Rabbit Show Manager

Final Report for CS39440 Major Project

Author: Paul Victor Massey(pvm@aber.ac.uk)
Supervisor: Dr. Edel Sherratt(eds@aber.ac.uk)

09 May 2014

Version 1.0 (Final)

This report is submitted as partial fulfilment of a BSc degree in
Open Source Computing
Computer Science (G402)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature …….. (Paul Massey)

Date ……

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature …….. (Paul Massey)

Date ……

Acknowledgements

I am grateful to

Edel for putting up with me, giving words of advice that everything doesn't have to be perfect.

Adrian Shaw for actually accepting me to transfer directly in to the second year.

I'd like to thank the following members of staff for the help and support

Neil Snooke for the advice on the JList.

Lynda Thomas for forwarding an email to Neal for me.

Dave Price for having patience with me and my nerves during the midpoint demo

Pam Milner for help during the 2 years here as my DSA teaching team in student support.

Abstract

Many years ago I was show secretary of my local rabbit club, the club holds four or five shows every year. The process for holding a show starts with a committee meeting held up to ten weeks before the show to discuss the classes, to be included in the show, the judges for the classes, these are required to obtain the BRC (British Rabbit Council) Certification for the show. Even today the BRC uses a paper based system for the certification of shows and requires up to four weeks to process the request. Once certification is received the show must be advertised in the monthly journal Fur & Feather, this can take a further four weeks for the advert to appear.

The current system paper based it takes two days before the show, and one or two days after the show. Changes are difficult to make, the BRC show rules state that no sick animals are permitted in the show pens/cages, so an exhibitor would lose out if the entered exhibit became ill after the close of entries and before the show. This would also reduced the entry fees for that exhibit.

This project is to partly automate the process of requesting certification from the BRC, sending an advert to the F&F. To fully automate the taking entries from from exhibitors, calculating the entry fees, sectary's book, creating booking in/out sheets, judging sheets, taking the results from the judging sheet returns, calculating the prize money, and ensuring that the judges' fees or expenses are paid. Producing reports for BRC, F&F and the club members.

Rabbit show manager will allow a later closing date, and the possibility to change an exhibitor's entries as some classes are based on gender or age of the exhibit, or the fact that it is the exhibitor originally put the exhibit's ring on it's hind leg above the hock (ankle) at the age of about six weeks. The replacement exhibit may be different in gender, age or breeder status.

Contents

1. Background, Analysis & Process

1.1. Background

1.1.1. My interest

I was at the AGM of the Chelmsford City Rabbit Society and the previous secretary resigned in protest over a disagreement, and after the end of the AGM I thought about it and offered to become secretary, the committee agreed to me becoming show secretary and not full club secretary as I was under 20 years of age. In 1978 a long time before I had encountered home computers, there were a few home computers about and however most computers were run by big businesses such as financial institutions, universities, and government departments.

The Rabbit Show management was and from what I gather was of course paper based and a former club secretary guided me through the first show every step of the way, a little on the second show and I was left to ask questions on subsequent shows. Since discovering home computers it is something I have thought of producing a computerised system for managing a rabbit show, never having the time to make a real attempt until this project.

1.1.2. Other systems

Research of the internet was undertaken to find similar small animal show management software, and there were several products all for the American systems but there was not one based for the UK.

(“Laughing Rabbit Software and Systems,” n.d.)

(“SHOW SECRETARIAL SOFTWARE,” n.d.)

(“Show Beacon,” n.d.)

(“The Show Program,” n.d.)

The is not much point in discussing these systems as they are all for the American Rabbit Fancy.

American rabbits have a tattoo in their ears as identification marks whereas, the British Rabbit Council supplies light weight metallic rings, with a standard identification mark consisting of two digit year 0-99 a size letter A-L or X and a serial number of five digits 00001-99999. The American system from what I can tell is far less standardised. That would not be a problem to change the American system to BRC one. The one point I did notice though is all but one of the American systems were closed source proprietary systems, and most require an annual licence fee. There was one that was open source however that was completely web based.

1.1.3. The British Rabbit Council

After proposing this as a project I did contact The British Rabbit Council and their reaction was less than enthusiastic to say the very least. They said in an email in reply to a query in what they would want a system to do.

Quote from the email “there is a system in place for secretaries of clubs for show supports, show returns etc and less than half of our secretaries have computers.”

This I find hard to believe as a high percentage of population have at least one computer in the home. I also put some feelers out on social media such as Facebook, and have to date received nothing in reply to any requests. So I was on my own.

My philosophy I have believed in for a long while has been to adopt Open Source Computing which is free for all to use, and users are free if they are skilled in

programming to make changes as they see fit. Therefore the choice of database engine was PostgreSQL especially for a multi-user environment that could be used on a website. Furthermore it is supported on most popular operating systems, Microsoft Windows, Apple Mac OSX, Linux and Unix for no charge.

The choice of PostgreSQL was also made because of licensing conditions of other relational database engines. One of the other common relational database system is MySQL which is currently owned by Oracle and is an open source database engine however under some conditions there are payment requirements for licences to remain legal.

The choice of implementation language was the next to consider, the author's favourite languages are C, and Pascal however even thought there are compilers C and Pascal available there needs to be an executable file for each of the operating systems, so Java was chosen as the implementation language. As long as there is a JVM Java Virtual Machine for the Operating System, Rabbit Show Manager will run without further problem, even without the source code. There is even a Java Virtual Machine for the Raspberry Pi, a really cheap computer that is very low power, runs Linux and connects to a domestic TV for it's display.

1.1.4. Some of the business rules.

In the rabbit fancy there are two types of show classes breed classes, and duplicates, for each show, an exhibit must be in only one breed class, the exhibit can be entered in zero or more duplicate classes.

The breed classes are divided into sections of Fancy, Lop, Fur and Rex.

All Angoras must always have a top of the tier pen.

