**Template for: Programming Design Principles Skills Demo Two**

**Learners are required to work as part of a team (between 3 and 5 learners inclusive) for this skills demonstration.**

Please identify the members of your team for this skills demonstration:

|  |  |
| --- | --- |
| **Name of student:** | **Key role on Team:** |
| Robert | Product Owner |
| Steven | Scrum Master |
| Gavin | Development Team member 1 |
| Laura | Development Team member 2 |

**SECTION ONE: ALGORITHM**

1. Develop a **top down algorithm** using pseudo code **or** flowchart(s) generated using a suitable tool such as draw.io to solve the problem posed by this skills demonstration.

Each member of the team should take responsibility for one section of the top down algorithm. Each section should provide a suitably detailed algorithm for each task. **Identify team member responsible for drafting each section.**

**START-** run the program

**OUTPUT**- java program window opens

**INPUT**- First Name, Surname

**INPUT-** Gender

**If** Gender =“Male”

quote= Double general Base

**else if** Gender =”Female”

quote= 70% of General Base

**INPUT-** Age

**If** Age= “under 35”

quote = quote/100\*20 + quote

**else if** Age= “ 35 - 55”

quote = quote/100\*40 + quote

**else if** Age= “56-70”

quote = quote/100\*65 + quote

**else** Age= “Over 70”

quote = 0- No quote for someone over the age of 70

**Completed by: Steven Joyce**

**INPUT** - Health Conditions

**If -** Health Conditions = “Bone Marrow and Immune System”

quote = quote/100\*20 + quote

**else if -**Health Conditions= “Cancer”

quote = quote/100\*25 + quote

**else if** -Health Conditions= “Cardiovascular Disease”

quote = quote/100\*30 + quote

**else if** -Health Conditions= “Gastrointestinal”

quote = quote/100\*10 + quote

**else if** -Health Conditions= “Infections”

quote = quote/100\*10 + quote

**else if -** Health Conditions= “Kidneys”

quote = quote/100\*25 + quote

**else if** - Health Conditions= “Lungs”

quote = quote/100\*25 + quote

**else if** - Health Conditions= “Musculoskeletal”

quote = quote/100\*30 + quote

**else** - Health Conditions= “None”

quote = quote

**INPUT** - Other

**If** - Other= “Yes”

quote = quote/2+quote

**else -** Other=“None”

quote = quote

**INPUT** - Smoker

**If** -Smoker = “yes”

quote = quote + 75

**else** -Smoker = “no”

quote = quote - 100

**Completed by: Laura O’Shea**

**INPUT -** Hospital Cover

**If** - Hospital Cover = “Public Cover”

quote = quote

**else if** - Hospital Cover = “Private Cover”

quote = quote/100\*20 + quote

**else** - Hospital Cover = “Comprehensive”

quote = quote/100\*30 + quote

**INPUT -** Day to Day Expenses

**If** - Day to Day Expenses= “Standard”

quote = quote

**else if** - Day to Day Expenses= “Enhanced”

quote = quote/100\*20 + quote

**else** - Day to Day Expenses= “Comprehensive”

quote = quote/100\*30 + quote

**INPUT -** Policy Type

**If** - Policy Type= “Individual”

quote = quote

**else**  - Policy Type= “Corporate”

quote = quote - quote/100\*10

**Completed by: Gavin Maher**

**INPUT-** Select Calculate Button

**OUTPUT** - ID no and Total

**INPUT-** Select Buy Policy Button

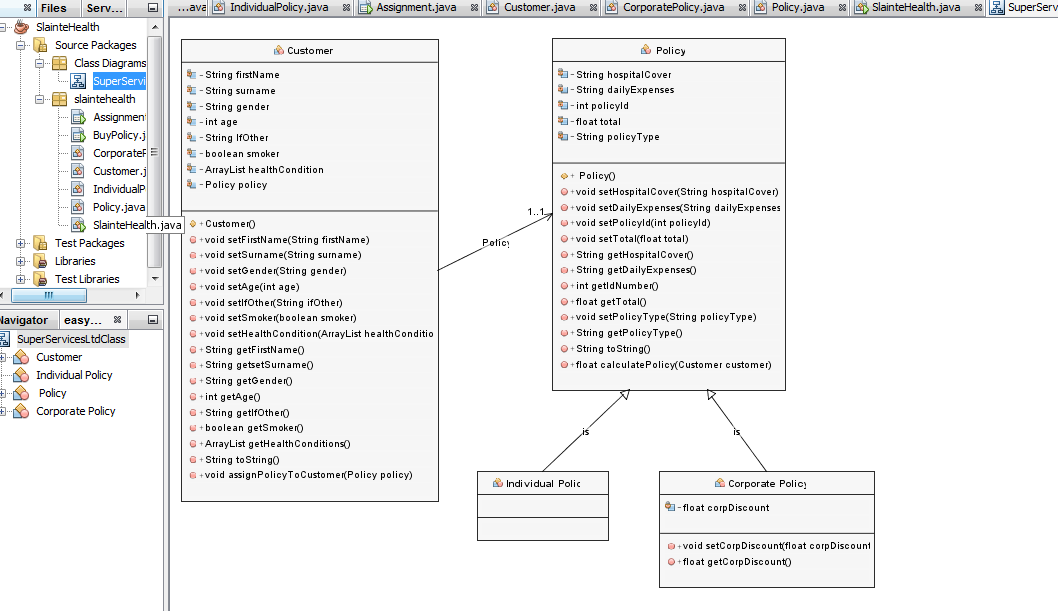
