Programming Applications (PRA)
Major Piece of Coursework (Group Project)
"We're going to need a bigger GUI" (15%, 100 marks)

If you have any questions about this assignment, please follow the steps listed in the 'Additional Support' section on KEATS.

Remember, we will be using the TeamFeedback system to elicit feedback from each of your fellow group members about your contribution to the group project. If significantly negative, this feedback will directly affect your grade in this assignment (15% of your PRA grade). Be sure to make a good impression on those students in your group throughout the assignment.

1 Aims

The aim of this assignment is to provide practice in:

- Developing user-friendly interfaces;
- Building self-contained applications with widgets, layout managers and event handlers;
- Using a version control system for managing source code;
- Interpreting application requirements that go beyond a Computer Science technical level and relate to application domain user needs;
- Identifying difficulties in GUI design;

- Working with pre-existing code, provided as a library;
- Performing task analysis and UI design;
- Documenting code.

2 Deadline

1. The deadline for completing Requirement 0 is Sunday 14th February, at 11.55pm.
2. The deadline for completing the remaining requirements is Saturday 26th March, at 11.55pm.

3 Domain Description

The [not-for-profit organisation] ***OCEARCH*** *has been busy tagging great white sharks and tracking their movements in near real time, using new technology to advance our understanding of these wondrous sea creatures.*

OCEARCH works with 20 different institutions and it even has its own at-sea laboratory. "[Utilising] a custom 75,000 lb.-capacity hydraulic platform designed to safely lift mature sharks for access by a multi-disciplined research team, up to 12 studies are conducted in approximately 15 minutes on a live mature shark," the [organisation] says of its M/V OCEARCH vessel. The data collected from these expeditions aids in conservation efforts for great white sharks as well as promoting a better understanding of these apex predators and their role in the ecosystem.

The research – such as tagging and blood sample collection – means only minimal discomfort for the sharks, and there's been no evidence of long-term problems or pain, according to OCEARCH (Source: **International Business Times**).

In this assignment, you will develop an application to work with this shark tracking data.

4 Requirements Overview

The application that you build for this assignment should exhibit the following functionality:

- The software should be able to connect to a server (using a provided API) in order to access shark tracking data.
- There should be a function to search for sharks that have appeared within a given time frame (in the last 24 hours, in the last week or in the last month).
- It should be possible to filter the search for sharks by gender, stage of life and tag location.
- The results of a search should display all pertinent shark details, including species, length, weight and a description of the shark. These results should be ordered by recency.
- It should be possible to 'follow' a shark such that they appear in a list of favourites. Clicking an entry in this list should display more information about that shark.
- Favourite sharks should be ordered by the shark's distance to our current location at King's (because everyone likes to know how close they are to their favourite sharks).
- It should be possible to unfollow a shark, if need be.
- Users should be informed if they have a *Sharknado* anomaly in their favourites list.
- It should be possible to arbitrarily switch between the search tool and the favourites list.
- Your application should permit the use of different user profiles.
- Your application should present a 'Shark of the day' feature.
- Your application should show visual statistics about the types of shark present in a given time period.
- Your application should display a map that depicts the geographical location of all followed sharks.

5 The Jaws API

The term API (Application Programming Interface) has developed a reasonably flexible definition in recent years. In general though, an API provides you with the *resources* to develop an application, your application 'plugs in' to these resources, and they become a fundamental part of its operation. Most typically you will be familiar with the Java API. In this API, the supplied resource is the Java library classes (e.g. Scanner, ArrayList etc.), which you use to produce your programs. When you import classes from this library you are plugging in to them, and the operation of your program relies on their presence.

The resources provided by an API don't have to be purely function based. Instead, an API can provide you with *data* as a resource. A good example of this is the **Twitter API**, which provides you with data from Twitter that can be used as a resource to build applications. Typically APIs of this type also provide you with the means to manipulate the data provided, at source. Unlike the Java library, which can be accessed locally, APIs like the one provided by Twitter, are accessed remotely over the Internet using a network connection.

The *Jaws* API provides a remote data resource for the shark tracking information mentioned in Section 3. In this piece of coursework, you will interact with the Jaws API via a local proxy (in the non-technical sense of the word) which takes the form of a custom Java library. As we know, custom Java libraries contain additional classes that do not exist in the standard Java library. In order to call the methods contained in a custom library, the associated jar file (in this case *jaws.jar*) must added to the build path of a project (or the classpath, when compiling from the terminal) so that classes from that library can be imported. The classes provided by the Jaws API Java library (the Jaws library) encapsulate calls to the Jaws API allowing you to obtain remote data via local method invocations. Thus, in this coursework, you will use an amalgam of a local API and a remote API.