There can be a duplicate show class called “Upside down”, where the prize money goes in the reverse order to the first seven, places, the exhibitor that comes seventh gets what the normally the first place exhibitor would receive, sixth place gets second place's prize money, and fifth get third, This is a “gentlemen's” way of spreading the prize money.

There can be a breeders only show class, meaning that the exhibitor owned the exhibit when the it's identification ring was placed on it's hind leg, the rings are lightweight metal purchased from the BRC and usually put in place at six weeks of age, and worn for life the BRC needs to be informed. The ring is the rabbit's equivalent to a car registration number, if the exhibit is passed on to another exhibitor sold or given, the new own is not permitted to show in the breeder only class. Although there is no need to for the no checking of this is done of this fact at the show level and taken on trust, although the BRC have been known to check up, when the show results gets back to them, and can take action as the BRC see fit.

There is normally a club members duplicate show class that means that the exhibitor is a member of the club that is holding the event. There is normally no prize money for this show class as the prize money for this show class is automatically donated to club funds, and as such there are club specific rewards for the exhibitor that show in this class.

In addition to these classifications, show classes can be based on an exhibit's age or gender, or that of the exhibitor. These increase the number of show classes.

1.2. Analysis

The project is to emulate the paper based system is required to replicate the reports that would normally be completed on paper

As the author is the only customer at present and the process is still carried out in the same of very similar way.

1.3. Process

The design process is TDD (Test Driven Design) although it has not been rigidly adhered to. The database of course had to be designed up front, as the whole system depended on it and a database is not something for most extreme programming design methodologies. Although there has been some modifications to the database from time to time.

Little of the system works at present. Even with another development process such as the Waterfall the author does not expect anything would be different, as the main problem was understanding some of the Java Swing components and how they worked. Although the business logic is not complete, the ideas are firm fix in the author's mind, there just was not sufficient time once the final advice came from the project supervisor.

2. Design

2.1. Overall Architecture

The overall architecture is a PostgreSQL database with thirteen tables. Using version 9 of PostgreSQL.

The development of the Java was carried out using Netbeans 7.4 IDE Integrated Development Environment.

The forms designer within Netbeans was used to create the forms.

The predicted Website was to use PHP/Javascript/CSS, however, as there has been significant development in Java with classes known to be working which could be used for a Java serve-let in conjunction with Javascript and CSS.

The Android application which will be developed in the future using Eclipse IDE.

2.2. Detailed Design

There are two abstract classes, one for the individual data records and one for the lists of the data.

There are three Interfaces, one for the data, one for the lists and one for the forms.

Enumerated types two, that are really substitutes for boolean that make the source more self documenting.

There is a class for each of the database tables, a list for each of those classes and a form to be able to edit the data in that table. The reason for this is standard type of user interface, is so that shows of a similar nature could be catered for, the other shows the author had in mind at the start of the project and through, are cavies also known as Guinea Pigs, rats, mice, and bantams. There would need to be some logic changes, but the overall design was to allow the user to be able to change most of the data so align the application for their own animals. It may be though inappropriate to have a database table with only few records such as one for the exhibit gender but rabbit genders are buck and doe, a bantams are cock and hen, cavies are boars and sows. There are other simple tables in the database that are similar. The exhibit age for rabbits is slightly more complex as there is at least one breed of rabbit that is considered adult at four months of age, whereas most are youngsters until the age of five months.

2.2.1. Even More Detail

In the design of the database records, it was considered that using individual bits of integer as flags for some of the conditional parts rather than individual boolean entries, although this has not completely replaced the boolean fields at present. that using bit manipulation would help to keep things simple for the automated entry generation and inserting any valid records in the junction table entries. The basic idea is for each of the ShowClass to compare each exhibit and exhibitor against the requirements for the ShowClass and add an entry in the entries table if the values are correct.

This probably better done in the application code, as the information of the exhibitor and their exhibits is entered in to the system, although the database is capable of doing this as a batch process with the correct series of SQL statements.

For example in the exhibitors gender show class gentleman has a value of 1 lady has value of 2 and therefore a show class for the gentlemen only will accept when the exhibit have set their gender to gentleman, show class for ladies only accept exhibitors have set their gender to lady, show classes set to open accept both and a gender neutral open of stud name here the gender option has no bits set.
The show class can have the bit values

Bit values	Meanings	In the show class record	In the exhibitor record
00	A BRC registered stud or Gender not disclosed	Not used	Registered Stud or Not Disclosed
01	Male	Gentlemen	Gentleman
10	Female	Ladies	Lady
11	Open	Open	Not Used

Exhibitor Genders

Bit values	Meanings	In the show class record	In the exhibit record
000	Not disclosed	Not Used	Exhibit age not disclosed
001	Under 14 Weeks	For exhibits under 14 weeks of age only	Exhibit is under 14 weeks of age.
011	Youngster (under 4 months or under 5 months) depending on breed	For exhibit non adult exhibits u14w, u4m or u5m	Exhibit is still under adult age
100	Adult	For Adult exhibits only	Exhibit is an Adult

Exhibit Ages

Similar design for other restrictions are coded in the same manor, exhibitor age (adult and juvenile) exhibit gender although not all have the neutral option with in the show classes.

In appendix 6 there is a very simplified set of tables

2.3. User Interface Design

The forms are at present less flexible than the author wanted them to be, for example upside down, members only and breeders options are hard coded in to the forms where appropriate these should be coded into the database itself or a configuration file.

All User interface forms are of the same basic design which the user is more or less familiar already and are easy to learn

Fig 1 in appendix 4 First opening of the Breed Editor Form this is the display what is currently seen.

Fig 2 A single click on the in this case the Angora puts the changeable data in the lower part of the form.

The same principle is used on all forms within the project. Yes the author wanted to use a JTable however it is one of the components the author had been struggling with so taking the project supervisors' advice the author created the majority of this in about three weeks, a system that doesn't look as exactly what was originally intended but works. Which is the principle of Open Source Computing and the Extreme Programming ethos get some of it working and released, improvements can be made later.