**OUTPUT** - All Customer details in buy Policy window

**INPUT**- Select the clear Form button to reset the form and increment the id number.

**Completed by: Robert Mroskowiak**

**END**

1. Using the **top down algorithm** from above provide a complete data dictionary of the identifiers you expect to use in your final program.
2. Insert **the UML Class Diagram** you created as part of your Software Architecture Skills Demonstrations.



1. Identify a selection of the key variables and methods you also created for the Main GUI and Dialog forms in the table below:

|  |  |  |
| --- | --- | --- |
| **Class Name** | Assignment | BuyPolicy |
| **Variables** | String firstName = "";  String surName = "";  String ifOther = "";  String gender = "";  String ageChoice ="";  String policyCover ="";  String hospitalCover="";  String dayExpenses="";  int idNumber= 0;  String total="";  boolean smoker = false;  //ArrayList  ArrayList healthConditions = new ArrayList();  int policyId = 1;  Policy p1;  Customer c1;    boolean policyForm = false; | String health = "";  ArrayList health2 = new ArrayList();  int i = 0; |
| **Methods** | validateForm()  extractFormDetails(); | BuyPolicy  run() |

**SECTION TWO: ACCURATE PROGRAMMING**

Use the developed algorithm to implement a compiled and released program that uses:

1. **Modularisation** (with appropriate use of local and global variables) – each member of the team should again take responsibility for writing code for at least one Class in the program.

|  |  |
| --- | --- |
| **Class Name** | **Team member primarily responsible for design and code** |
| **Policy** | **Gavin** |
| **Individual Policy** | **Steven** |
| **Corporate Policy** | **Steven** |
| **Customer** | **Laura** |
| **Assignment** | **Robert** |
| **BuyPolicy** | **Steven** |
| **SlainteHealth** | **Laura** |

1. **User defined functions** designed to encapsulate single specific tasks e.g. the team may wish to create user defined functions for some of the following e.g.: extracting user input from the GUI form, validating the form, checking certain insurance criteria are met, performing calculations, populating the summary dialog. Again each member of the team should again take responsibility for at least one user defined function in the program.

|  |  |
| --- | --- |
| **Name of user defined function** | **Team member primarily responsible for code** |
| Designing 2 GUI forms | Gavin |
| Extracting user input from the Class Diagram into the GUI | Steven |
| calculate the policy | Laura |
| validate Form | Robert |
| Extracting user input from the GUI form into the summary Dialog | Steven |

1. At least one s**ystem defined function** – e.g. the output of the final price is formatted to Euro and rounded to nearest whole value.

|  |
| --- |
| **Name of system defined function(s) used** |
| numberFormat.getCurrencyInstance() |
| Math.round() |

**SECTION THREE: APPROPRIATE TESTING**

Develop **test data** that thoroughly tests the given problem **and apply that test data to the coded solution.** Each member of the team should take responsibility for one test applied to the code.

