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 —

Abstract

Abstract-like things

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 [6]. 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" [7]. 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 [4]. 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 [6], 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 [5], 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" [7]. 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

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, including of course, how the team would be distributed between implementing the GUI and the data models.

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's important to note that other GUI options are available, but upon further analysis, it became clear that Swing and JavaFX were the most suitable to our needs.

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 [1], and in-depth tutorials [3]. 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 [2], 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 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

11

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 not much better than 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 problems becomes very tedious very quickly, and unlike IDEs, you cannot automatically import a missing package or method, nor can there be any auto-generated code 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*)
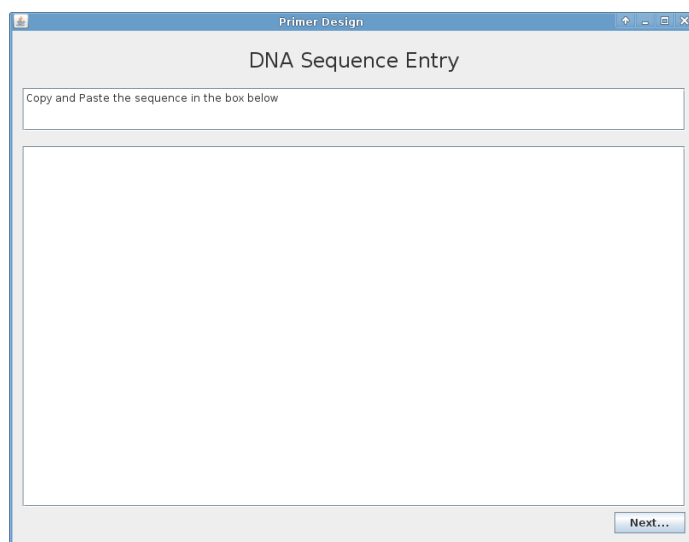
Demo Build

Figure 3.1: Demo Build, sequence entry panel

Start Figure 3.1 shows the first panel, referred to by the team as the "sequence entry panel", to appear when you run the application. 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.

Area Selection Following the Sequence Entry panel is the "Area Selection" panel, seen in figure 3.2, 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
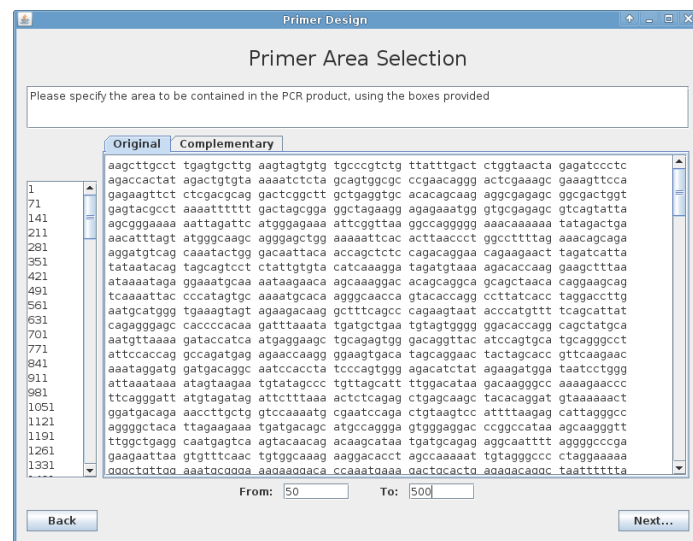
Figure 3.2: Demo Build, Area Selection Panel

level 3 Computing Science Laboratory computers running Scientific Linux) and this misalignment can be seen in figure 3.2.

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 3.3, which allows the user to enter forward and reverse primers.

One of the design features we had intended to provide was "dynamic highlighting", as 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 3.4.

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. In terms of user-friendliness and design it could have used a lot of improvement however this was a feature still in early development.

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*)
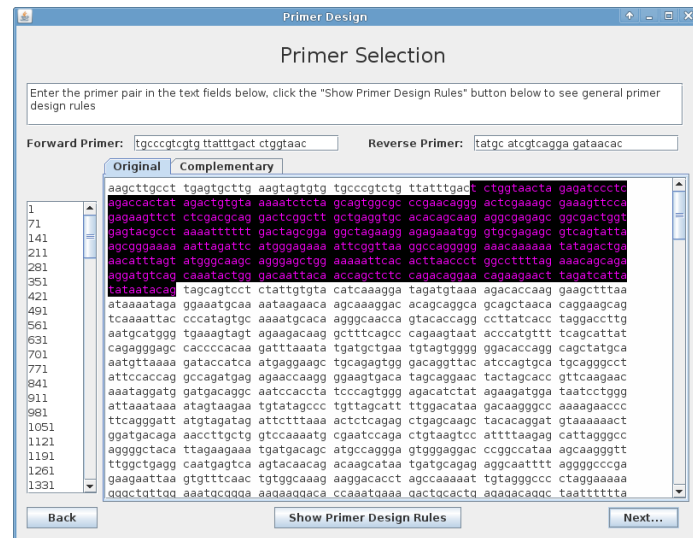
14

Figure 3.3: Demo Build, Primer Design Panel

Melting Temperature   After designing a primer, the user is presented with the "Melting Temperature" panel as seen in figure 3.5

3.3 Models & Custom Methods

3.3.1 Models

- 3 objects, Sequence, Primer, TestResult

Basic in construction, all classes have toStrings and booleans.