Fig 5 is the overall database design.

Breeds table and Colours table would be a many to many relation so junction table BreedColours is used. This junction table also has a second function to ensure that all possible breed colour combination for the show are at least covered. As there are two special records in Breed they are AV (Any Variety) and AOV (Any Other Variety), and similarly in Colours AC (Any Colour) and AOC (Any Other Colour). The idea behind that is the show would list some colours available for a breed such as the Dutch Breed it has 9 different colours, however most shows would not list all colours for the breed, so a breed classes for Dutch can be Dutch Black & Blue, Dutch Chocolate and for all of the other colours Dutch AOC can be used once selected it disable any further selection of Dutch Breed classes. This restriction only applies in Breed Classes.

The other junction table required is Entries Table a junction between Exhibits and ShowClasses which would be a many to many relation, this table is purely to be able two many to one relations Exhibits Entries, and ShowClasses, as an exhibit can be entered in to many show classes and a show class can have many exhibits this also ensures that no exhibit can be entered in to the same show class more than once.

3. Implementation

3.1. The database singleton.

The database access is through a singleton class. Which supposed to be difficult to use unit testing against. However I found that once the database tables became stable, the singleton design pattern was the best way to access the database. The Database code is completely portable the author the even to another database engine.

3.2. SQL (Structured Query Language)

At first the some of more the complex SQL (Structured Query Language) statements to do with the junction tables were not working as first thought they would, however with a bit of practice of refining these statements they now work as required.

The original design and implementation of the forms was totally wrong mainly due to the lack of understanding of Swing component library, even in the midpoint demo was using the original structure that really was not going to be successful on multiple forms. Since that time a complete overhaul of the system was required, now stable and a common theme is used through out.

3.3. The lists

The list structures are instances at present, and work, as only one form with lists is active at one time having them as instance variables is okay, this can however eat into large amounts of garbage collecting on the heap. It is thought that maybe the lists should also be singleton design patterned classes. There was a quick attempt at doing so, however there was too many errors encountered and a choice was to leave functioning code alone.

There is a special note to be made about the junction tables, the Java classes that are used with in the program has two values for each of the breedId and colourId the the first assignment to an instance of BreedClass copies the breed_id and colour_id both values only one is ever changed so that the individual record can be safely updated as when an update is done, the original breedcolours entry can be found and updated, it needs to be changed, there is a dilemma, an update is not going to work if original values are not used, write in it's place the new value, as the key is a complex key with no other index way of address the record, keeping the original values enables this.

3.4. Ring Numbers

The regular expression to ensure the user enters ring numbers correctly took a bit of time to workout, the first two components of a ring number were no problem, they are two digit year number 0-99 that is [0-9][0-9] the second part was also easy [A-LX] the last part is a value from 1-99999 [[0-9]4[1-9]] | [[0-9]3[1-9][0-9]] | [[0-9]2[1-9][0-9]2] [[0-9][1-9][0-9]3] | [[1-9][0-9]4] the method that checks the validity of the ring number

(“Java Regular Expression Tutorial with Examples | Java Code Geeks,” n.d.) is a static method as this method is better linked to the class rather than being linked to any or all individual instances of the class and as a result can be accessed whether or not an instance exists.

As there so much still to implement there is a not a lot more that can be discussed here.

4. Testing

4.1. Overall Approach to Testing

Originally the whole project was to be Test Driven Design, so testing was at the based of everything. This approach was working with the non graphical user interface parts, although the strict regime of creating a test getting it to fail and then pass with re-factoring the so called red/green system of development was for the most part abandoned, partly because some of the code turns out to be simple straight forward

and really doesn't warrant TDD, mainly however because of the GUI components actions are required and research picked up a few libraries to use, but lack of time to learn to use these tools, means there is little automated testing.

4.2. Automated Testing

4.2.1. Unit Tests

Some of the classes have unit test. Some of simple methods are not covered by unit tests, TDD is a good practice, it really does help to reduce the number of bugs, especially when dealing with complex algorithms, if extreme programming is taken to it limits every single getter and setter for every single instance variable, will have a unit testing. That is in the author's humble opinion a little excessive for simple getters and setters, however the more complex getters and setters for example specific range checking then a unit test is a good idea.

The unit tests for the database singleton are specific to the structure of the underlying database, as specific tables and fields are used, so the tests would need to be re-factored for another database.

4.2.2. User Interface Testing

The author would like to automate the User Interface Testing however not finding useful tools it has not been done.

4.2.3. Stress Testing

The program is not yet at a suitable point for stress testing. Once the first version is complete it is thought to be able to handle a standard Chelmsford show of roughly 100 exhibits and 25 to 75 exhibitors with a breed classes that cover the entire range of breeds with the use of AOV Any Other Variety and AOC Any Other Colour clauses in the Show Class list as a capture any that are not specified explicitly this is common

practice in all but the largest shows where there is likely to be many more breed classes created, this is very in specific breed club shows. For instance there is an East Anglia Netherland Dwarf club and they hold at least two or three shows at year can have a breed class for each of the 43 possible colours at various venues in the region held in combination with a local show. It is standard practice when entering the specialist show, to enter the local show hosting the specialist as well, even it is just a token entry.

4.2.4. Other Types of Testing

Most of the testing has been done in an interactive manor as changes have been made in the code, a suitable point was chosen to test these changes. The test would have been created as JUnit Tests or robotic testing, however create the test in that way.

4.3. Integration Testing

As the project is not currently complete, there is no other integration except interacting with the database, and that is covered within the JUint testing, that is as far as it goes at present.

4.4. User Testing

As currently the author is the only user at present there can not be significant user test without an underlying knowledge of the code.

5. Critical Evaluation

5.1. Were the requirement correctly identified?

I am sure that the requirements were specified correctly, what I want the project to achieve is still the same, to take a lot of the sheer grudge out of the process. It will enable late entries and changes to the entries almost up until the show commences.