1. Using the following test data table indicate the output you expect to see before the program is compiled and record the actual output from running your final program in the next table below:

**Test Data Table:**

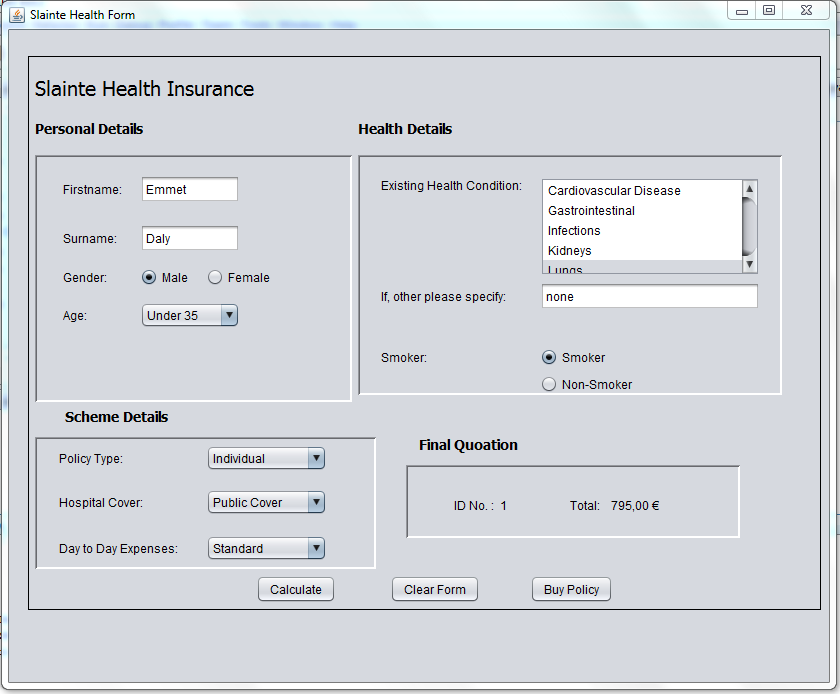
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** | **First Name** | **Surname** | **Gender** | **Age Bracket** | **Policy Type** | **Hospital Cover** | **Day to Day Expenses** | **Health Condition** | **Other** | **Smoke** |
| A | Emmet | Daly | Male | Under 35 | Individual Policy | Public Hospital | Standard | Bone Marrow  Lungs |  | Yes |
| B | Patricia | Folan | Female | 35 to 55 inclusive | Individual Policy | Comprehensive Hospital Cover | Enhanced | None | Yes | No |
| C | Donal | Russell | Male | Under 35 | Individual Policy | Selected Private Hospitals | Standard | Cancer |  | No |
| D | Chuck | Norris | Male | Under 35 | Individual Policy | Selected Private Hospitals | Enhanced | Infections  Kidneys | Yes | No |
| E | Daniel | Kiely | Male | 35 to 55 inclusive | Corporate Policy | Public Hospitals | Standard | None |  | Yes |

