Dependable Systems (Team Beta) : Programming

For the application, it was decided that a desktop application will be developed in Java using FX and Maven. This decision was made by carrying out a SWOT Analysis:

Desktop Application

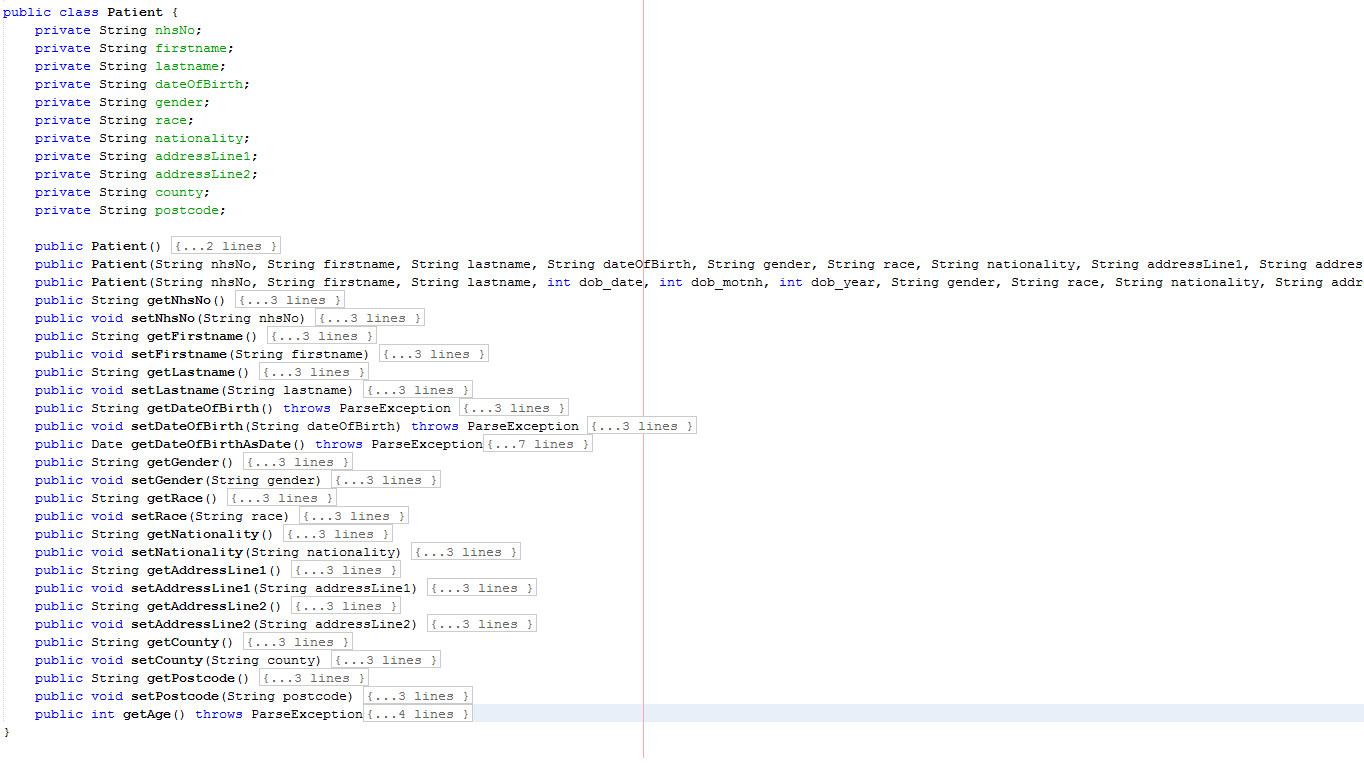
|  |  |
| --- | --- |
| Strengths | Weaknesses |
| * More Secure (less vulnerable to Attacks) * Cheaper to Develop (no costs for hosting) * Programmers and testers are familiar with desktop application development in Java Swing and FX as well as Maven * More time can be spent on developing rather than learning a new language * Testing can be done on all levels, i.e. from simple unit tests, to GUI tests to usability tests * Application will work without internet * No need to spend too much time on design (HTML, CSS, BootStrap, JavaScript, JQuery etc.); this is suitable for a tight schedule * Easy to debug, using IDEs such as NetBeans | * Must be installed on every PC of the hospital * Updates made by one clinician/admin will not be accessible by other admin/clinician * Users may be less willing to install software onto their PCs |
| Opportunities | Threats |
| * The application will not be affected by internet connection: more likely to retain users | * Possible competitors who may be more experienced may provide a better solution |

