
University of Glasgow | School of Computing Science

# A Hands-on Approach to Learning Molecular Biology Techniques

Ross Eric Barnie
Dmitrijs Jonins
Daniel McElroy
Murray Ross
Ross Taylor

Level 3 Project — 18th March 2013

**Abstract**

Abstract-like things

Acknowledgements

First of all, we would like to thank our supervisor, Dr Gethin Norman, for his guidance, organisational skills and support throughout this project. We would also like to thank Dr Nicola Veitch and Dr Pamela Scott for their assistance with all matters relating to PCR and Primer design.

## Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

Name: _____ Signature: _____

Name: _____ Signature: _____

Name: _____ Signature: _____

Name: _____ Signature: _____

Name: _____ Signature: _____

# Contents

# Chapter 1

# Introduction

## 1.1 Preliminaries

There are some terms that will be used later in the report that should be clarified now, so as to avoid confusion. The aim of the project is to creating a teaching tool for PCR, or Polymerase Chain Reactions. This is the process of amplifying a sequence of DNA thousands to millions of times. It should be explained that DNA sequences are made up of two strands, comprised of bases of the nucleotides Adenine, Thymine, Guanine and Cytosine, represented by the letters `a`, `t`, `g`, and `c` respectively. Base pairing is when one base bonds with its complement on the other strand. `a` and `t` complement each other, and `g` and `c` complement each other.

Primers, used to select the sequence for PCR in a given selection of DNA, are shorter fragments of DNA, usually between 20 and 30 bases in length. For use in PCR, a primer must be chosen from the ”left” of one strand (this is the forward primer) and the ”right” of the other (this is the reverse primer), and these must obey a number of rules, which are the focus of the teaching tool:

- Neither primer should self-anneal. This means that if the primer were to fold in on itself at any point in such a way that that more than 3 bases in a row on one side paired to the base on the opposite side, this primer would pair with itself and become useless for the purposes of PCR.
- The melting temperature of each primer, calculated in degrees Celsius using a simple mathematical formula involving the frequency of `a`s, `t`s, `g`s and `c`s, should be between 50 and 65°C, and within 2-3°C of each other.
- The forward primer should be unique within the first strand, and not appear in the second, complementary strand. Likewise, the reverse primer should be unique within and to the second strand.
- The percentage of `g`s and `c`s within each primer should be between 40% and 60%.
- The length of the primer should be between 20 and 30 bases.
- The same base should not be repeated several times in a row.
- The last base of each primer should be either a `g` or a `c`.

- The primers should not anneal to each other. This means that if the primers were put side by side, at no point of overlap should more than 3 bases in a row complement the overlapping primers base.

It should be noted that many of these rules are not precise, and do not involve rigid limits for success or failure. In fact, these ”rules” more closely resemble rough guides. Obviously, the nature of programming does not lend itself to ”rough guides”, and so we followed advice from Pamela Scott and Nicola Veitch on how best to structure these tests to effectively represent the imprecision of their boundaries.

It should also be explained that we are developing the teaching tool in Java, using Netbeans, a free IDE primarily designed to be used with Java, and developing the user interface with Swing, the primary Java GUI widget toolkit.

## 1.2 Aims

The overall aim of this project is to produce a piece of software to help Level 3 Life Sciences students taking the Molecular Methods course (taught by Drs. Pamela Scott and Nicola Veitch) learn about PCR and Primer Design Techniques and to allow them to test their knowledge of these subjects. At the outset of the project, Scott and Veitch helped us to separate this aim into key tasks to be completed and important aspects of the interface design to be implemented:

1. The software should work as an interactive tutorial which users can work through. This requires:

    - A number of areas for users to enter their own choice of data, such as a choice of DNA sequence to work with and the primers with which to operate on the selected strand. For this feature to be useful as an educational tool feedback must be provided upon data entry.
    - Users should be able to experiment with different data e.g. examining the different melting temperatures of different primers. This requires the ability to easily move forwards and backwards between the different stages of the tutorial.
    - To help newer users and students who are unfamiliar with PCR there should be simple instructions to guide users through the process and explain PCR throughout the application. There should also be a page displaying the rules of PCR and primer design which should be available at all times.

2. The software should be accessible to all users with a basic understanding of molecular biology, regardless of their different levels of knowledge, ability etc.:

    - To achieve this, the interface should be uncomplicated and intuitive without compromising the required functionality. This will be aided by the instructions and help section mentioned above as well as labels placed next to any areas users can interact with.
    - Any section which makes use of colour should be designed with colour blind users in mind.

3. The software should improve upon the tools currently available for learning primer design. The main issues with these systems are:

    - The low level of interactivity offered by the systems, such as the numerous YouTube videos available on the subject [**?**]. Users who are not actively working through a tutorial or a demonstration are likely to lose interest faster so it is important to make them involved with every step of the tutorial by having them design their own primers etc.
    - The available tools rarely go into detail about primer design specifically. One example of an interactive, well designed application that fails to convey the process of designing primers to a satisfactory degree is University of Utah's "PCR Virtual Lab" [**?**]. Therefore, an important aim for the project is that primer design must be explained in detail and provide enough information to be informative, whilst remaining interesting to students using the system.

4. Another aim related to accessibility is that the users should be able to download and use the software from home. This means that the program must be able to run on a variety of different operating systems and computers with varying performance levels. With this in mind it was decided that the program should be written in Java due to it being highly portable.

## 1.3 Background

## 1.4 Motivation

When selecting a project at the outset of the course, we identified several factors associated with this project that motivated us to take it on.

Chief among these factors was the aim of building an interactive teaching tool. Some members of the team expressed an interest in going on to create educational software after completion of their degree, and this project would serve as ideal experience in developing such software.

Another part of the project that excited us was the opportunity of working within the university to potentially improve the education of our peers. Several members of the group have friends on the course in question, and these friends have provided valuable feedback along with their colleagues. Another aspect of the involvement of Drs. Scott and Veitch was in gaining valuable experience in client relations. Several of the projects on offer, while interesting, did not involve any stakeholders other than their supervisors, and so we felt this project would be a unique opportunity to put into practice the lessons we had learned from other courses about requirements gathering, without the risk associated with the involvement of an external business entity, for whom the consequences of failure might be more severe.

Lastly, the element of the project that excited us the most was the chance to do work related to a field that we had absolutely minimal experience with. The sum total of biology-related experience on the team was Ross Taylor's Higher qualification in Biology in secondary school, and that placed us in a great position to learn about certain elements of molecular biology from an outsider's perspective.

As previously explained in section 1.3, the project came from a dissatisfaction from the teaching staff of Molecular Methods with the current method of teaching PCR, but in order to better under-

stand what it is the lecturers sought after, research into PCR education systems currently in place became a necessity.

### Current Systems

In order to understand the motivation for the development of the system, Drs. Scott and Veitch provided us with links to several systems currently in place which attempt to make learning this process more interactive and/or visual. However, videos and multimedia in general have been questioned as teaching aids in the past [**?**]. As expressed in this paper, simply because the information is in video or multimedia format does not necessarily mean that it is benefiting the learning of its viewers, or creating the correct environment to encourage learning. Interactivity, along with other factors, are key to engaging people to learn.