5.2. The choice of implementation language Java?

No! Java is good programming language but the Graphical User Interface is so convoluted everything is delegated to another class until it has to be dealt with, and is so confusing. I made several attempts at learning the Java Persistence Interface, Swing components, all too confusing. Some of the components I still don't understand and it was only on the advice of my supervisor that I made changes that have made some of the project to actually work. Before that advice I had abandoned the original language and tried an Open Source version of Object Delphi named Lazarus but there were problems updating the database. Reading the database wasn't a problem. I even contemplated a text based menu driven system in Java and work was progressing, I then received that advice to use a graphical user interface not using the confusing parts. That was just before the Easter vacation. So using the text based code I have rewritten almost from scratch GUI. It is not perfect, and incomplete.

5.3. Could a more suitable set of tools have been chosen?

May be a different graphical user interface such as Java FX could have been used, however like Java in general it is confusing, to use at times. A different programming language with different interface components. C and GTK+ or C++ and QT but not knowing the interfaces could have a bad situation worse, the languages are not too much of a problem, this includes Java. Just the libraries never seem to do anything easily in my humble opinion. For the database access maybe the Java Persistence API could be used.

5.4. How well did the software meet the needs of those who were expecting to use it?

As the desktop application development is still on going, the answer to this is no, not at present. As the desktop application is the fundamentally to the whole system to create the classes, judges is the area where to create the data for the web site, and the Android application, their development hasn't been started. At present the author is the only proposed user, and will continue development during any spare time once the marks have been published.

5.5. How well were any other project aims achieved?

None of the aims have been completely achieved at present.

The desktop application is partly working there are a few bugs in at present, there is a problem when adding any colour or any other colour the colour list for a class, the appropriate item is removed from the available colour options and but not displayed in the selected colours list. The author suspects the problem is caused in the GUI and therefore at present has to tested manually.

5.6. If you were starting again, what would you do differently?

I am not sure I would use Java swing for the desktop application, the database access code in Java is good and robust and can be reused on this or any other database based application. As I believe it to be generic enough drop the singleton database access code to work with any underlying database. The unit tests would of course need to be changed as the current tests are specific to the current structure of the tables.

5.7. Things that went well!

The one thing that almost surprised the author, once the information that was required to be in the database was defined, it was almost instinctive to put it directly

in to 3rd Normal Form, the author was unsure at first, and had got a friend to check it over, and they did not have any concerns over the design. Also with reference to (Lynn Beighley)[1] Chapter 7 to ensure that the database was correct. Once the design of the database was fixed typing the original data for the tables went well, with which the initial values for a show are in place, 73 different breeds, and 138 colours defined. Most breeds are just one colour and a few have in excess of 10 different colours. There are two junction tables one dynamic for the class and exhibits many to many relation, and one static, an additional use was found and implemented for the breedcolours table the static junction table, it could specify the number of colours left for a breed as they are allocated to the list of growing show classes.

Development for the midpoint demonstration went better than I could have ever hoped, the bash script for creating the database on system with the database server on, went well, and being able to run in side the IDE and as a command line program was demonstrated.

The ability to detect the database and close the program with a reasonably understandable error message. All was easier than first though.

The enhancements that have been made have it possible to connect to a number of registered databases, currently only allowed to add in the source code only allow to access hosts on the same subnet this is of course it is for security reasons for the program and the university network set up.

5.8. Things that didn't go so well.

To be entirely honest, most of the problems with the user interface are probably caused by the developer being a perfectionist, if it is not right then don't deliver, also not understanding exactly how the classes and interfaces of the GUI components and how they blend together. Which delayed the whole desktop, emails asking at for at least hints were sent to various members of staff, and external friends for help, searching the internet was so confusing, at last one member of staff forwarded an

email I sent to them to another member of staff and a useful reply came from the second member of staff, that made the click of enlightenment, for that component, this was a couple weeks before the Easter vacation, and the author could and should have asked for more help. That wasn't done because of a bad experience in industry some years before.

To get something working there was a couple of side steps, in an attempt to use Lazarus, there is a bug in the compiled form designer with the installation of the operating system installed, so another version of the operating system was installed in a virtual machine, which also had similar problems.
So a text based menu driven interface was considered in Java. Working with the text based interface after about three hours things were beginning to come together, proving to the author that the problems were more use of graphical user interface components. There was more completed in those three hours than had been done for weeks of struggling with Swing components. However a reply to an email came from the project supervisor changed that suggesting that a working GUI application was expected, and something working is as a GUI, would be acceptable. So during the Easter Vacation the application was developed.

5.9. Room for Improvement.

There is no such thing as a finished project, this is true of almost any project with more than 10 lines of code, this project is not ready for a first release, there is considerable development work still to do, however at the end of a project there is usually relief and especially a single person project of 400 hours. For the author / programmer though really wants the project to continue, using the appropriate components rather than the current set, more at their own pace right through to completion, including the website, the Android application, once that is done then

getting a programmer that has experience to produce the iOS for the iPhone and iPad application. This has to done on Mac hardware due to licensing conditions.

5.10. Was this a good project to do and what have I learnt from doing this?

Yes it is a good project, not the best one may be, however I have learnt that the basic idea for the project is good. The biggest lesson learnt from this has been that whatever time is allocated to a task, that time will always be filled, and require more time, and a first draft of a program doesn't need to be perfect that is a hard for a perfectionist (where code is concerned anyway). To ask more questions, and share problems more with others. Which is a basic element of working within teams.

Appendices

Appendix 1 The current system for a Rabbit Show.

Obtain BRC certification and star rating for the show from the classes agreed upon on at the committee meeting.

Requires a list of all breed classes.

Breed challenge classes in the show.

The proposed the judges.

The entry fees and prize money structures.

Once confirmation from the BRC is received.

The advert for the Fur & Feather is produced and sent.