Like all remote APIs, use of the Jaws API, and thus the Jaws library, is subject to the constraints of network communication. You may therefore experience latency issues when accessing the Jaws API, but this is normal. If the Jaws API goes offline for any reason, details will be posted here:

`http://www.dcs.kcl.ac.uk/staff/martin/jaws/`

In addition, like all software, the Jaws library will be subject to revisions throughout the lifetime of the major project. Keep an eye on your repository for new versions.

5.1 Connecting to and testing the Jaws API

Your first preliminary task should be to ensure that you can use the Jaws library to communicate with the Jaws API. You will find the Jaws library jar file in your repository. Download the file, and include it in the build path of a project, or at save it in an accessible location locally if you intend to compile and run from the command line. You will need to conduct some additional research to find out how to do this from your chosen IDE or, if you so choose, how to compile and run a program written to work with a custom library from the command line.

Once added, import the library as follows:

```
import api.jaws.Jaws;
```

You can then make a `Jaws` object, and interact with it in order to experiment with various method calls. In order to make a `Jaws` object, you will need to pass both your group's public and private keys (provided as a cover page to this document). In addition, you will need to specify whether you want to connect to the API over a secure connection. Secure connections are recommended, but if you experience slow communication with the remote API, you can construct a `Jaws` object with the secure connection parameter set to false, or without a secure connection parameter at all, in order to connect to the API normally. All these details are confirmed in the Jaws library *documentation*, which is also available in your repository. You should use this documentation from now on to understand how to use the Jaws library. If you are using an IDE, it is recommended that you associate this documentation with the Jaws library jar in your build path, so that you can view the documentation in the relevant tooltips.

The first method I recommend you call is the simplest: `getLastUpdated`. This method will tell you when the last set of shark tracking data was received. Print the result of calling this method in order to confirm your connection with the API.

5.2 Access in the Informatics Labs

If you are having trouble using the Jaws library to access the remote Jaws API in the Informatics labs, then you may have to configure your IDE to use a HTTP proxy. Conduct the appropriate research in order to learn how to do this, and share your experiences with your peers via the PRA KEATS discussion forum. The common proxy settings required are proxy.inf.kcl.ac.uk as the host, and 3128 as the port.

6 Requirement 0 (5 marks, 5%)

You will find details of the tasks associated with this requirement on KEATS.

7 Remaining requirements (95 marks, 95%)

Elaborating on Section 4, the remainder of this document details the features that your application should provide.

7.1 Searching

When the application is loaded, a window should appear similar to the one show in Figure 1. This will be referred to as the menu frame in the remainder of this document.

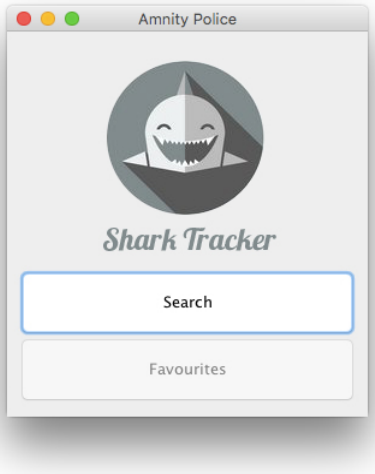

Figure 1: The menu frame.

There should be two buttons on the menu frame that read ‘Search’ and ‘Favourites’. Select your own graphic and use it in the place of the one currently shown on the menu frame.

When no sharks are being followed (for example when you first run the program), the favourites button should be disabled. When this window is closed, the entire application should exit. Give this frame a suitable title.