The first was a video hosted on YouTube [**?**], made by demonstrators within the School of Life Sciences. During its eighteen second duration, the video shows various elements of the PCR process including change in temperature and the role of the primer. However, it was commented by the team and by the clients that it was insubstantial in terms of information delivery, several of the stages of PCR are omitted with no mention of primer design, and in terms of interactivity.

Another video hosted on YouTube [**?**], currently referred to on School of Life Sciences' website, is similar in style to a lecture with slides and a voice-over which repeats the textual information on each slide. While this video is far more informative than the previous one, with each stage of PCR clearly described, and with visually pleasing animations, it lacks in explicit primer design and again in interactivity.

Finally, an animation from the University of Utah, titled "PCR Virtual Lab" [**?**]. This is a much more interactive experience and allows the user to use virtual pipettes in order to simulate what you would do in a lab situation when performing PCR. Additionally, the information it provides, while slightly basic in the beginning for our target users, is extensive and very informative to the novice user, such as Biology-illiterate Computing Scientists. While this is a much more interactive and, compared to the alternatives described above, much more informative experience, it fails to provide the user with the theoretical background information, particularly on primer design (required to fully understand the process and why the reaction occurs), and does not allow the user to test their ability to select good primers, the most difficult aspect of PCR.

# Chapter 2

# Design

## 2.1 Requirements

### Inital Requirements Gathering

The requirements gathering process for the application began immediately. At the first meeting, our clients presented us with a document outlining what it was that they wanted from the end product, including a very early step-by-step walkthrough of the application they envisioned. This proved to be a key tool in bringing us up to speed with what the should accomplish, and really sped up the initial requirements gathering phase. Obviously, this design was altered and adapted throughout the project, but the steps served to provide a rough guidline that we followed throughout development. The aim of the project, as described in the aforementioned document, are as follows:

> To design a PCR-primer design exercise to complement teaching of a Molecular Methods course to Level 3 Life Sciences Undergraduates. This exercise will be integrated into a new part of the lab which we are designing based around diagnosis of HIV using PCR. You will need to understand the theory behind PCR and primer design in order to achieve this.

This statement alone is helpful as it tells us about our userbase, where and how the application will be used, and what background knowledge is required in order to thoroughly understand the premise of the project.

On the subject of background knowledge, along with this document, we were given the Molecular Methods lab book, in order to see how Primer Design is currently taught in the course and get a better idea of how it worked ourselves.

Finally, within the first two or three meetings we were sent links to various multimedia teaching tools for Primer Design, as described in Section 1.4. Along with our own research, this gave us an informed view of what else is out there, the positives and negatives of these current approaches, and what we could improve upon in our own product.

From these first few weeks of meetings with the clients, we drafted a requirements document, which was presented to the clients and agreed upon, and presented the following requirements:

**System Scope**

- The main aim of this system is to act as a teaching tool to aid students in learning how to design primers for PCR experiments and should be usable in a teaching environment or by people on their home computers.
- It should function as an interactive, step-by-step guide through the process of PCR on a DNA sequence of the users choice. The user is required to access the NCBI website and copy and paste their choice of DNA sequence into the system. The system should provide feedback if the user enters incorrect primers. The system should then check if the melting temperatures of the primers are in the required range. The user is then given a link to perform primer blast to check if the primers they have chosen are unique.
- The system should also provide the user help with completing each task by providing relevant rules for each task and giving the user instructions about how to use websites and resources outwith the system (NCBI, primer blast etc.).
- When the user has provided an appropriate pair of primers the system will then show an animation of the PCR reaction taking place.

**Non-Functional Requirements**

- The system is expected to be used at students homes or in the Biology lab computers, so portability is essential for the system to work to the clients expectations.

**Design Feedback**

Throughout the project, we maintained a weekly meeting schedule with our supervisor and clients, and despite scheduling difficulties at least one of the clients was present at every one of these meetings. This allowed us the opportunity to improve our design iteratively through multiple pitches, internalising the feedback given over the following week to produce a design more in line with their requirements.

Over the course of these meetings, the clients provided us with

When the team had formed a solid idea of the layout and flow of the system, we drew up some early mockups of the system's user interface (discussed in further detail in Section 3.2

**Implementation Feedback**

## 2.2 UI

# Chapter 3

# Implementation

## 3.1 Team Distribution

Before the team began implementing the application, we decided to split the team into three smaller sub-teams, in order to maximise the use of everyone's time. These groups were to:

- Design and implement data models and associated custom methods
- Implement the graphical user interface
- Design and implement an animation to show the process of PCR

The team's lead programmer, Daniel McElroy, took the lead on this decision and, while noting each member's particular preferences, decided to split the team in the following way:

**GUI** Ross Eric Barnie, Murray Ross

**Data Models and Custom Methods** Daniel McElroy, Ross Taylor

**Animation** Dmitrijs Jonins

While it may have been unnecessary to assign a team-member entirely to the animation, the team felt that Dmitrijs would work best on his own and meant that the rest of the team could work as they had done up to this point, as a team.

## 3.2 User Interface

The implementation of the graphical user interface (GUI) required a number of decisions to be made before writing it could begin.

### 3.2.1 Programming Language

When discussing implementation, the group quickly settled on Java as the language in which to implement the application, due to our collective experience with it as a consequence of the Java Programming course taken in the previous academic year, and our knowledge of existing GUI frameworks that would suit our purposes.

### 3.2.2 GUI Framework

From a brief research period at the start of the implementation process, we settled on two possible options for a GUI framework to use for the application. It is important to note that other GUI options are available, but based on the team's experience, it became clear that Swing or JavaFX would be the most suitable.

#### Swing

Each member of the group had some limited experience with the Swing framework, though not all of it had been positive. The experience each member of the team had with Swing varied, and although every member had agreed that their experience had not been entirely problem-free, we conceded that its integration with the Netbeans Integrated Development Environment (IDE), discussed in section 3.2.3, was extremely useful.

However, on investigating the framework more closely it was clear that Swing was extremely well documented with full API specification [**?**], and in-depth tutorials [**?**]. This was a huge part of our decision as we felt that the documentation provided would be more than adequate to allow us to use the framework with relative comfort.

#### JavaFX

Another framework considered was JavaFX which no member of the team had any experience with. Some members felt that this was a risk worth taking, given how much they disliked Swing, discussed above. In reality, JavaFX was only briefly considered and totally disregarded when, upon brief investigation, JavaFX was still a relatively new framework, and consequently, comprehensive documentation was not as readily available for JavaFX as with Swing, particularly when it came to troubleshooting on online forums.

In addition, JavaFX required Java 7, which, again, no member of the group had used before and which was not available, at the time, in the Level 3 Laboratory where we would be working for the majority of the year. It seemed like too much of a risk to try to learn two different technologies at the same time, while having to provide our own development platforms, which, with various members of the team never having used the Linux OS before, could potentially cause a number of problems.