**Results Table:**

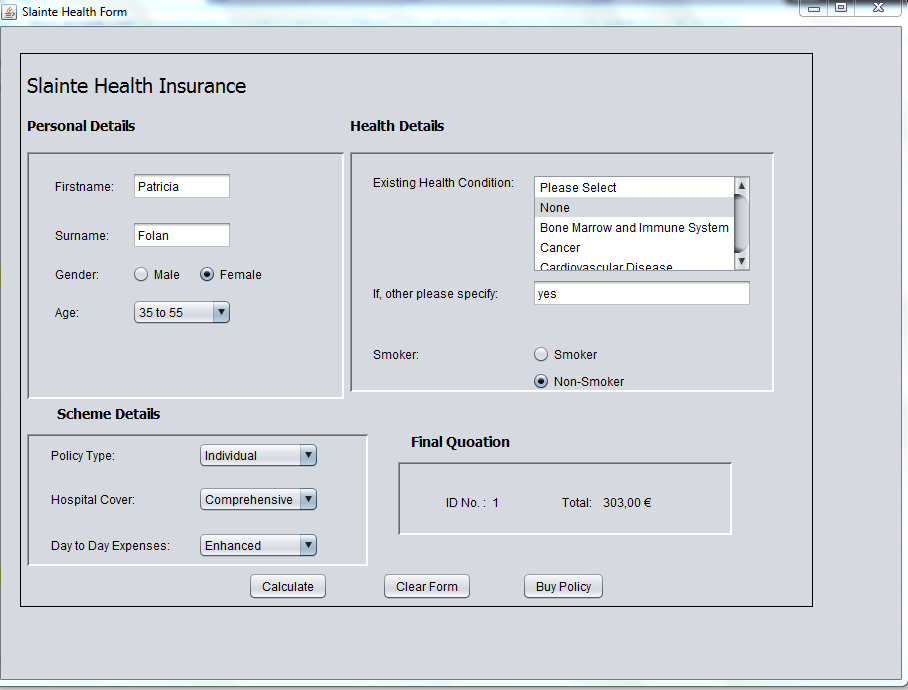
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario** | **Provide calculations and expected output prior to running your program here:** | **Record actual output after running your final program here:** | **Member of team responsible for test calculation and screenshot** |
| **A** | **€795** | **€795** | **Steven** |
| **B** | **€303** | **€303** | **Gavin** |
| **C** | **€600** | **€600** | **Robert** |
| **D** | **I am sorry to inform you that you are over our age limit for a policy...** | **I am sorry to inform you that you are over our age limit for a policy...** | **Laura** |
| **E** | **€553** | **€553** | **Steven** |

1. Provide screenshots of your final programming running using the input data provided in the table above that show the results of the compiled test data used on your coded solution.

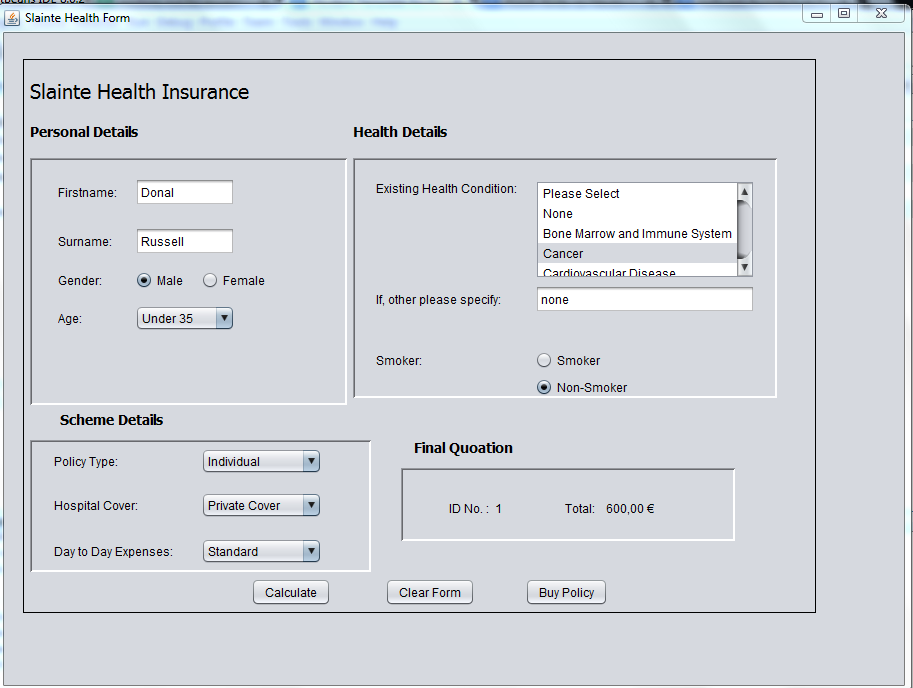
Test Scenario A:



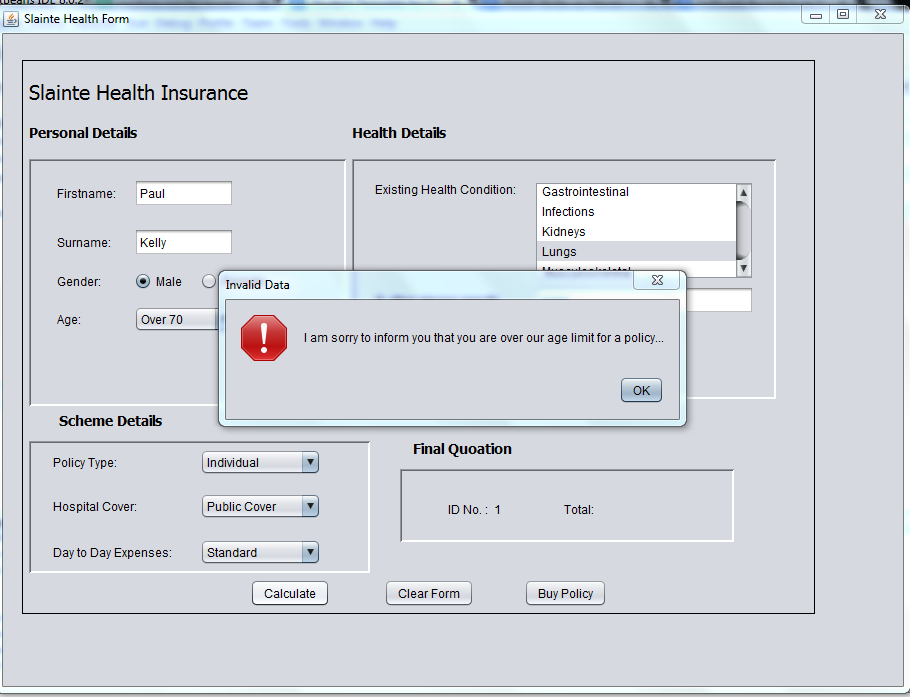
Test Scenario B:



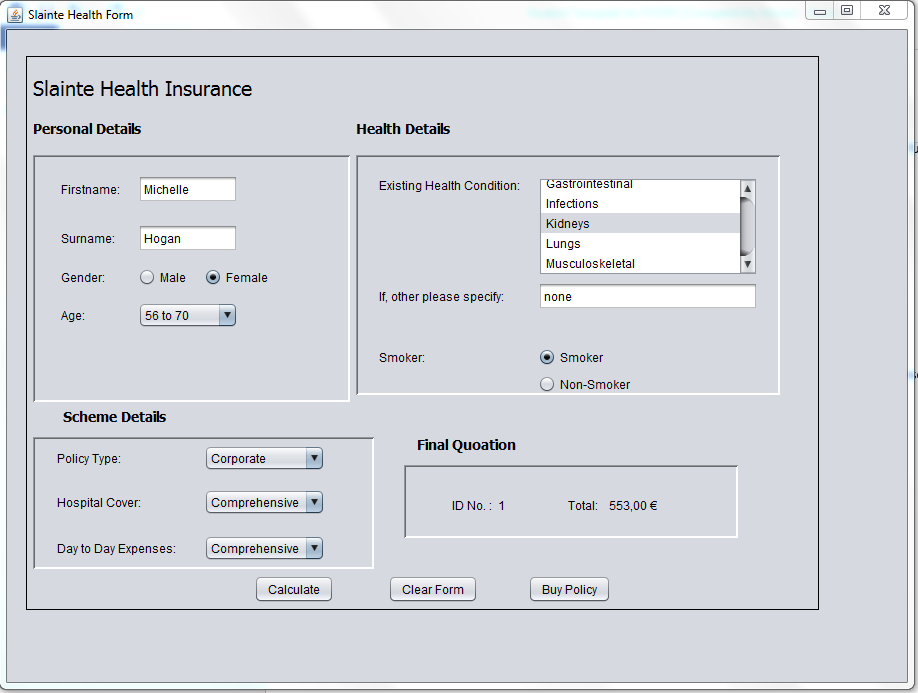
Test Scenario C:



Test Scenario D:



Test Scenario E:



**SECTION FOUR: ACCEPTED INDUSTRY STANDARD FOR CODING**

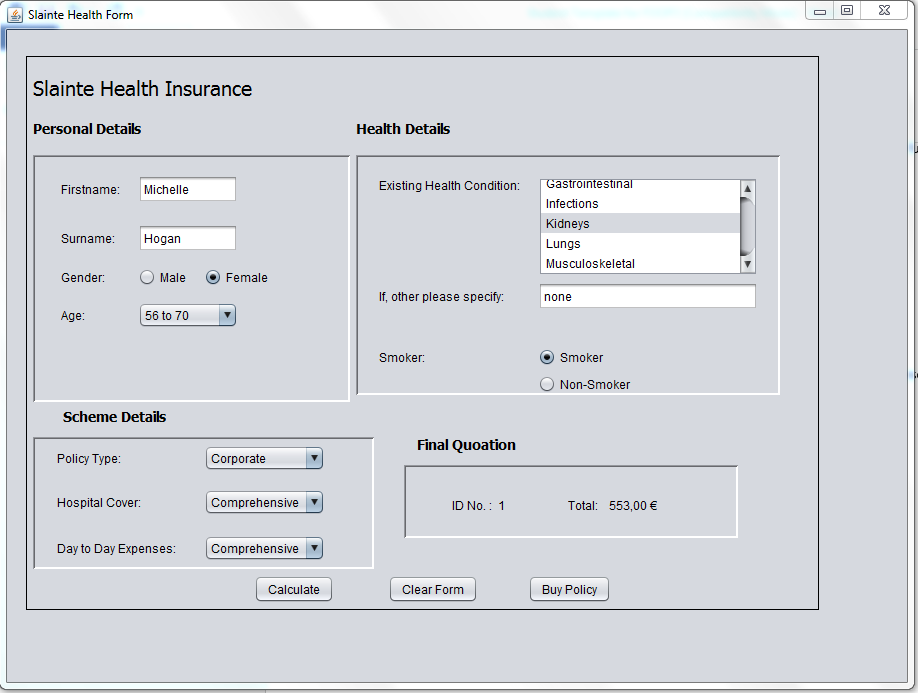
1. Please provide a well laid out printout of the final program code **with line numbers** showing the following:

* Logical sequence to program following top down designed algorithm provided.
* Code suitably commented.
* Indenting conforms to industry standard.

1. Please provide an appropriately cropped and resized screenshot(s) of your **final** program working and insert in the section below:

* Clear and consistent input prompts given to user.
* Clear and consistent output from the program, suitably displayed.

*Input and Output Screenshot displaying Print Formatting*



**SECTION FIVE: EVIDENCE OF GOOD TEAM PARTICIPATION**

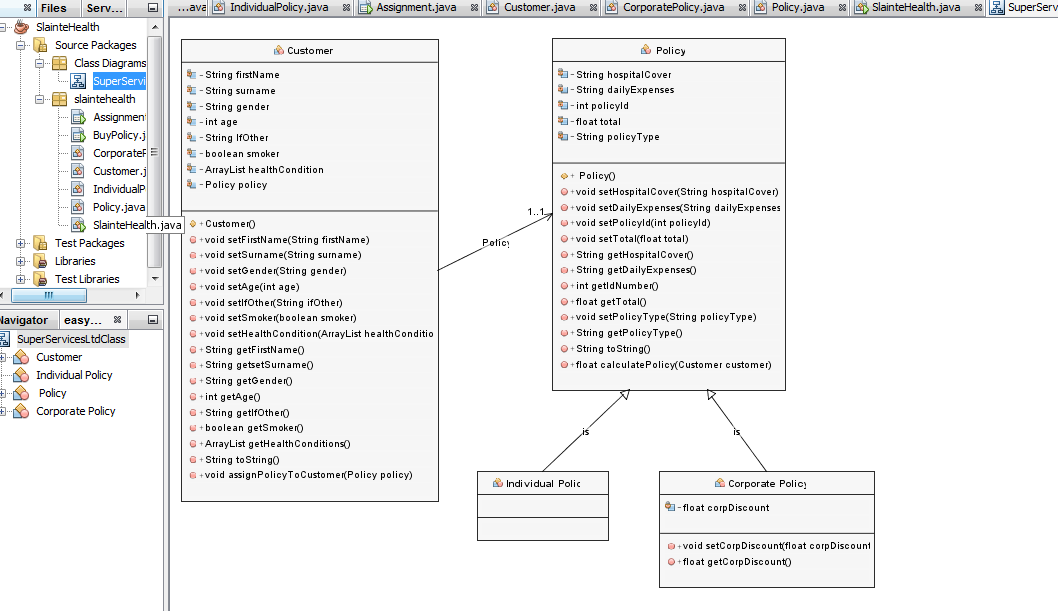
**Release a solution** to the devised problem