Taking entries - From the adverts appearance in the F&F there will be.

Entries arrive by telephone and posted to the show secretary, all stored in paper in a box file.

The close of entries is the Wednesday 21:30 giving two days for the major part of the paperwork to be done.

After the close of entries before the show day.

No further entries are permitted

From the entries in the box files create the master document named the secretary's book according to BRC rules, some breeds must have top tier pens

The entry fees are calculated, if the show doesn't have a fixed entry fee.

Create the Judging sheets.

Create booking in/out sheet

Booking in/out sheet lists each exhibit it's breed class and assigned pen number with the exhibitor's name. This is often done when the exhibitor has more than possible exhibit and has not decided as to which one of them to take. Using this option the exhibit can be entered in fewer classes, unless exhibits are of same gender, age group breeder status.

Booking in/out sheet lists each exhibit it's breed class and assigned pen number and if known at entry time the ring number, age and gender, of the exhibit with the exhibitor's name.

Prepare as far as possible the prize cards.

Prepare as far as possible the Challenge Certificates.

Show day

Book in own exhibits.

Book in other exhibitors live stock until another committee member arrives then delegate the booking in to the committee member.

Be at the secretary's desk for most of the rest of the show. Deal with issues such as replacement animals, class/breed/gender/breeder status change which is at the discretion of the committee.

Take entry fees during the day making a note that they have been paid.

Take results enter the these in to the secretary's master book.

Log the prize money against the exhibitors entry in the secretary's book as appropriate.

Finish the appropriate prize cards and challenge certificates.

Sales and exchanges of exhibits is permitted, the change of ownership has to be registered and the club is permitted to charge a percentage of the price, because the booking in and out sheet needs to be changed to reflect this. The ownership change is not reflected in the status of the show, the entry fees come from and prize money goes to the original exhibitor.

As the show finishes the final results are entered in to the secretary's book, and cards completed and distributed to the pens.

The booking out commences.

The exhibitors collect their prize money.

Judges are paid their requested expenses.

After the show

Prepare BRC report of results.

Prepare F&F report of results.

Prepare Club reports of results.

Appendix 2 - The improved system.

Requests for improved performance and slacken the workload for the show secretary.

The BRC still runs almost exclusively on paper documents, therefore a printed request for show status is required assistance with the preparation this document.

There is a BRC rule that sick animals are not permitted in to the show hall. This means that an exhibitor may not be able to bring the exhibit entered originally. In the paper based system it was always difficult if not impossible to change the classes that an exhibit is entered in. However a computerised system might be able to make the change easier, if not a trivial matter.

The after show reports take a day of two to sort, and as it is manual process mistakes can occur, and cause all sorts of problems at the BRC especially as their records have to be accurate the computer should be able to produce the result reports quickly and as accurately as the results were originally entered.

The prize money is added to the exhibitor's record as the results come in and when the exhibitor comes to collect the additions need to done on the spot and quickly. May be a computer will be able to help here.

Appendix 3 Third-Party Code and Libraries

The project as is at present has not made use of any third-party libraries, JUnit test, Swing which are part of the standard Netbeans development system, there is the PostgreSQL JDBC driver of course, and I have set aside a small regular expression code to check that BRC ring numbers are valid (“Java Regular Expression Tutorial with Examples | Java Code Geeks,” n.d.)