#### Decision

The investigation was carried out by the group's Toolsmith, Ross Barnie, who presented the evidence discussed in the sections above regarding the two frameworks to the rest of the team. With this evidence the team voted in favor of using the Swing framework with Java 6.

Retrospectively, Swing, and Java 6, are out-of-date technologies and JavaFX is now packaged with Java 7 [**?**], so the application would have been more up-to-date or future-proof had we used JavaFX. Additionally, (some of) the computers in the level 3 lab now do have Java 7 installed upon another project team requesting it, so our fears over development platform problems were nullified, though this was only after we had started development.

It was an unfortunate shortcoming of the research into JavaFX that the group did not know about JavaFX's integration with the Netbeans IDE which was seen as one of the key differences between the two frameworks at the time of making the decision.

### 3.2.3 Integrated Development Environment (IDE)

One concern was that, in some members' experience, using two separate IDEs was extremely time consuming, particularly while using version control. This was mostly due to various metadata that IDEs keep track of in various files, however this meant that any small change to the source code would change the metadata and therefore each commit would have to involve adding it, which would be very time-consuming.

It is because of this experience that the group decided to work from a single IDE, researched again by Ross Barnie.

#### Netbeans

Netbeans is an IDE which the team had had little experience with and had only used in the context of building applications with GUIs created using the Swing framework. There was some trepidation to using Netbeans since most of the team had associated their problems with Swing with Netbeans itself. Upon further research, which involved using the IDE to build small applications, Netbeans started much faster than Eclipse, discussed below. And the design interface was very simple and easy to use, with each element being laid out the way you wish and the associated source code being generated for you. This meant that the design layout could be finished very quickly, rather than spending our time writing hundreds of lines of source code just for the interface.

In terms of Netbeans' metadata, it was quite minimal and would not clutter the version control repository to an unacceptable degree.

#### Eclipse

The team had substantial knowledge of Eclipse from its mandated use in Java Programming 2 [**?**]. Again, our experience of Eclipse is somewhat tainted by associations with problems we faced at the

time, such as a bug on the version for Windows which meant that Eclipse would freeze if you tried to copy or paste anything.

In our experience, we found Eclipse to be very slow, both during start-up and normal operation. Editing-wise, Eclipse was rather cumbersome and had few benefits over a text editor. Also, the requirement to bind the "Workspace" was seen as a potential point for confusion and errors.

In addition, the team felt that the missing design interface seen on Netbeans, discussed above, was a huge disadvantage and would cause a significant loss of time, simply due to the volume of code we would have to write instead of being auto-generated.

Members of the team also pointed out that Eclipse has a tendency to create a large amount of metadata which would clutter the version control repository.

#### No IDE

It was briefly considered to have no IDE at all and simply use text editors. This would allow for extremely fast editing in a very comfortable environment, since most text editors, such as Vim [**?**] or Emacs [**?**], are highly customisable and can launch in a matter of seconds. Text editors would also not require metadata, keeping our version controlled directories clean.

However, the obvious problem with no IDE is that troubleshooting source code problems without any real-time error-checking like in IDEs is more difficult and, unlike with IDEs, you cannot automatically import a missing package or method, nor can there be any auto-generated code at all for that matter.

#### Decision

When the evidence above was given to the team, we were also discussing which GUI Framework to use (as discussed in section 3.2.2) and it became obvious that integration with the framework would be key to helping us develop the GUI.

We therefore decided to work with the Netbeans IDE because of the design interface, minimal metadata, and lack of (known) bugs that would affect us in any meaningful way.

Retrospectively, this was the correct decision. Even if we had chosen a different GUI framework, the advantages of the easy-to-edit design interface far outweigh any problems we had with it.

### 3.2.4 Builds

To demonstrate the GUI and the changes we made to it over time, we will discuss two builds of the system at two crucial points in time.

The first is what the team refer to as the "demo build", which was the first build of the system in general to be used by anyone outwith the project. The demonstration itself is discussed in more detail in section *REFERENCE TO DEMONSTRATION FEEDBACK*.

The second is the current build of the system, which is currently linked to on the Molecular Methods moodle site to be used by any of its 160 students. This build by nature has developed from the demo build in that most of the changes made were based on the evaluation and feedback we received from the demonstration itself (discussed in section *SECTION REFERENCE*)

### Demo Build

**Splash** Before the demonstration (discussed in section *REFERENCE*), the team were asked to include an "overview" screen to tell the user what they can expect from the application, as well as show the primer design rules to remind the user about them. This can be seen in figure 3.1.

It's design is to maximise the separation of ideas, so the Overview section is to the left, which due to the way English is read, is the more likely of the three sections to be read first, at least by fluent English readers. In this overview section it was decided to include contact details of the team in case the user found technical problems with the program since, as discussed in section *REFERENCE FUTURE WORK*, the team plan on maintaining the system for future use.

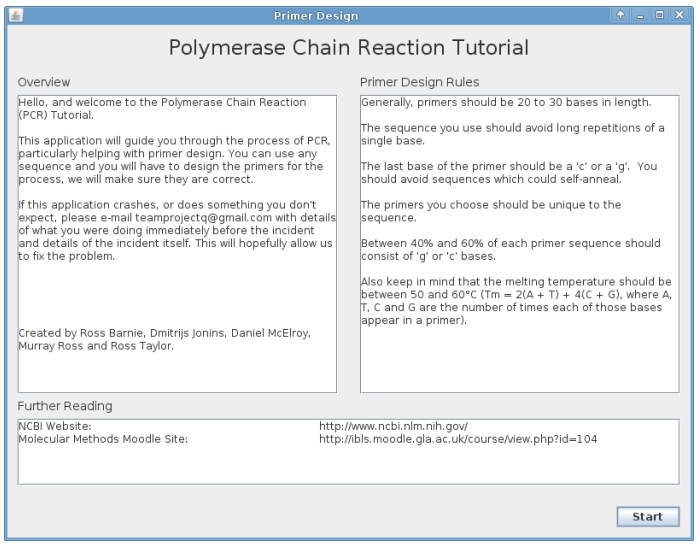


Figure 3.1: Demo Build, Overview Panel

**Sequence Entry** Figure 3.2 shows the next panel, referred to by the team as the "sequence entry" panel. While based on the design in figure *REFERENCE INITIAL UI DESIGN* it has been altered slightly to maximise the amount of space to be used for entering in the sequence, as this is the primary purpose of this panel.

It was expected of the user to go to the National Center for Biotechnology Information (NCBI) website and obtain a DNA sequence by copying it to their clipboard and then pasting this into the sequence entry panel and this was explained in the accompanying user guide (appendix *USER GUIDE APPENDIX*). Although this relied heavily on the users' ability to use keyboard shortcuts, it was assumed that all students at university level would at least have an awareness of these shortcuts. We also assumed that once students were told of these shortcuts, as they were in the user guide, that they would be comfortable using them.

