CST2110 Individual Programming Assignment #2

Deadline for submission 16:00, Friday 3rd April, 2020

Please read all the assignment specification carefully first, before asking your tutor any questions.

General information

You are required to submit your work via the dedicated Unihub assignment link in the 'week 24' folder by the specified deadline. This link will 'timeout' at the submission deadline. Your work will not be accepted as an email attachment if you miss this deadline. Therefore, you are strongly advised to allow plenty of time to upload your work prior to the deadline.

Submission should comprise a single 'ZIP' file. This file should contain a separate, cleaned[1], NetBeans project for each of the three tasks described below. The work will be compiled and run in a Windows environment, so it is strongly advised that you test your work in the University labs prior to cleaning and submission.

That is to say, when the ZIP file is extracted, there should be three folders representing three independent NetBeans projects as illustrated by Figure 1 below.

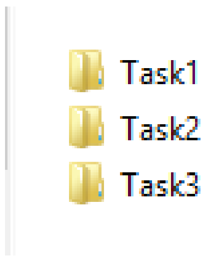


Figure 1: When the ZIP file is extracted there should be three folders named Task1, Task2 and Task3

Accordingly, when loaded into NetBeans, each task must be a separate project as illustrated by Figure 2 below.

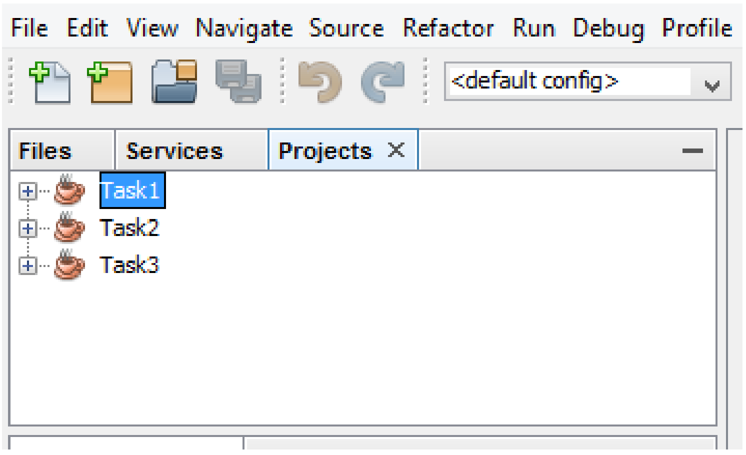


Figure 2: Each task must be a separate NetBeans Project

To make this easier, a template NetBeans project structure is provided for you to download.

[1] In the NetBeans project navigator window, right-click on the project and select 'clean' from the drop-down menu. This will remove .class files and reduce the project file size for submission.