Sequence: 2 strings to represent each strand of the sequence, ints to represent the start and end of the desired PART of that sequence, 2 Primer objects.

Complement method taking a base and returning its complement, isUnique tests if primer is unique (durr), tempDifference compares the temperature of both primers, primerTest combines all methods that return a TestResult and produce an overall TestResult, the toString of which should be shown in the Primer Evaluation Dialog.

Primer: Basically a class to test user primers, string to represent this primer, and all primer tests methods, each returning a test result. Method named test() adds all of these TestResults together when run against the code String, and returns a larger TestResult to be used both in the individual primer checks and in Sequence's primerTest method.

TestResult: A class to format output of one or multiple primer tests. Uses an enum called PassState with three states, PASS, FAIL, CLOSEFAIL, indicating the outcome of a particular test. An ArrayList of these Pass States is used to keep track of all individual results, and an ArrayList of Strings contains each test's corresponding message.
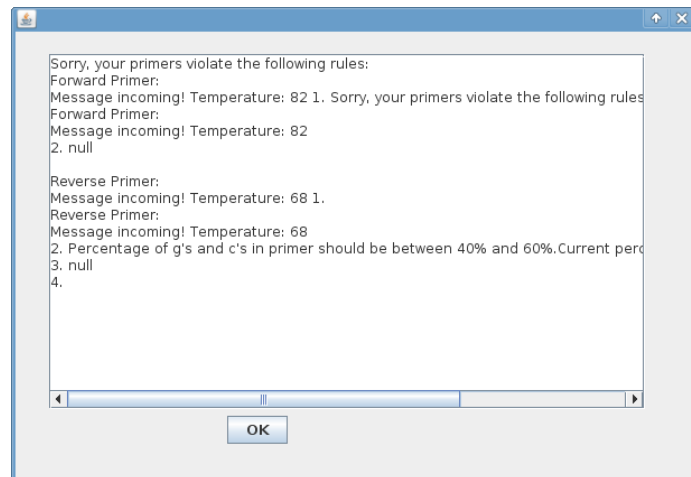
Figure 3.4: Demo Build, Feedback on User-entered Primer

Methods used to add TestResults together and access or mutate various elements of the ArrayLists.

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.

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.

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.
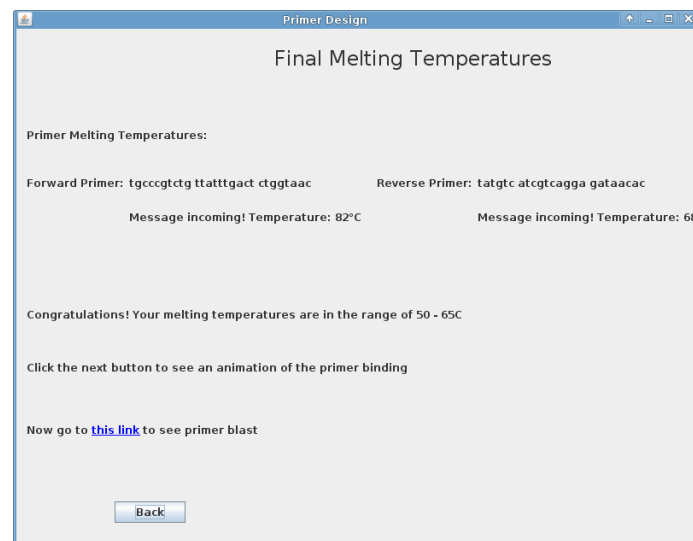
16

Figure 3.5: Demo Build, Melting Temperatures of User's Primers.

3.3.3 Other Methods

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.

Ultimately, 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

17

'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

NEED SECTION ABOUT THE FINAL SOLUTION TO PROBLEM

Chapter 4

Evaluation

4.1 Testing

4.1.1 Demonstration

4.1.2 Questionnaire

Chapter 5

Conclusion

Bibliography

[1] Java Swing API summary.

[2] JavaFX overview. http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm.

[3] Trail: Creating a gui with jfc/swing. http://docs.oracle.com/javase/tutorial/uiswing/.

[4] Nick DeKanter. Gaming redefines interactivity for learning. *TechTrends*, 49(3):26–31, 2004.

[5] User 'DNALearningCentre'. Polymerase Chain Reaction (PCR). http://youtu.be/XXkG6m3yT1M, 2010.

[6] User 'dwildridge'. Taq Extension (test).wmv. http://youtu.be/XXkG6m3yT1M, 2012.

[7] Genetic Science Learning Center. PCR Virtual Lab. http://learn.genetics.utah.edu/content/labs/pcr/, August 2012.