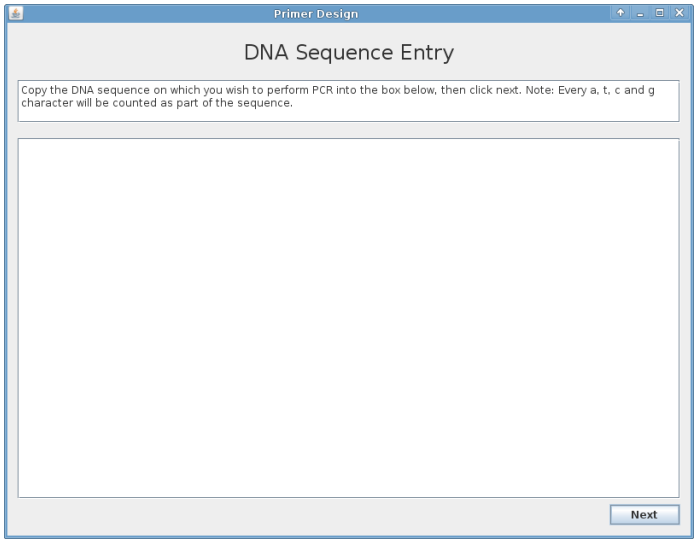


Figure 3.2: Demo Build, sequence entry panel


Figure 3.3: Demo Build, Area Selection Panel

**Target Selection** Following the Sequence Entry panel is the "Area Selection" or "Target Selection" panel, seen in figure 3.3, which requires the user to specify the "target" sequence, ie the desired output sequence of the PCR process. This is accomplished by the user entering the start of the sequence that they want and the end of the sequence they want, both by the index of that base in the sequence, into the "From" and "To" fields at the bottom of the panel.

This would, ideally, be helped by the text pane at the left of the screen which shows the base number of the first base on its line. Unfortunately, for an unknown reason, the text panes became misaligned when viewed from any platform other than the one we were using for development (the level 3 Computing Science Laboratory computers running Scientific Linux) and this misalignment can be seen in figure 3.3.

An addition made to the design discussed in *REFERENCE DESIGN* is the tabs above the main text area, which allow the user to switch between the sequence they entered, and its complementary equivalent, generated by the program. It was a suggestion by the clients to have this feature as it would greatly increase the speed at which the user could design the reverse primer. Without the complementary tab, not only would the user have to manually convert the primer to its complementary equivalent, but also reverse its order, which neither the team or the clients felt was a useful way for students to spend their time.

**Primer Design** Following the Area Selection panel is the "Primer Design" panel, shown in figure 3.4, which allows the user to enter forward and reverse primers.

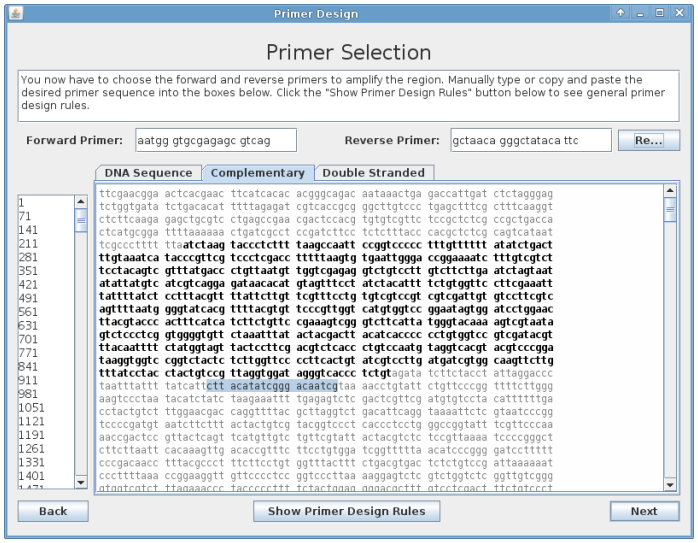


Figure 3.4: Demo Build, Primer Design Panel

One of the design features we had intended to provide was "dynamic highlighting", as it was referred to by the team, which was going to provide a highlight around what the user enters into the primer text fields. This highlighting was unfortunately missing in the demo build due to time constraints.

However, we had always intended to give feedback to the user should they break rules of primer design and the demo build version of this can be seen in figure 3.5 and appears when the user clicks the "Next" button. Again based on the design *REFERENCE*, this dialogue window shows the user any rules which they have broken, and which primer the feedback is referring to.

Again, due to time constraints this was not as fully featured as we had hoped for in the demonstration, however it did display enough information to give an idea to our clients of what the feedback might look like in the future (see further discussion in section *REFERENCE DEMO FEEDBACK SECTION*).

In order to design a reverse primer outwith the system, the complementary strand would have to be calculated (which would be in the 3'—5' direction) and reversed to be in the correct direction (5'—3'). In the application, the complementary strand is already calculated, so the user can simply copy and paste a primer of their choosing from the sequence and put it in the reverse primer text

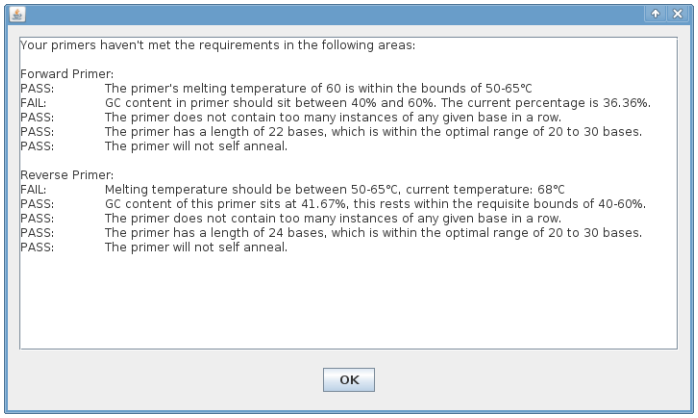


Figure 3.5: Demo Build, Feedback on User-entered Primer

field. However, this does not solve the problem of reversing it, so the "Reverse" button was put next to the reverse primer text field, which, intuitively, reverses the order of the primer in the text field.

At the bottom of the panel in the middle of the "Back" and "Next" buttons is the "Primer Design Rules" button, which shows, intuitively enough, the primer design rules set out at the beginning of the program. This was part of the initial design *REFERENCE* and can be seen in figure 3.6.