Web Application

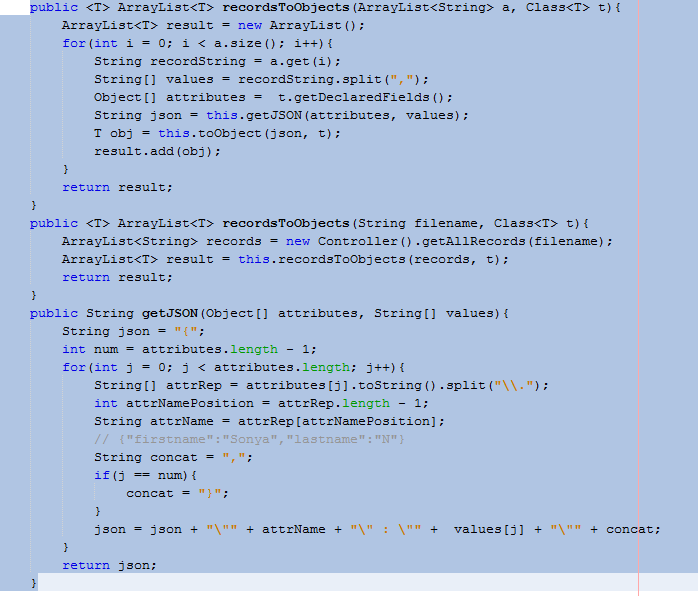
|  |  |
| --- | --- |
| Strengths | Weaknesses |
| * Can be accessed anywhere, provided that there is internet connection * No need for installation on every computer * Updates can be accessed by any authorised user | * Programmers’ knowledge in web development is mainly in PHP, which is not as secure as Java * Requires internet connection: availability may be poor * Security is worse in web applications as data is shared over networks |
| Opportunities | Threats |
| * Links to social media may attract new users | * More vulnerable to attacks: More likely to lose users |

All work done were pushed into a GitHub repository so that members of the team could view the application from home.

Model Package

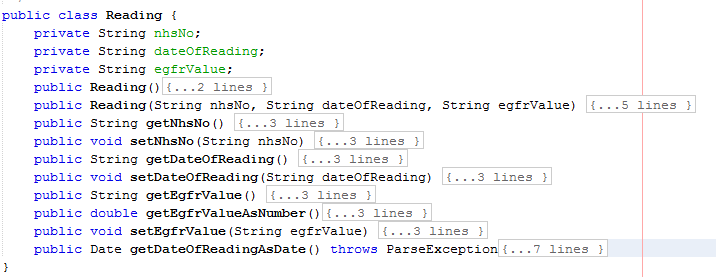


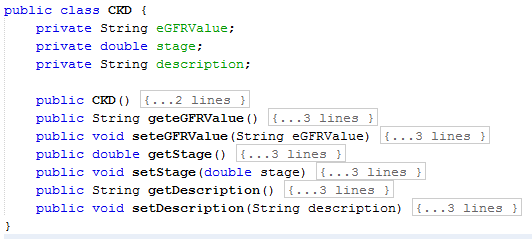
All attributes were stored as Strings; getters convert data to the appropriate data type. This was done deliberately as a method was developed to read files and convert each record into an instance of a specified class. This method can be found in the MavenFX project, which is a project that we developed over time, which prevents us from having to re-invent the wheel. Using this project, it was possible to keep our code tidy, and prevent repetition of code.



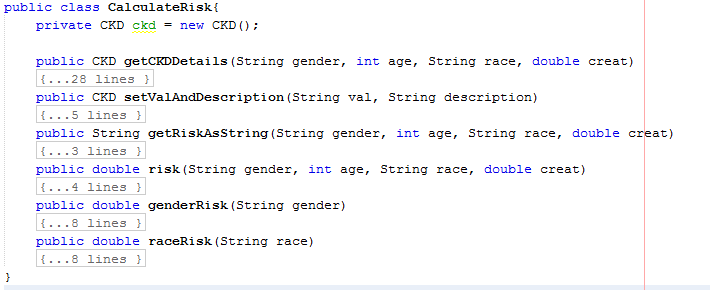
Examples of reusable code that we have developed over time that can be found on the MavenFX project:

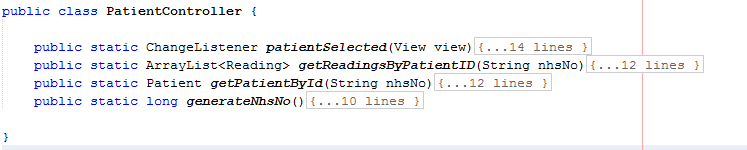
* AddressField
* NumberField
* Double Field
* Calendar
* Combo
* Controller
* Converter
* CurrentDateDisplay
* DateDisplay
* DateFeatures
* EmalField
* LinearChartView
* ImageBox
* MonthsCombo
* MyArrayList
* MyPasswordField
* MyTextField
* Checkboxes
* RadioButtons
* Table
* View