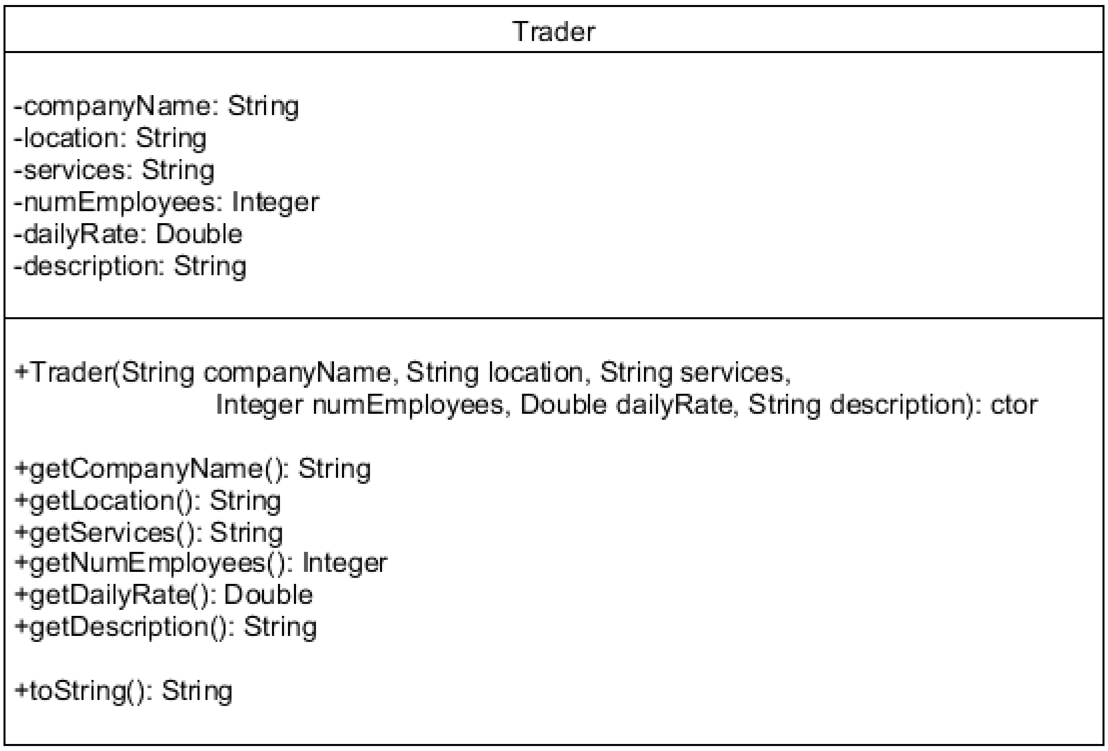
<u>Task 1 (20 marks)</u>

Create a NetBeans 8 project for this task, named *Task1*.

You are required to write a Java 8 program that opens and reads a delimited data file that is located relative to the NetBeans project root folder. The delimited data file contains fictional information about trading companies in North London. The data file is called *traders.txt*. The data file should not be altered and should be considered as a *read-only* data file.

Within the 'traders' data file, there are 25 lines each representing a single trading company. On each line there are six data fields representing the following information (in order): company name, location (in North London), the services offered by the company, the number of employees, a daily rate (pounds), and a general description of the trading company.

You are required to implement a Java class to represent a trading company. The program should parse the data file, create an object for each trading company, and store all the objects into a suitable collection. Figure 3 provides a UML class representation of the class that you will need to implement. It indicates the six data members (denoted as private data), a series of public accessor (i.e., *getter*) methods that map to the data members, a single public constructor, and a *toString()* method.



```
Trader

-companyName: String
-location: String
-services: String
-numEmployees: Integer
-dailyRate: Double
-description: String

+Trader(String companyName, String location, String services,
                 Integer numEmployees, Double dailyRate, String description): ctor

+getCompanyName(): String
+getLocation(): String
+getServices(): String
+getNumEmployees(): Integer
+getDailyRate(): Double
+getDescription(): String

+toString(): String
```

Figure 3: UML class specification for *Trader* class

Once all the objects are loaded into the collection, the program should present the User with a console-based menu to interact with the data set. This menu should loop until the User enters a character to exit the menu (e.g., zero as illustrated below). In addition to an exit option, the menu should offer <u>four</u> other options: list all traders, select a single trader, search on company location, and search for listed services.

On starting the program, the following menu should be displayed to the console:

```
List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:>
```

The User can simply exit the program by entering zero. The four other menu options allow the User to inspect the information in the data set (note again that this program is entirely *read-only* and there is no requirement to add, update or delete any part of the data set). The necessary interaction of the program with respect to these four options is illustrated in Appendix A.

Note that console output should be neatly formatted, and some consideration will be given to formatting when the program is assessed. In particular, when the option to view a single trade company details is selected, it should result in the invocation of the *toString()* method for that particular *Trader* object. You are encouraged to explore and utilise a *StringBuilder* object when implementing the *toString()* method for the *Trader* class.

Task 2 (25 marks)

Create a new NetBeans project, called Task2.

For this task you are required to write a Java 8 program that incorporates a series of classes that represent the core logic of an application and provides a NetBeans 8 console user interface. The required Java application relates to a proposed software system to manage some aspects of a gymnastics competition. Below is an initial description of the system domain with a few very simple use cases (i.e., just simple, natural language statements).

System domain

A gymnastics club holds an annual competition that includes events for both male and female gymnasts. For male gymnasts, there are six available events: Floor Exercise, Pommel Horse, Still Rings, Vault, Parallel Bars, and High Bar. For female gymnasts there are four possible events: Vault, Uneven Bars, Balance Beam, and Floor Exercise. Each gymnastics event has a specified start time and there is, for each event, a limit of 10 on the number of competitors who can compete in that event.

Each gymnastics event is organised by one or two club committee members, with each committee member organising at least one gymnastics event. Each competitor is allocated a unique competitor's number. Both male and female gymnasts may compete in a minimum of two events and a maximum of four events.

