generated And the AgeCalculator object and interface are registered with the ComponenetRunner with the ComponentRegistry registerComponent method.