Carefully document which team member is responsible for each release and review and document the key differences between each release. Take a screenshot of each release and make sure to **submit the final released version with your code**.

**Release Version Number: \_ 1**

**Team member responsible: Steven**

**Screenshot of release:**

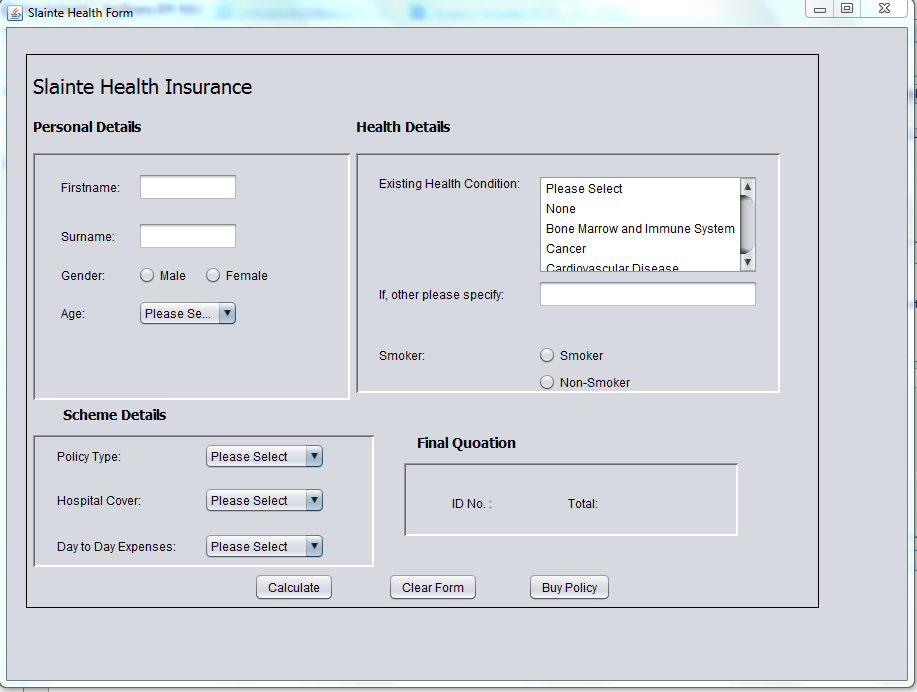


We used this UML class diagram to generate code for our programme. We designed it to have a plan of what classes are in our programme.