Appendix 4 Screen Shots

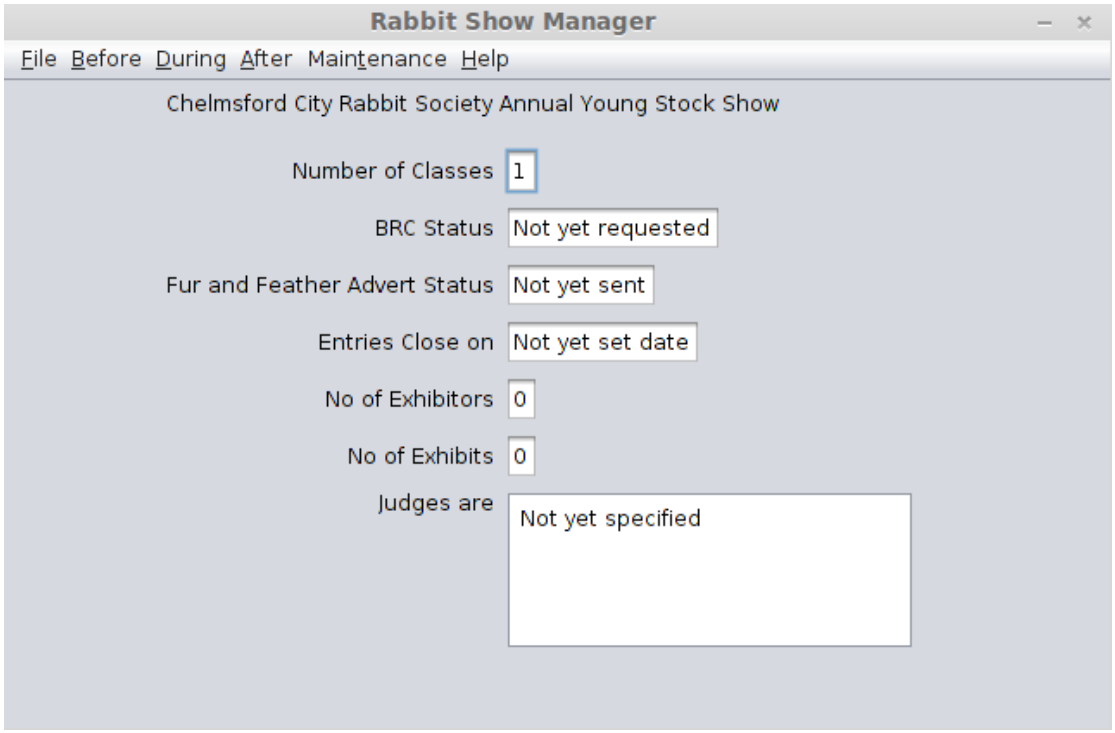


Fig 1 The opening screen

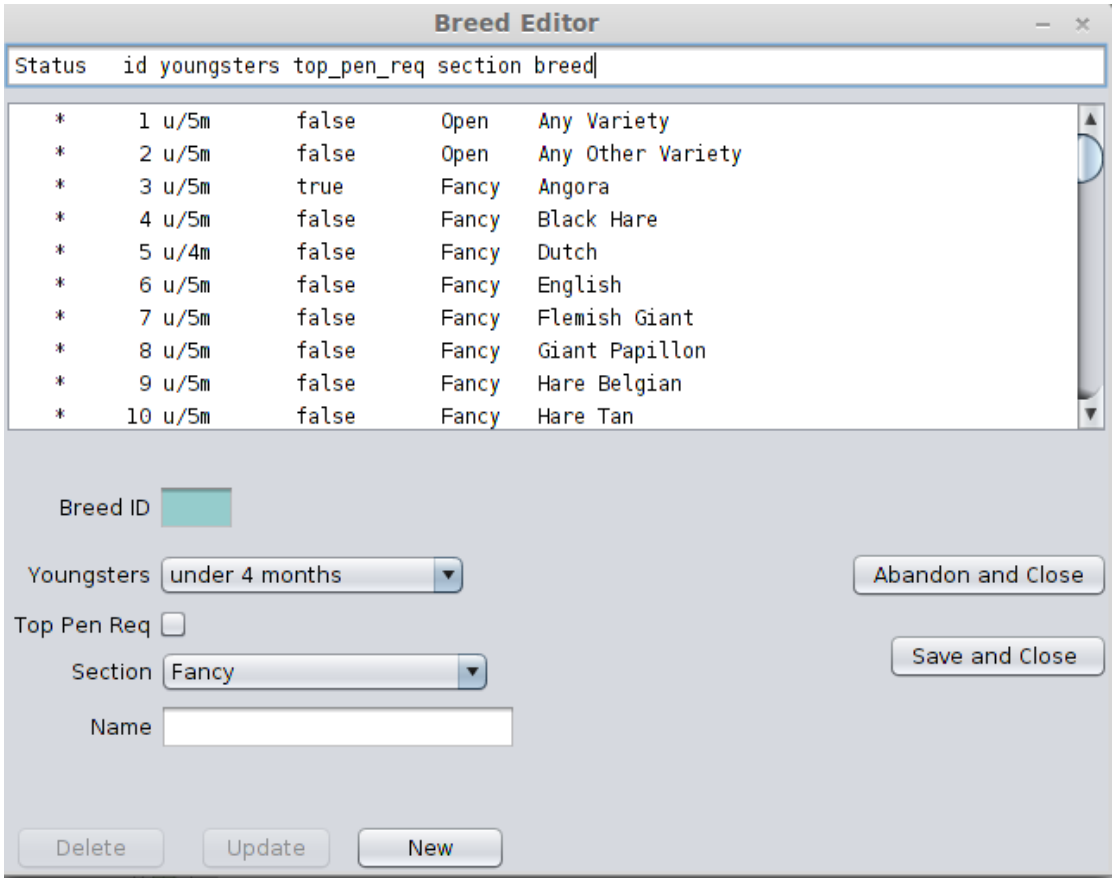


Fig 2 The Breed Editor has just been opened

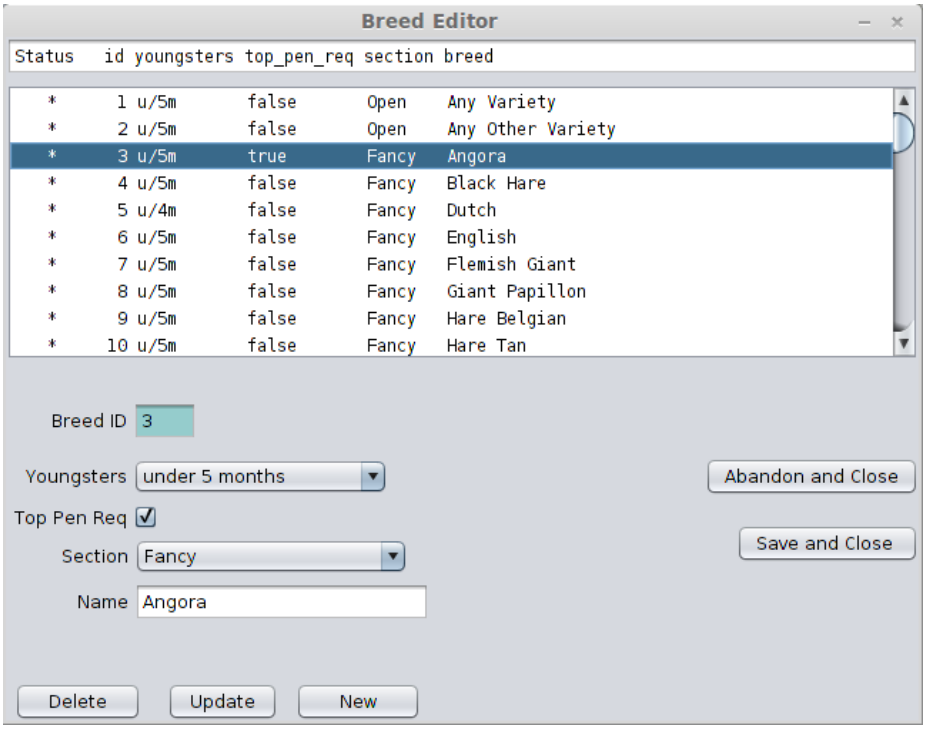
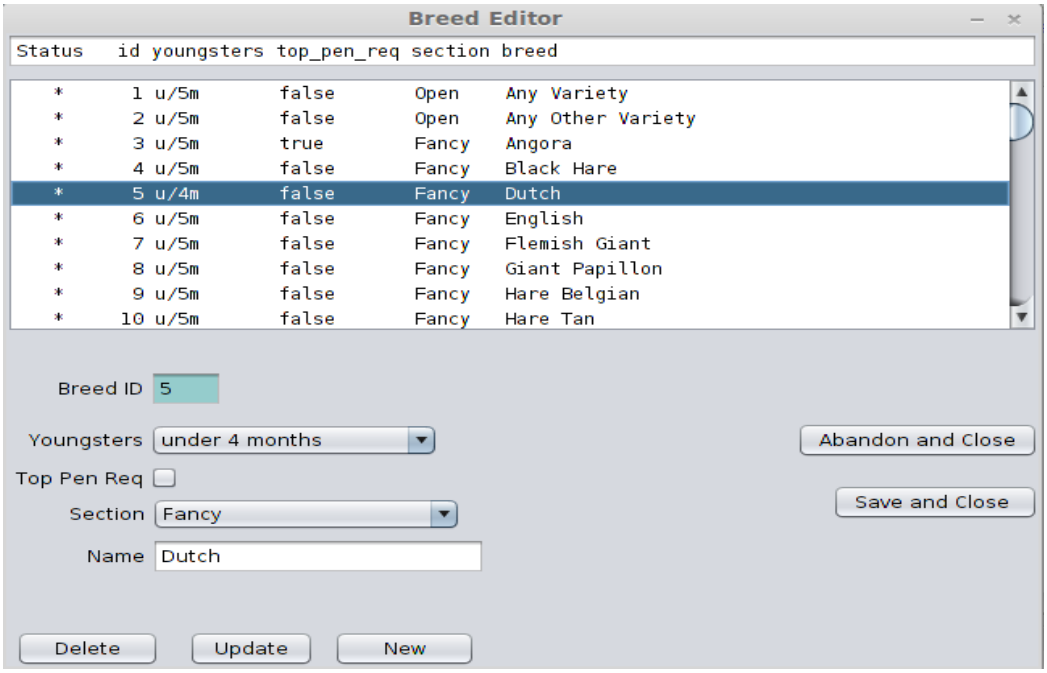