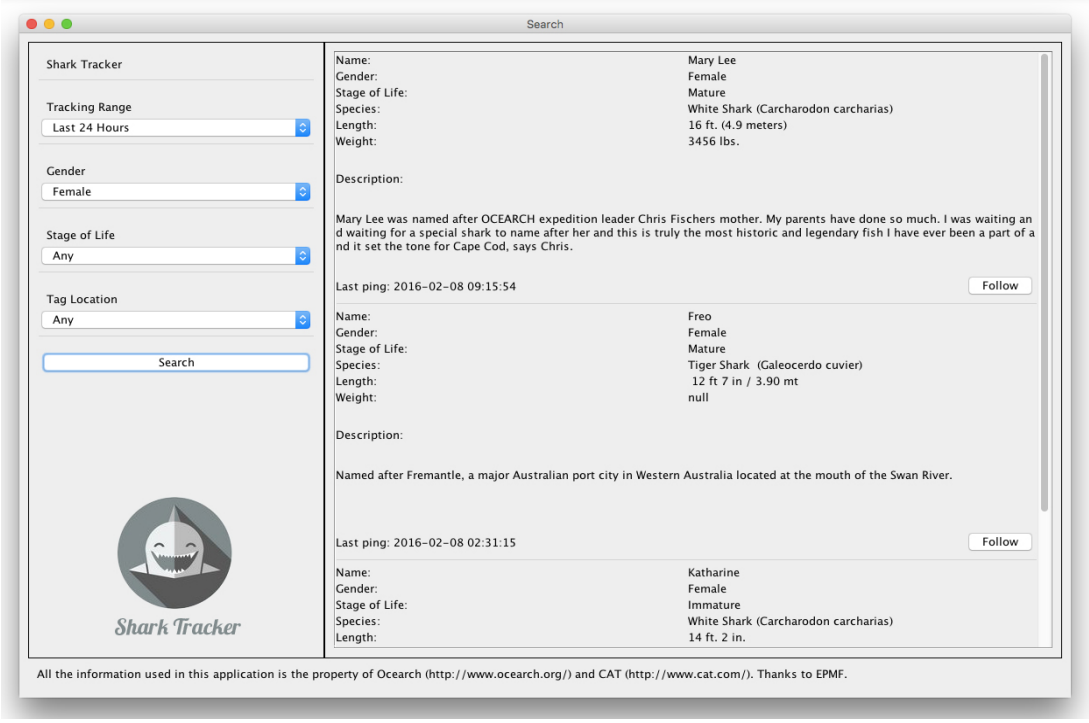

Figure 2: The search frame (excluding additions made for the advanced requirements (Section 7.3)).

7.1.1 Setting up the search frame

Pressing the 'Search' button on the menu frame should display a window similar to the one shown in Figure 2. To ensure you are fully comfortable with layout managers, you should replicate the layout of this window as closely as possible. This window will contain the main complexity associated with the application, and will be referred to as the search frame in the remainder of this document.

When the search frame is first opened, the right hand side of it, which holds the search results, should be empty. This area of the frame will be referred to as the search results pane in the remainder of this document. The left hand side of the frame should contain

various drop-down option boxes (option sets) that can be used to refine the search results for different sharks (similar functionality can be seen on the OCEARCH website), which are shown when the user presses 'Search' (more on this in the following section). This area of the frame will be referred to as the options pane in the remainder of this document. The option boxes, and their associated categories, in the options pane should be as follows: Tracking Range; Last 24 Hours, Last Week or Last Month; Gender; Male or Female; Stage of Life; Mature, Immature or Undetermined. The list of options for Tag Location can be obtained from the API. In addition, each drop-down option box (except Tracking Range) should have a further option 'All', which can be selected if a user does not wish to refine the results of a search in a particular category.

Most importantly, this window **must** show the acknowledgement text at the bottom of the screen, which is available using the API. **Students who do not display this acknowledgement will receive zero, as a group, for their submission**. You should once again substitute the image shown in the bottom left of the frame with a suitable image of your own. When implementing the advanced requirements, you will have to add additional components to this frame, which are not shown in Figure 2.

7.1.2 Displaying tracking data

When the 'Search' button is pressed, the search results pane should be populated with the details of those sharks that match all of the criteria selected in the options pane. In the search frame shown in Figure 2, a search has just been performed, and the only sharks in the results pane are those that have had their coordinates tracked in the past 24 hours (not longer ago) and those that are female, because the options in the options pane have been set as such. No preference is shown towards the stage of life, or the tag location.

Note: Depending on when the dataset was last updated, and then frequency with which sharks surface, there may be no shark appearances within the last 24 hours. This is normal.

The details displayed for each shark should be exactly as shown in Figure 2: Name, Gender, Stage of Life, Species, Length, Weight and Description. An area of the frame that shows information about one particular shark will be referred to as the details for that shark in the remainder of this document. In addition to the information already specified, it should be possible to see the last time tracking data was received on a particular shark (i.e. the last ping) within the details relating to that shark, if shown in the results pane.

The details shown in the search results pane should be ordered by ping time, so that the

details for a shark that matches the criteria specified in the options pane, and has been tracked most recently, appear first, and then the details for the next recent shark matching the criteria, and so on. It is essential that, for each shark matching the specified criteria, only their most recent ping is shown, such that there is never more than one entry for a shark in the results pane. The series of sub-tables in Table 1 shows how a set of shark results must be refined and ordered in order to implement this requirement.