When competing in an event, the competitor is awarded a score that is calculated with respect to the difficulty level of the competitor's chosen routine. Each competitor is allowed only one attempt at a selected event, apart from the Vault for which there are two attempts. The two attempts at the Vault must be different routines.

Use cases

1. **List Event Information**. The user of the software system identifies a gymnastics event. If the event is either the Vault or Floor Exercise, then the user also identifies if it is a male or female event. In response, the system displays the number and name of every competitor in that event, along with the name of the Committee Member(s) that are organising that event.
2. **List Competitor Events.** The user of the software system identifies a competitor. In response, the system displays a list of all events that the competitor is entered in, and for each of those entered events, the system lists the routine description and difficulty rating for the attempt at that event.
3. **List Event's Attempts.** The user of the software system identifies a gymnastics event. If the event is either the Vault or Floor Exercise, then the user also identifies if it is a male or female event. In response, the system displays details of all attempts that have been completed for that event, including the name of the competitor, the routine description and difficulty rating, and the score that was achieved by the competitor for that attempt at the event if the attempt has already been completed.

Your task is to design a Java 8 program for the gymnastics event management system described above, which provides a console-based (text I/O) user interface that facilitates the three listed use cases. An initial analysis of the domain has already been partially completed, and a ***partial*** class model has been designed as shown in Figure 4.

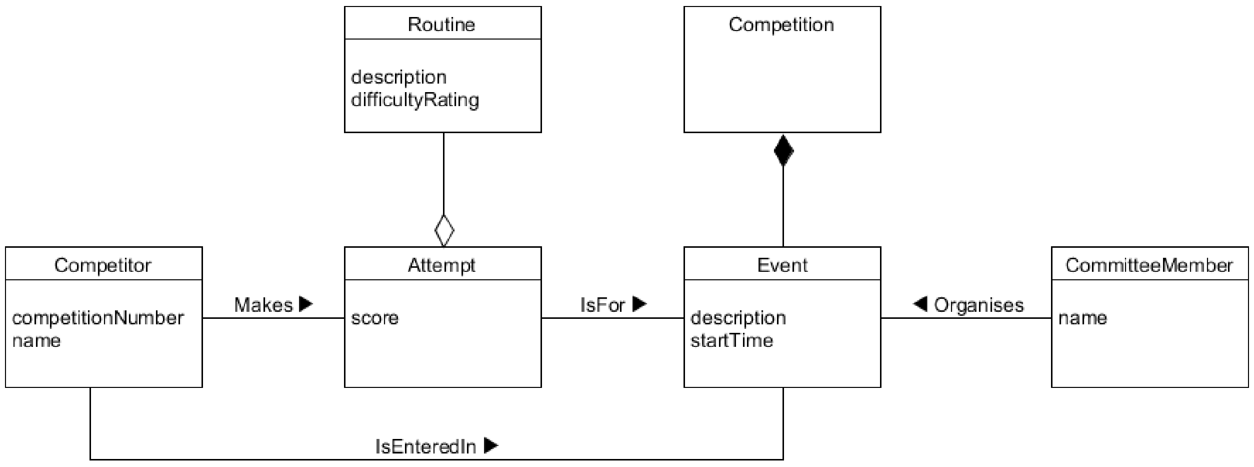


Figure 4: Partial class model of the gymnastics domain

Your application should include Java classes for the above classes as a minimum, but you should bear in mind that the above illustration is only a partial model. You are not required to submit any UML for this task. The partial class diagram above is provided as a design guide for your Java implementation.

In order to operate with the console-based User interface, your application will require some 'dummy' data to be pre-coded into the application. This should be achieved by 'hard-coding' data within the Java program. It should **not** use any data layer component or SQL database link. There is no requirement to 'write' any data for the task. All pre-coded data should be *read-only*.

Task 3 (15 marks)

Create a new NetBeans project, called Task3.

For this task you are required to program a JavaFX Graphical User Interface (GUI), with a focus on layout and event handling.

This program must be coded as a JavaFX program following the approach described in Chapters 14 to 16 of the *kortext*. It must not be created using a visual drag-and-drop tool, and must not be an FXML program, neither of which have been taught on the module. If the submission is either a drag-and-drop application, or a FXML application, then it will simply be awarded zero marks.

The GUI application that you need to create is a basic 'smart home controller' application. The required design is illustrated in Figure 5 below.