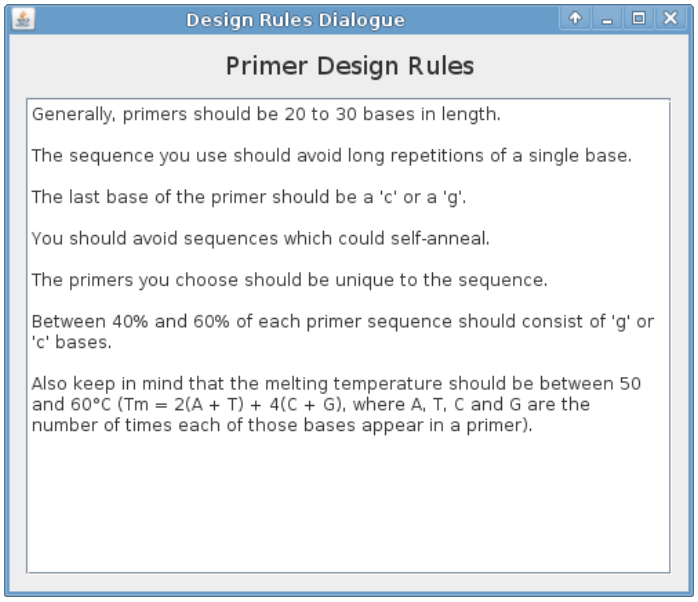


Figure 3.6: Demo Build, Primer Design Rules Dialogue

**Melting Temperature** After designing a primer, the user is presented with the "Melting Temperature" panel as seen in figure 3.7 for the user to evaluate their primers.

In the case of the demonstration, this panel was blocked from the user unless they had a correct primer.

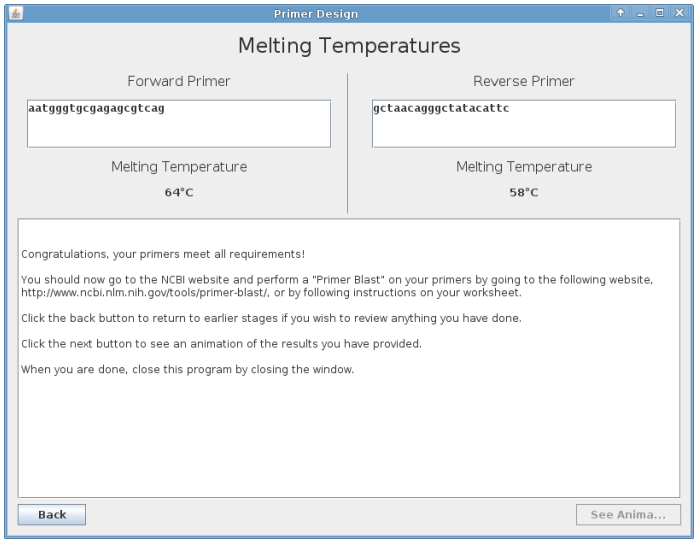


Figure 3.7: Demo Build, Melting Temperatures of User's Primers.

In terms of design, the initial design proved to be near-impossible to reproduce in Swing and make it look professional and after several iterations became what it is. The emphasis on the primers and the melting temperatures in bold means that the user can easily see their primers and the associated melting temperatures, while the separator down the center visually separates the forward from the reverse primer.

Unfortunately the animation was not available in time for the demonstration and although the button for it was included in this panel it was disabled.

### Current Build

Addressing feedback from the demonstration (see section *REFERENCE*), and adding new elements to provide extra functionality, the current build is several iterations ahead of the demo build discussed above and is currently available to all Molecular Methods students.

While there have been many changes to the build in this time, not much has been changed in terms of the UI as most feedback about it at the demonstration was positive. Most changes stem from either additional requirements or from the demonstration feedback.

**Overview** The overview screen (seen in figure 3.1) changed only because of the addition of the menu bar, discussed below. It was felt that this needed very little change, if any, from the demo build and is therefore the same as the demo build version in almost every respect.

**Sequence Entry** In terms of changes made after the demo build, this panel is similar to the Overview panel above in that neither have been changed to any significant degree.

Though, as can be seen in figure 3.8 there have been some minor changes.

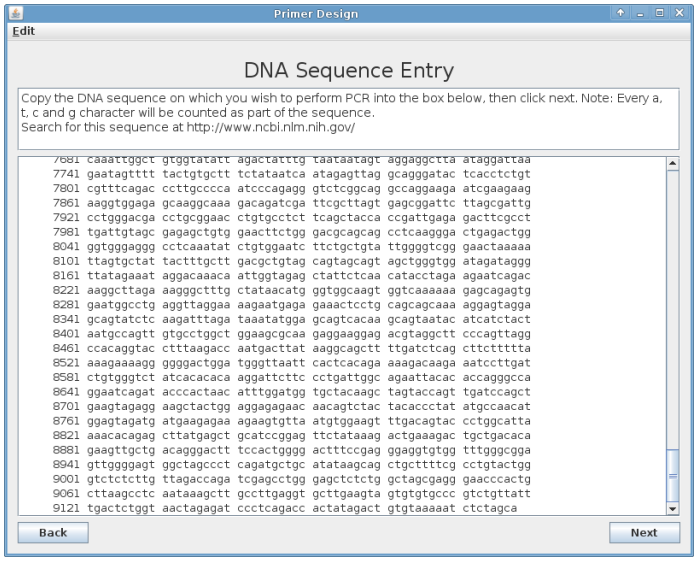


Figure 3.8: Current Build, Sequence Entry Panel

Firstly, the menu bar at the top of the panel, which was added to address the issue of people not being familiar with keyboard shortcuts. The menu itself was named “Edit” to conform to standards set by most applications with menu bars in that the “Copy” and “Paste” features are usually found under the “Edit” menu.

Lastly, the “Back” button on this panel which was not present for the demo build. For the demo build, it felt unnecessary to have a back button since any information that people would actually need to use the program is available throughout the application (ie Primer Design Rules is available on Primer Design panel, NCBI website is given to user in the instructions at top of panel). However, the clients asked that we put the “Back” button in.

**Target Selection** While the interface of the Target Selection panel may not have changed (much) from the demo build, the way a user pulls their desired sequence has changed to make the process much easier.

Previously, as discussed in section 3.3, the user would have to find the indexes of the start and end of the sequence they wanted to copy. Now, as can be seen in figure 3.9, the user simply has to highlight the desired sequence and the indexes are calculated automatically. This change means that users do not have to depend on the unreliable line numbers, and can focus instead on the actual sequence. It also provides a visual way of seeing where your sequence is, without having to switch between this and the next panel, which was one of the issues brought up in the demonstration (see section *REFERENCE DEMO EVAL*).

**Primer Design** Primer design received a lot of attention in that it received the majority of new features in the application since the demo build.

Firstly, a feature referred to by the team as “dynamic highlighting” which, when a user types in a sequence to the primer text fields, highlights that sequence within the whole sequence and this


Figure 3.9: Current Build, Target Selection Panel

can be seen in figure 3.10. Initially, this highlighting was going to be a single colour simply to help the user see if the primer is unique to the sequence, and so that they can visualise where the primer is within the sequence. The lack of visual aids in the demo build was something we strove to rectify especially after the feedback from the demonstration (section *REFERENCE DEMO FEEDBACK*). To this end we also made the highlight change colour depending on the correctness of the primer, how this is decided is discussed in section *REFERENCE DYNAMIC HIGHLIGHTING SECTION*. This can be seen in figure 3.10 with an incorrect primer, and figure 3.11 with a perfect primer.