Controller Package







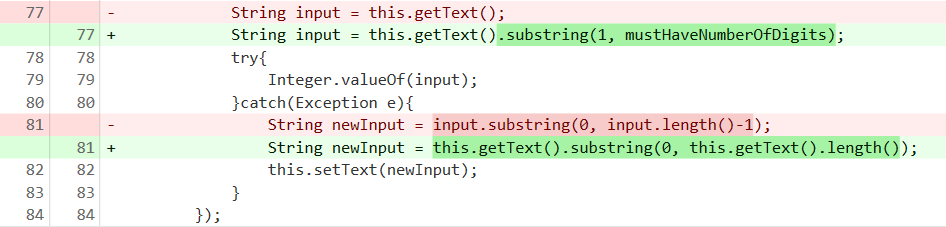
The getPatientById() method demonstrates the usage of the MavenFX project, where data in the Patients.csv file was converted to instances of the class Patient.

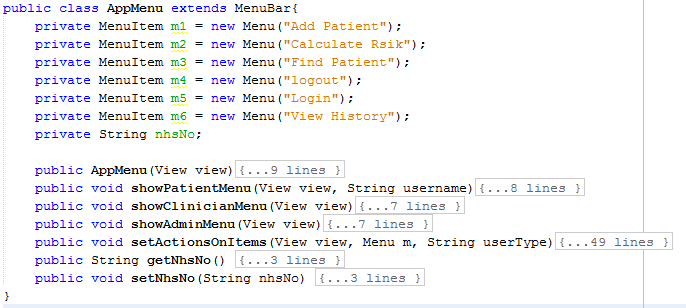
The generateNhsNo() initially declared nhsNo of an int data type. However, when sample data was provided, the method did not output the expected result. Having done some research, it was concluded that the NHS numbers, that were provided were too long for an int data type, and that a data type called *long* was to be used.

View Package

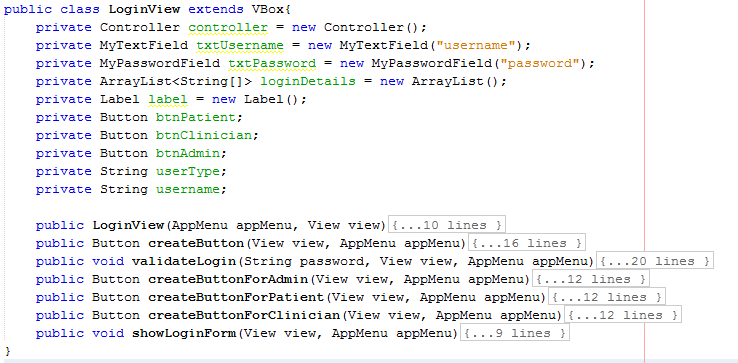


One of the issues identified was related to the number of digits allowed for telephone/mobile numbers. It was found that converting a string that starts with a zero, was impossible; the zero is not ignored. Therefore, some changes to the NumberField class was made:





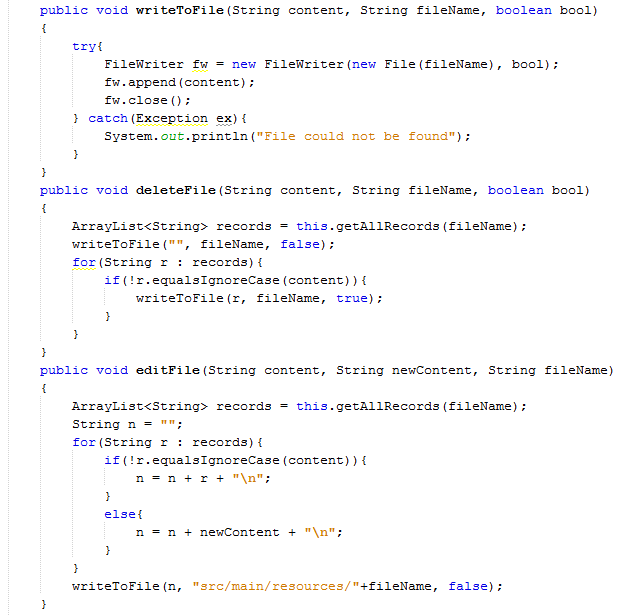




All passwords were encrypted using the MD5() hashing algorithm. On login, the menu changes according to the user.