Name	**Gender**	**Ping Time**
Alice	Female	11.00am, Monday 8th February
Alice	Female	1.00pm, Monday 8th Februrary
Einstein	Male	12.00pm, Monday 8th February
Lonesome Jorgita	Female	10.00am, Monday 8th February
	↓	
Name	**Gender**	**Ping Time**
Alice	Female	11.00am, Monday 8th February
Alice	Female	1.00pm, Monday 8th February
Lonesome Jorgita	Female	10.00am, Monday 8th February
	↓	
Name	**Gender**	**Ping Time**
Alice	Female	1.00pm, Monday 8th February
Alice	Female	11.00am, Monday 8th February
Lonesome Jorgita	Female	10.00am, Monday 8th February
	↓	
Name	**Gender**	**Ping Time**
Alice	Female	1.00pm, Monday 8th Februrary
Lonesome Jorgita	Female	10.00am, Monday 8th February

Table 1: The refinement of a set of results received from the Jaws API, to a search naturally phrased as "Show me all female sharks that have been tracked in the past 24 hours ", made at 2.00pm on Monday 8th February. In the first iteration we remove the sharks that don't match the specified criteria in the initial result from the API: Einstein, the male shark. Then, we order by recency, before finally removing the duplicate, later, entry for Alice, even though it falls after the entry for Lonesome Jorgita.

Note: The times received from the Jaws API are not necessarily in GMT, and thus may appear to be 'in the future'. This is normal.

If there are no sharks matching the given criteria, the result area should say something appropriate, like 'No results'. If multiple searches are performed, the last set of results should be cleared from the search results pane before the results of the current search are displayed.

7.2 Favourites

Once a shark has appeared as part of the tracking results, the user should have an option to follow a shark, by clicking a button marked 'Follow' within the relevant set of shark details (Figure 2). Once clicked, this button should change to read 'Following', if the user was not previously following that shark.

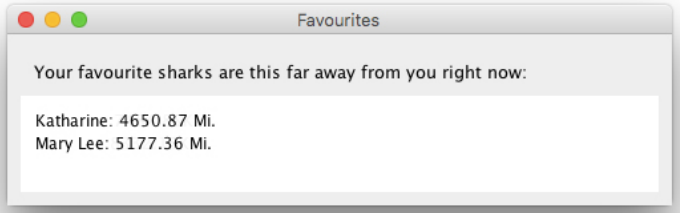

Figure 3: The favourites frame (excluding additions made for the advanced requirements (Section 7.3)).

When a shark is being followed, it should be shown in a list of favourite sharks in a new window, which can be reached from the initial menu by clicking 'Favourites' (where the Favourites button should, naturally, be enabled, when any sharks are being followed). An example of what this window might look like is shown in Figure 3. This window will referred to as the favourites frame in the remainder of this document.

7.2.1 Setting up the favourites frame

The favourites frame should have an appropriate title, and a heading that introduces the content of the frame In Figure 3 this heading reads 'Your favourite sharks are this far away from you right now'. This frame should also show an interactive list of a user's favourite sharks. This will be referred to as the favourites list in the remainder of this document. When implementing the advanced requirements, you will have to add components to this frame, which are not shown in Figure 3.

7.2.2 Closest to King's

The sharks in the favourites list should be ordered according to their distance from King's College London, where the closest shark is at the top of the list, and the shark furthest away is at the bottom. You can find location data about King's, which is interoperable with the location data that is received from the Jaws API, online. The unit of distance you choose to show is up to you. In Figure 3, I have chosen miles.

When clicking a shark in the favourites list, the favourites frame should be hidden (and the search frame shown, if it isn't already), and the current contents of the results pane on the search frame should be replaced with a single set of shark details, that relate to the shark that was just selected in the favourites list, along with this shark's most recent ping (i.e. the same information should be shown as would be if this shark was displayed in the search results pane as the result of a normal search).

7.2.3 Don't forget my favourites

Once your program terminates, it is essential that a record is kept of the sharks that have been followed during that execution of the program. When the program is run again, therefore, the favourites list in the favourites frame should be populated with all the sharks that the user has followed during their use of the program (i.e. selected as favourites), and a button in the search results pane should read 'Following' (as it would have done prior to the last termination of the program) if it appears within the shark details relating to a favourite shark, when subsequent searches are performed.

If a user wishes to unfollow a shark, clicking a button marked 'Following', within a set of shark details, should change this back to say 'Follow', and the record of the shark to which the shark details pertain, as a favourite shark, should be removed from within the program (e.g. removed from the favourites list) and from any records held independently from the program. As mentioned, if a user loads the program with no favourite sharks, thus rendering the 'Favourites' button on the menu frame disabled, this button should be enabled when the first favourite shark is added. In addition to this, if a users unfollows all their favourite sharks, such that the content of the favourites list is empty, then this button should once again be disabled.