Another new feature added based on the demonstration feedback was the two new buttons at the bottom of the panel, which give feedback on a single primer, depending on which button the user presses. An example of what that feedback looks like can be seen in figure 3.12.

This feedback is also colour-coded to make any problems with the user's primer(s) more immediately apparent than in the demo build.

With the emphasis on direction of the sequence being made more obvious in this version with the direction shown in the tab names, it is more obvious as to which direction the user should be thinking about when creating the reverse primer. To allow the user to only have to think about one direction, the "Reverse" button from the demo build was replaced by an uneditable text field which auto-generates the reverse order of the reverse primer, see figure 3.13.

When the user presses the "Next" button, they are presented with a dialogue box, giving a list of all the passed, failed and "close-fail"-ed rules for the primers individually and the more general rules (see figure 3.14. Again, these are colour-coded.

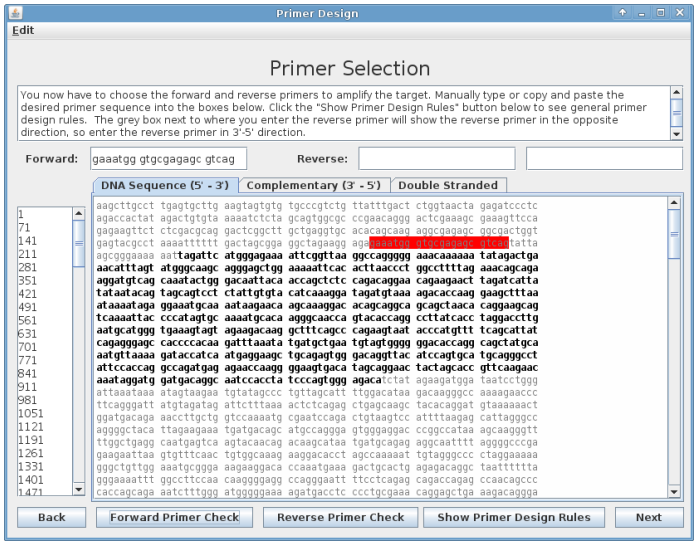


Figure 3.10: Current Build, (Forward) Primer Design, showing invalid primer selection

## 3.3 Models & Custom Methods

### 3.3.1 Models

The models used in the application are few in number and very simple, BLAH BLAH BLAH BLAH

As seen in Figure 3.15, we used three classes to represent the data: Primer, TestResult and Sequence.

**Primer** The Primer class is purely designed to test user-designed primers. It has one attribute, `code`, the String representing the user's primer which is tested against in the primer test methods, which make up the remainder of the primer. The class is primarily made up of methods designed to test `code` against the various rules described in Section 1.1. The method `test()` gathers runs all above test methods and returns a larger TestResult, indicating if the user's Primer is adequate outside of the larger context of the sequence.

**TestResult** TestResult is a class used to format the output of one or multiple primer tests. TestResult uses an enumerated type called PassState with values `PASS`, `FAIL` and `CLOSEFAIL`, the last of which describes a state where the primer's value from a test lies outside of the recommended values, but is close enough to a pass to be acceptable, provided this is only the state of a minority of tests. TestResult uses two ArrayLists, one of PassStates (`passes`) and another of Strings (`out`), to keep track of the state and informative message to be displayed to the user for each test.

Its methods are concerned with concatenating results into larger TestResults. `perfect()` will return true if all entries in `passes` equal a `PASS`. `adequate()`, the method that is checked to gate the user's access past the Primer Selection panel returns false if any of the tests returned extttFAIL, or if more than 60% returned `CLOSEFAIL`, and returns true otherwise.

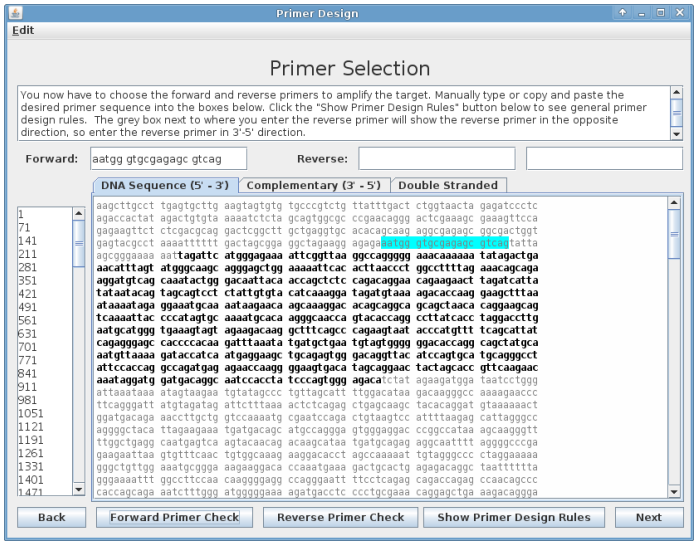


Figure 3.11: Current Build, (Forward) Primer Design, showing perfect primer selection

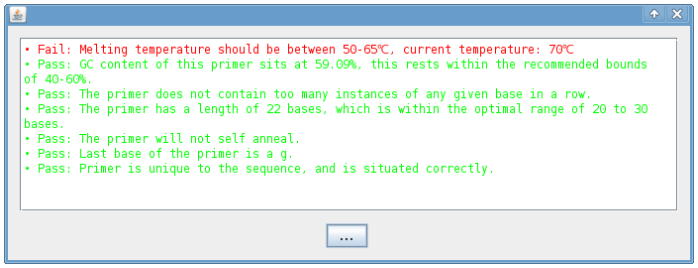


Figure 3.12: Current Build, Single Primer Feedback

**Sequence** This class contains two Strings, `oStrand` and `cStrand`, representing the strand of DNA that the user took in and the "complementary" strand that is generated when the sequence is constructed. Integers `start` and `end` represent the indexes of the start and end of the selected area in the sequence, and Primers `fPrimer` and `rPrimer` are, obviously, representations of the user's primers.

`parser()` is used for both sequence entry and primer input, by taking in a String and returning a new String with all non-`atgc` characters removed. `complement()` is a very simple function used throughout the application that takes in one character representing a base, and returns its complement, i.e. `complement('a')` would return `t`. `isUnique()` and `tempDifference()` check the user's primers against the rules concerned with the larger Sequence. `primerTest()` uses all other test methods to return a TestResult in accordance with the

### 3.3.2 Primer Checking

The 'Primer Checks' methods are implementations of the established rules and guidelines which are used in the process of Primer Design (seen in the preliminaries section) to evaluate the effectiveness of a given Primer when used in the PCR process.