**Release Version Number: \_ 2**

**Team member responsible: Gavin**

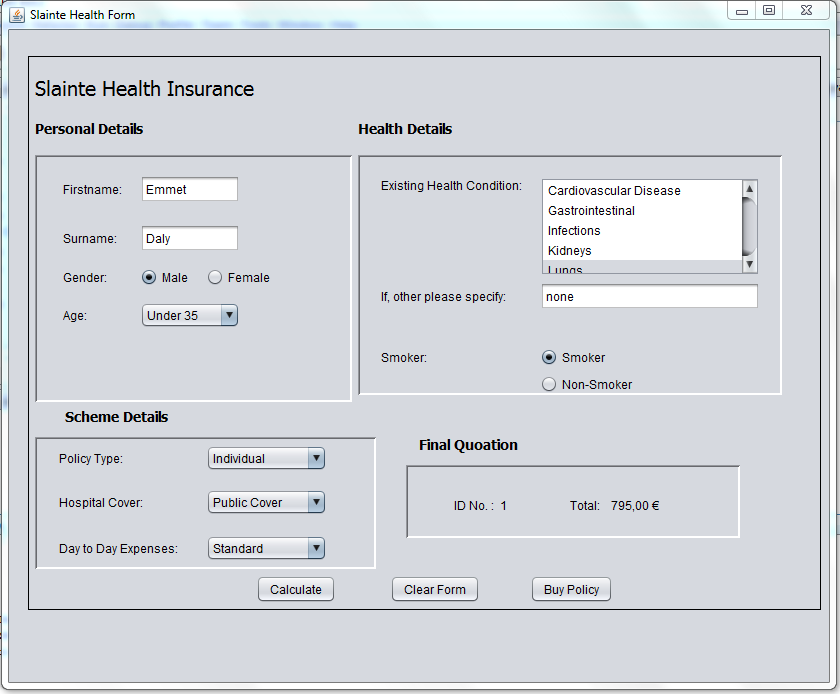
**Screenshot of release:**



We created a GUI Form to have a place for the user to input data in a clear and well designed format. We coded for the back end of our programme in our customer and policy class. We tested our calculations after each stage.

**Release Version Number: \_ 3**

**Team member responsible: Laura**

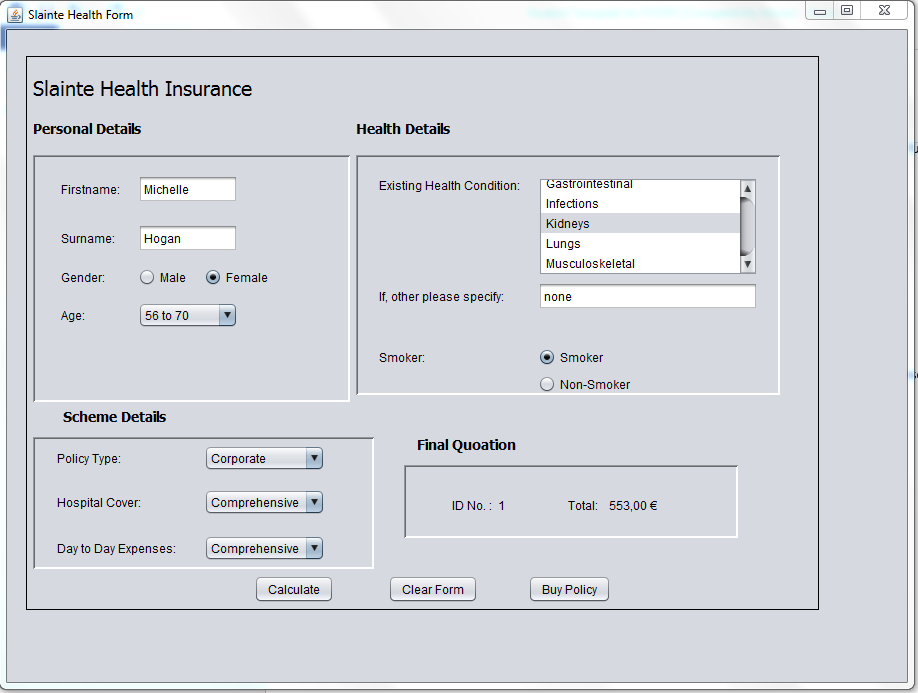
**Screenshot of release:**

We created a calculate button that would give the user a total for their policy and an ID number unique to them.

**Release Version Number: \_ 4**

**Team member responsible: Robert**

**Screenshot of release:**





We coded our clear form button which reset the form and incremented the ID number. The buy policy button was coded to bring up a jDialog window to show all the inputted data.

**Review**

**Key issues discovered:**

**Sprint 1:** The class diagram would not read in a constructor method.

**Sprint 2:** The buttongroup did not work properly when i was creating the GUI form.

**Sprint 3:** Getting the Over 70 to return the quote to 0 and end the running of the program. Getting the ArrayList to function correctly.

**Sprint 4:** Getting the user Input transferred into the jDialog form. The health conditions and smoker were the most difficult while total and idNo were also a challenge.