7.2.4 Sharknado

Every so often, due to a quirk in the data (and due to some 'fun' locations thrown in by me to the data stream), a shark's location will not be in the middle of the sea, but instead over land. I will refer to this as a 'Sharknado event' in the remainder of this document[1]. When viewing the favourites list, a user should be informed if any of their favourite sharks are currently over land, and thus involved in a Sharknado event. You should register and use an appropriate, additional API to determine if a Sharknado event has occurred for each favourite shark (e.g. `https://console.developers.google.com/apis/`).

7.3 Advanced Requirements

Each member of your group should take responsibility for one of the following advanced tasks, and co-ordinate the other group members in its completion:

1. Add a menu bar to your program that facilitates the loading of different sets of favourite sharks for different users. We might refer to this as a user profile. When storing a new user profile, you should prompt the user to enter their name, and use this to index each user profile. Locating and loading a user profile should be done by asking a user for their name. If you choose to fulfil this requirement, a default user profile should be used for the existing functionality (i.e. loading and storing favourites automatically) if no user profile is loaded. It will also be necessary to perform the appropriate validation, including handling the case in which a user tries to create a user profile with the same name as a profile that already exists, or the case in which a user tries to load a profile that does not exist.

2. Add, to your search frame, in an appropriate location, a 'Shark of the day' feature, which shows the name of a random shark, accompanied by a link to a video about that shark (obtained via the Jaws API). This feature should adhere to its name strictly, and show the same shark every time your application is loaded within a 24 hour period, and then a new shark once that 24 hour period has passed.

3. Add an additional button, below 'Search', called 'Statistics'. When the statistics button is pressed, a new frame should be loaded. This frame should be divided into three different sections. Each of these sections will be referred to as a graph pane in the remainder of this document. Each graph pane should hold a pie chart (`http://www.jfree.org/jfreechart/`), that indicates, for each drop-down option

[1] `http://www.imdb.com/title/tt2724064/`

box in the options pane, the number of sharks, within the selected tracking range, that match each possible category in that option box. For example, if, when a tracking range of 'Last 24 hours' is selected, there are 5 (unique) female sharks and 5 (unique) male sharks returned, one of the graph panes, appropriate labelled with 'Gender', should present a pie chart that shows an equal distribution between female and male sharks. The term unique here is used to emphasise that the rule about duplicate entries (explained in Section 7.1.2) also applies here, such that each shark only contributes to the relevant portion of the pie chart once, even if they appear multiple times during the specified tracking range. The other two graph panes should show similar information for stage of life and tag location. If there are no sharks that match a certain category within the specified tracking range, then that portion of the pie chart related to the relevant option set should not appear. If only one category is matched during the specified tracking range, then the pie chart shown for that option set should reflect this accordingly. If, because of a lack of relevant data, the distribution for a given option set cannot be known, then a relevant message in the graph pane should be shown in place of the pie chart.

4. Recreate the shark map shown on the OCEARCH website in your application. This map should show the locations of all the user's favourite sharks on a flat image of the Earth, and should be accessible via an additional button added to the favourites frame.

7.3.1 Surprise me

Make your program do something additional, and interesting, that it doesn't do already. Marks for this section will be awarded liberally, so be as creative or reserved as you like, but make sure your functional addition is evident. Only one functional addition is required per group.

7.4 Project report

This requirement is to be completed individually, not as a group

In this requirement you are asked to think of scenarios in which your application will be used and how it should be extended in order to be of better help potential users. You are not asked to actually implement anything. Rather you are asked to write a report about your extensions (maximum 6 pages). Your report should be written based on an analysis of the tasks users need to perform (through a *hierarchical task analysis*) and the

related domain concepts (through a *domain analysis*). Your report should include models of *virtual windows* and the *global navigation structure* of your extended application. Decisions or assumptions made in analysis and design need to be clearly identified. Details of all these design techniques will be given during the PRA lectures.

7.5 Code quality

The purpose of each public class, public method, and public field must be explained with Javadoc documentation. Inline comments must be included where appropriate to explain the logic of your code.

Your application must be well-structured, aiming to avoid duplication of similar code by encapsulating it in appropriate methods or classes and using anonymous classes where appropraite. Use loops rather than repeating code, and so on. Minimising calls to the API by caching information at all appropriate points will be rewarded.

8 Assessment

Details of the assessment procedure for the major piece of coursework, including a full mark scheme and submission instructions, will be push to your repository.