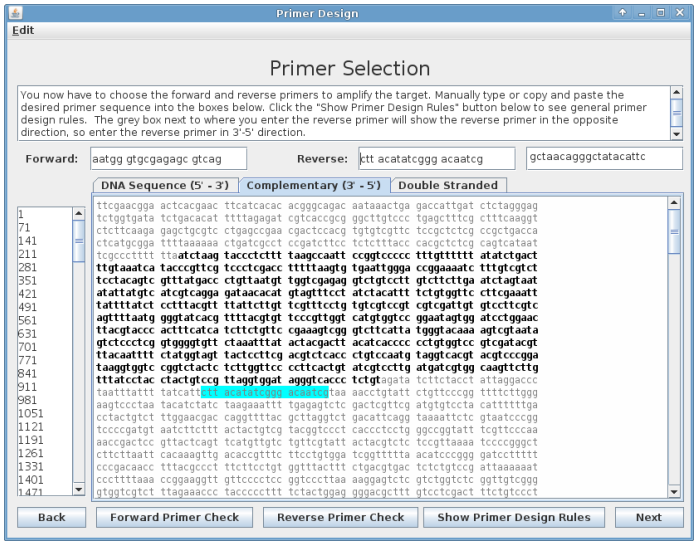


Figure 3.13: Current Build, Reverse Primer (showing perfect primer)

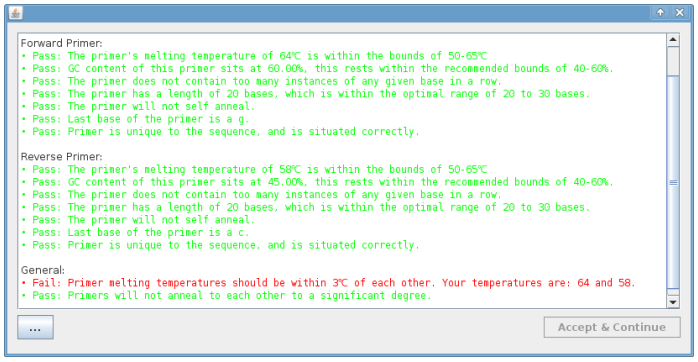


Figure 3.14: Current Build, Feedback on both primers dialogue

As with many other aspects of the project, the primer checks were split between the two members of the 'back-end' sub-team. As well as making this task more manageable, this approach offered the added benefit of limiting the researching of primer design rules to 2 members, who each only had to learn how to apply about 4 methods each.

The complexity of these rules varied greatly in difficulty, from trivial checks such as Primers should end in a base 'g' or 'c' to challenging checks such as checking how likely it is that one end of a primer will anneal to the other. The majority of these methods fell into the first category and were fairly straightforward to implement, however, implementing the more difficult methods posed a serious challenge.

Firstly, none of the members of the team had any experience with PCR or Primer Design prior to the start of the project so every rule had to be thoroughly studied and understood before beginning to design the methods. However, even after spending time learning how the design rules and guidelines are used, some methods still proved problematic.

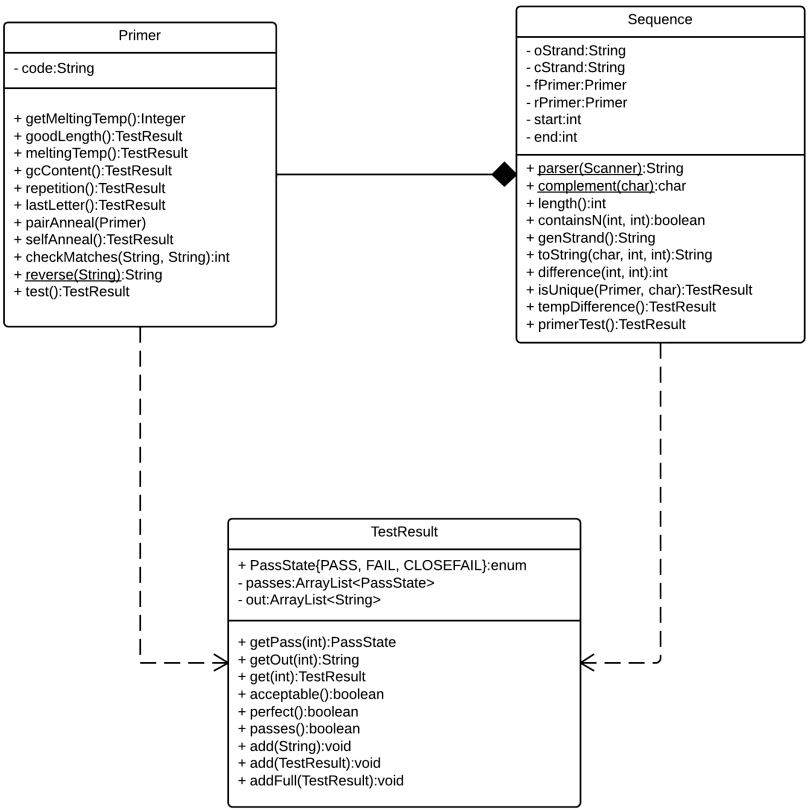


Figure 3.15: Model Class Diagram

**NEED SECTIONS FOR THE INTERESTING/DIFFICULT METHODS*** Several of these methods were trivial to implement, including `gc` percentage, appropriate primer length

The melting temperature was among the easiest to implement, having been given a standard formula for calculating the temperature at which a primer would melt (2*sum(`a, t`) + 4*sum`a, t`) and the desired range in which a primer should rest.

### 3.3.3 Dynamic Primer Highlighting

Dynamic Primer Highlighting was suggested early on in the requirements elicitation process by the clients as something that would be very helpful to students studying primer design. The initial specification for this feature was as follows:

As the user enters their choice of primer in one of the boxes at the top of the page, instances of the primer should be highlighted in real-time in the corresponding box below containing the DNA sequence for the strand on which this primer should appear.

In a later client meeting the following addition was made to improve the effectiveness of the feature:

Primers should be highlighted in different colours to indicate their suitability.

This was a feature that all members of the team liked from a very early stage because, even with our limited knowledge about primer design we could see how this form of immediate feedback had the potential to improve the usability of the system if it were to be implemented well. Due to this level of popularity among both the clients and the team members the feature was given high priority. However, since we knew it would be complex to implement and would require calls to other planned modules of the system, such as the primer checking functions, it was also decided that this should be one of the last features to be implement.

This feature turned out to be one of the most problematic features to implement as it involved using features of Java which we had no real experience with, specifically the Swing classes `ChangeListener`, `Highlighter` and `Painter` as well as integrating other modules of our code to provide the required primer checking. This level of difficulty meant that this feature actually took multiple attempts to implement correctly.

***NEED SECTION ABOUT THE PREVIOUS ATTEMPTS***

### Final Solution

Ultimately, the final implementation of this feature can be split into two main components:

- The code to listen for user input in the primer entry text fields and call the appropriate methods to deal with the input.
- The runnable objects `searchO` and `searchC` which are called by the listener and update the highlighted text.

### Listening for Input

The first challenge we were met with when implementing this feature was deciding how to listen for user input in the `forwardPrimerTextField` and `reversePrimerTextField`.

Due to the fact that only one strand's display tab would be shown at any given time, only the TextField related to the active tab would have to be listened to and any user input in the other field could be ignored until the active tab changed. This meant that only one listener would be needed and the TextField it was listening for updates on could be switched when required.