Fig 3 Click on the Angora Breed



Appendix 5 The overall database design

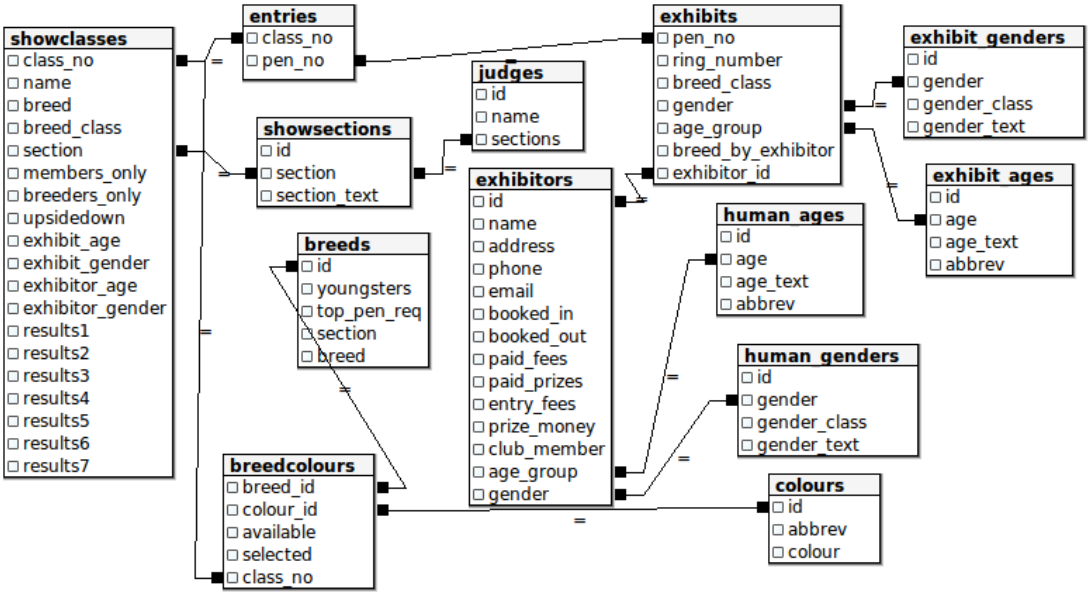


Fig 5 The Overall Database design

This is a diagram created in pgAdmin III tool graphical query builder, so table relations are not shown as they should be.

Appendix 6 Entries sample only part of the data.

Exhibitor details	Age Group	Gender	Club member
Exhibitor *1*	Adult *01*	Gentleman *01*	*true*

Exhibit	Age group	Gender	Breed by exhibitor
pen *1*	Adult *100*	Buck *01*	*false*
pen *2*	U/5m *010*	Doe *10*	*true*

Exhibitor details	Age Group	Gender	Club member
Exhibitor *2*	Juvenile *02*	Lady *02*	*false*

Exhibit	Age group	Gender	Breed by exhibitor
Rabbit pen 3	Adult *100*	Doe *10*	*false*
Rabbit pen *4*	U/14w *001*	Buck *01*	*true*

Class Name	Exhibitor Age	Exhibit Gender	Breeders Only
500 AV AA Does	Open *11*	Does 10	*false*
501 AV AA Bucks	Open *11*	Bucks *01*	*false*
502 AV AA Juvenile	Juvenile *10*	Open *11*	*false*
503 AV AA Breeders	Open *11*	Open *11*	*true*

Entries Class	Entries Pen
500	2
500	3
501	1
501	4
502	3
502	4
503	2
503	4

Key	
italics	Binary values
AA	Any Age
AC	Any Colour
AV	Any Variety
U/14w	Under 14 weeks
U/5m	Under 5 months
Pen	Cage for an Exhibit

Some paper forms that are used.