The constructor that takes in a final argument to determine whether or not the user should be able to update/view the patients’ history. Clinicians and users will both be able to view a list of patients in a table and search for a patient by names, NHS number, address etc. Clinicians will be provided with extended functionality; on selecting a patient, they will be able to view history and update patient information. Here are the methods responsible for writing to, deleting from and editing a file:





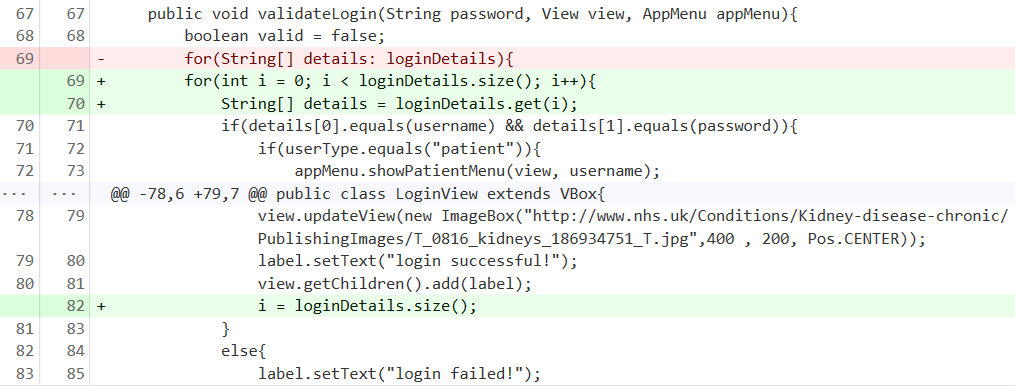
Validation has been put into place to:

* Help users recognise and correct errors

For example, if they enter their date of birth as 29-February-1995, the year field will turn yellow, indicating that this is not a valid date. NumberFields ensure that the field only accepts

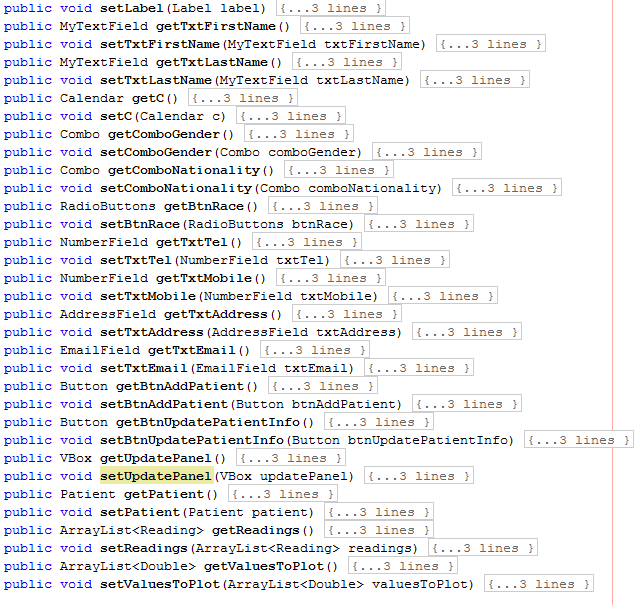
* Prevent users from making a mistake

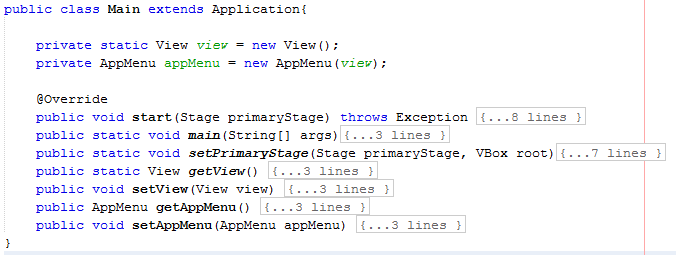
For example, combos have been used to prevent spelling errors and invalid input

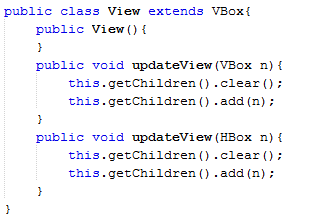


One of the testers pointed out that on logging onto the system, there is a message that reads “login failed!” instead of “login successful!” Having debugged the feature, it was clear that the text inside the label was being reset to “login failed!”, as it went through the for loop. This problem was addressed by escaping the for loop by setting the variable *i* to loginDetails.size(), once a matching record was found. This was the rationale behind switching to a for loop from a foreach. This not only solves the problem, but also improves the efficiency of the code by removing the unnecessary iterations that follow.

Other Methods Implemented in PatientView.java:







This class allows easy updates to the view, removing the need of clearing the panel before adding an element.

Features Implemented:

* Login System
* Any user can calculate their risk of CKD by entering their age, gender, creatin level and race
* Patients can view their eGFR records
* Administrators and clinicians can search for patients by NHS Number, name, address, race etc.
* Clinicians can update patients’ details and add new creatin levels (eGFR is calculated by the system to prevent human error). On inserting a new reading, the eGFR table is updated; patients’ results can be viewed as a table or graph