Since this feature was implemented late on in the development cycle, code already existed to deal with changes upon switching tabs. Therefore, we were able to add the calls to switch the TextField being listened to into the existing `updateLineNums()` function which was only being used to update the line numbers upon switching between single and double-stranded views.

As well as switching the target of the listener upon switching tabs, a a call must be made to the search function related to the new tab since any user input into the corresponding TextField while the tab was not active will not yet be reflected in the display tab. This call is made using and `invokeLater()` call of the runnable search function as oppposed to a standard `invoke()` call

to allow the system to continue listening for updates if a call to one of the search functions takes a long time to process.

Once we had worked out how we were going to switch the listener focus all that was left to do in terms of listening for input was to add code to the `DocumentListener` functions `insertUpdate()` and `removeUpdate()` to run the appropriate search function. All this required was an if statement to determine the source of the update and inside the if statement a call to the appropriate search.

### Runnable Search Functions

After dealing with listening for user input, the next task was to create the functions to search for the user's primer in the DNA sequence and highlight the appropriate sections.

To perform this task we created two similar methods: `searchO()` to update forward primer highlights and `searchC()` to update reverse primer primer highlights.

The first task these functions perform is to remove all existing highlights from the display. The other option here was to keep a list of all highlights and their positions in the sequence then check each one individually to determine if they were still valid after the latest input and highlight an extra base at the start of end of the old highlight. This list based solution was not chosen because, for example, if a user's first input was a single letter then the program would potentially have to store the positions of thousands of bases and perform thousands of comparisons on their next input.

The next step is to get the position of the first instance of the primer in the sequence using the java `indexOf()` function. This is then used as part of the condition of the while loop which carries out most of the functionality in these methods. The while loop is excecuted while the result of the `indexOf()` call is not negative (i.e. the primer is found in the text being searched) and the user input has a length greater than 0 (i.e. The call to the function was not because the user deleted everything in the text field).

Inside the while loop the colour to highlight the current instance is determined by the following if statement:

```
if (sC.length() > 15){
    Primer rPrimer = new Primer(Primer.reverse(sC));
    rTest = new model.TestResult();
    rTest.addFull(rPrimer.test());
    rTest.add(
        rPrimer.isUnique(PrimerDesign.start.getInSequence(),
            'c'));
    if (rTest.perfect()){
        activePaint = perfectPaint;
    } else if (rTest.acceptable()){
        activePaint = acceptPaint;
    } else {
        activePaint = failPaint;
    }
} else {
```

```
        activePaint = failPaint;
    }
```

Here, `sC` is the user's primer input and `perfectPaint`, `acceptPaint` and `failPaint` are different coloured `Painter` objects initialised when `PrimerSelectionPanel` is first loaded. The only difference between the three `Painter` objects is their colour with `perfectPaint` being blue, `acceptPaint` being yellow and `failPaint` being red. These were initially green, yellow and red but green was eventually substituted for cyan due to concerns about colour-blind users.

The late stage at which this feature was implemented meant that the process of evaluating the user's primer was a trivial task as all the methods check the primer were already written, as were the methods to evaluate the success of the primer overall. So the method simply has to create a new `Primer` instance with the user's input and create an instance of `TestResult` to store the results of the primer checking methods. Then the method calls the `perfect()` and `acceptable()` methods to determine the colour the primer should be highlighted.

Once the colour of highlight has been determined the highlight is applied to the correct section of the DNA sequence using the following code:

```
endC = indexC + sC.length();
highC.addHighlight(realIndex(indexC + checked, 10),
    realIndex(endC + checked, 10), activePaint);
```

`endC` is calculated by adding the length of the user's primer to the index of the current instance of the primer in the sequence, `indexC`.

### 3.3.4 Other Methods

# Chapter 4

# Evaluation

## 4.1 Testing

## 4.2 Introduction to Testing

As with any software development, testing is a necessary component. During our time with the PCR application, testing begun almost as soon as implementation started. Our testing fell into two categories:

- Testing of the backend, such as primer checking formulas etc.
- Testing of the user interface.

Also, with reference to the (edit ME)[**?**], we accomodated for both white and black box testing.

### 4.2.1 White Box Testing

White box testing was used throughout the development of the implementation, as we all had an indepth knoweledge of the source code. This enabled us to test for defects by creating scenarios which tested the programs to its limits and beyond, in order for us to assur

### 4.2.2 Demonstration

### 4.2.3 Questionnaire

In order to test the final system against the requirements set out in section *REFERENCE REQUIREMENTS*, the team decided to produce a questionnaire which would be given to students

using the application for them to give us feedback. The raw data for this feedback is shown in appendix A.

While other methods of testing exist (e.g. experimental) it was felt that with the time constraints in place we would not be able to run a test requiring our direct involvement and that we would receive more data from a questionnaire.

### 4.2.4 Questionnaire System

Many questionnaire services exist on the internet, such as SurveyMonkey [**?**], FreeOnlineSurveys [**?**], and Google Docs Forms [**?**].

The team decided to use the Google Form, simply because most members of the team have a Google account, and because it was free.

On closer inspection it could also export its responses to a Google Docs Spreadsheet which itself could be exported to a variety of formats.

### 4.2.5 The Questions

Generally, the questionnaire needed to be as concise as possible to avoid any data being corrupted by frustration at the questionnaire. To this end we kept the number of questions in general to a minimum and only asked for a description if it was necessary.

#### Skill Level

At the demonstration, the team realised that not every user will be in the expected age bracket of 18 to 20-years-old who have used computers their entire lives. This led to the decision that the feedback should take the user's age into account, to ensure that there are no patterns in the data suggesting a particular age group struggles with the application.

In addition, we ask the user for their "Confidence" with computers in general, on a scale from 0 to 5 ie no middle option so there has to be a bias to confident or not confident. This completely subjective question should allow us to see if people who perhaps do not use computers on a daily basis handle the application, since we believe we made the application (and accompanying user guide in appendix **??**) as user-friendly as possible.

We are also aware of some colour-use in the program that could be problamatic for colour-blind people so we ask the user if they are or not.

To get an idea of the user's knowledge of primer design, we ask the user for their self-assessed understanding before using the system, on a 0 to 5 scale.

All of this data provides us with the baseline skill level with computers and with primer design of the user, before they use the system.

**User's Experience with the System**

The following questions were designed to provide us with feedback on the user's experience with the system.

The first of these follows the self-assessed understanding of primer design before using the system, with a self-assessed understanding after using the system, on the same scale. This will provide us with test data towards the requirement to teach students about primer design and PCR.

Ideally, as discussed in section *REFERENCE REQUIREMENTS GATHERING*, the system would be used as a revision tool in students' own time and/or in a laboratory setting with tutors on hand to help the student with any problems. To see if we met this requirement, we asked the user if they would use the system to study one of the following options:

- Both Primer Design and PCR
- Just Primer Design
- Just PCR
- Neither

Which provides the user, and the team, with every possible answer to the question, in the most concise way possible.

### 4.2.6 Feedback Analysis

# Chapter 5

# Conclusion