A generic entry form

Breed Class	**Breed**	**Ring Number**	**Duplicate Classes.**
5	ND BEW AA	12X 00124	10, 101, 102, 105, 120

Name	Paul Massey
Address	120 Lawn Lane Anytown Acounty PC1 2TG
Phone	07984-123456
Email	anemail@anaccount.com

A judging sheet for a completed class, the greyed area is retained by the Judge and Steward the white area is detached to record the result of the judging of the class. The type set is what is printed on the sheet before it is distributed to the Judge and Steward, the script is added by the judge's steward.

4 Places		Class 5: Netherland Dwarf Blue Eyed White Any Age	C5 P1 of 1	
1	3rd		1	3rd
2			2	
5	1st	Possible Best In Show 12X00124	5	1st
7	RES		7	RES
11			11	
18	2nd		18	2nd
20			20	

Appendix 7 Code Samples

There was a problem in calling type specific overridden methods that appeared not to work with this small gem of code it is used in the abstract class BaseDataList twice once for readList method and once for the writeList method, these two methods are also called when re-reading from the database if an update is to be abandoned. This resynchronises the record and the instance of the record in memory.

```
public void readList(HeaderRequired hr, Object dataItem, String table, String with,
                                                                    String orderBy ){

       ResultSet rs = DBA.executeSQL(String.format(
            "SELECT * FROM %s %s %s",table,
                with != null?String.format("WITH %s",with):" ",
                orderBy != null?String.format("ORDER BY %s", orderBy):" "));
       try {
            if (hr == HeaderRequired.HEADERS){
                header = DBA.getHeader(rs);
                formatString = DBA.getFormatFromHeader(header);
                header = header.replace(';', ' ');
            }
            while (rs.next()){
                BaseDataItem x;
                if (dataItem instanceof Breed){
                    x = new Breed();
                    x.getData(rs);
                    dataItem = x;
                    //list.add(dataItem);
                } else if (dataItem instanceof Colour){
                    x = new Colour();
                    x.getData(rs);
                    dataItem = x;
                } else if (dataItem instanceof BreedColour){
                    x = new BreedColour();
                    x.getData(rs);
                … /* there rest of the class types follow */
                }
                list.add(dataItem);
            }
       } catch (SQLException ex) {
            Logger.getLogger(BreedList.class.getName()).log(Level.SEVERE, null, ex);
       }
  }

 public void writeList(BaseDataList theList, BaseDataItem paramDataItem){
       //BaseDataList dataList;
       BaseDataItem localDataItem = null;
       // locate the currect polymorph
       for (Iterator it = list.iterator(); it.hasNext();) {
            if (paramDataItem instanceof Breed){
                 localDataItem  = (Breed)it.next();
            } else if ( paramDataItem instanceof Colour){
                 localDataItem  = (Colour) it.next();
            } else if (paramDataItem instanceof BreedColour){
                 localDataItem  = (BreedColour) it.next();
            else if … /*the other classes follow */
       }
            // found the correct polymorph now perform the process it
            if (localDataItem.isDirty()){
                localDataItem.performUpdate();
            } else if (localDataItem.isReadyToDelete()) {
                // delete will also delete any constrained records.
                localDataItem.performDelete();
            } else if (localDataItem.isNewItem() &&
                       localDataItem.isCancelledNewItem()){
                localDataItem.performInsert();
                localDataItem.setNewItem(false);
            }
       }
  }
```

Annotated Bibliography

[1] Beighley, L., 2007. Head First SQL, First. ed. O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Read chapter 7 to confirm the database was in 3rd normal form

[2] Java Regular Expression Tutorial with Examples | Java Code Geeks [WWW Document], n.d. URL http://www.javacodegeeks.com/2012/11/java-regular-expression-tutorial-with-examples.html (accessed 5.2.14).

Helped with the creation of the regular expression for checking the ring number entered by the user

[3] Java Tutorial [WWW Document], n.d. URL http://www.learn-java-tutorial.com/inheritance.cfm (accessed 5.3.14).

Used to ensure the method from a super class uses the correct class to get or set records to and from the database.

[4] Laughing Rabbit Software and Systems [WWW Document], n.d. URL http://laughingrabbit.com/ (accessed 5.2.14).

One of the competing applications.

[5] Make your own DEB and RPM packages | Linux User & Developer - the Linux and FOSS mag for a GNU generation [WWW Document], n.d. URL http://www.linuxuser.co.uk/tutorials/make-your-own-deb-and-rpm-packages (accessed 5.2.14).

Read early on to create an installation package this has not yet done.

[6] PostgreSQL: The world's most advanced open source database [WWW Document], n.d. URL http://www.postgresql.org/ (accessed 5.3.14).

Used to help with developing some of the more complex SQL statements

[7] PostgreSQL Tutorial [WWW Document], n.d. URL http://www.tutorialspoint.com/postgresql/ (accessed 5.3.14).

Used to help with developing some of the more complex SQL statements

[8] Show Beacon [WWW Document], n.d. URL http://www.theshowbeacon.com/ (accessed 5.2.14).

Second of the competing products.

[9] SHOW SECRETARIAL SOFTWARE [WWW Document], n.d. URL http://showprogram.com/ (accessed 5.2.14).

Third competing product

[10] SQLite Tutorial [WWW Document], n.d. URL http://www.tutorialspoint.com/sqlite/ (accessed 5.3.14).

read to prepare the database using SQLite3 for the Android Application

[11] The BRC - Welcome to the Official website of The British Rabbit Council [WWW Document], n.d. URL http://www.thebrc.org/ (accessed 5.2.14).

The information to create breed and colours tables and some of the business rules that are needed for the application.

[12] The Show Program [WWW Document], n.d. Gd. Champion Softw. Show Program. URL http://www.grandchampionsoftware.com/ShowProgram/ (accessed 5.2.14).

The last of the competing products.