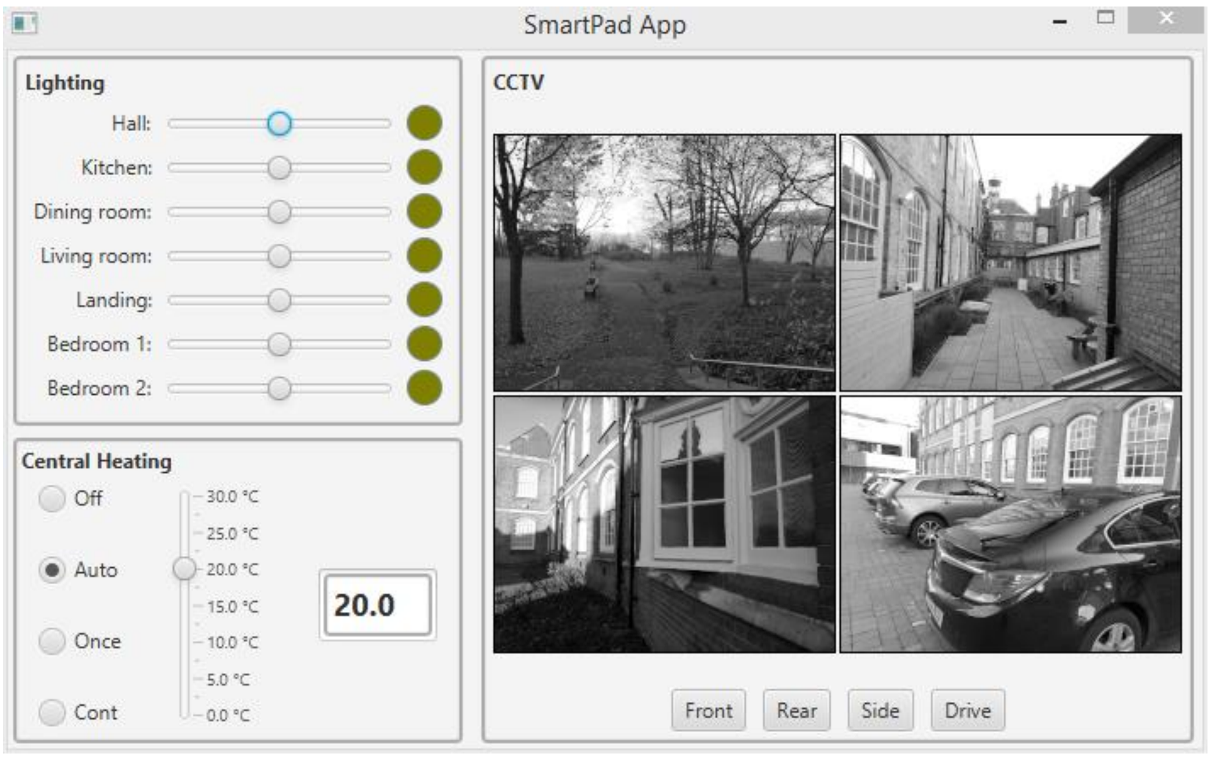


Figure 5: Smart home controller application user interface

The application comprises three areas representing controls for lighting, central heating and CCTV. Each area on screen comprises JavaFX nodes and controls. The lighting area comprises Labels, Sliders and Circle objects. The central heating area comprises a Label, RadioButtons, a Slider and a TextArea. The CCTV area comprises a Label, four ImageViews and four associated ToggleButtons.

a) Write a JavaFX program that replicates (accurately) the GUI layout as illustrated in Figure 5. The focus should be entirely on accuracy of layout, and there are no marks for styling using CSS. The illustration shown in Figure 5 is the default look and feel of JavaFX in Windows NetBeans 8 and that is all that is expected of your program.
b) Implement event handling for the GUI as follows:
   - For the lighting area, the colour of each Circle object should change in response to the associated horizontal slider being dragged. When the Slider is dragged to the left the colour should 'darken', and when dragged to the right, the colour should 'lighten'.

- For the central heating area, the RadioButtons should operate correctly in that only one RadioButton can be selected at a time (i.e., they are mutually exclusive in terms of selection). The Slider must be vertical and when dragged up or down the value in the TextField must be updated to reflect the value on the Slider.
- For the CCTV area, each of the four ImageViews should be mapped to one of the four ToggleButtons. Clicking on any of the four ToggleButtons should result in the associated image being 'blacked out' (clicking again would reveal the image again).

The results of these interactions are illustrated in Figure 6 below

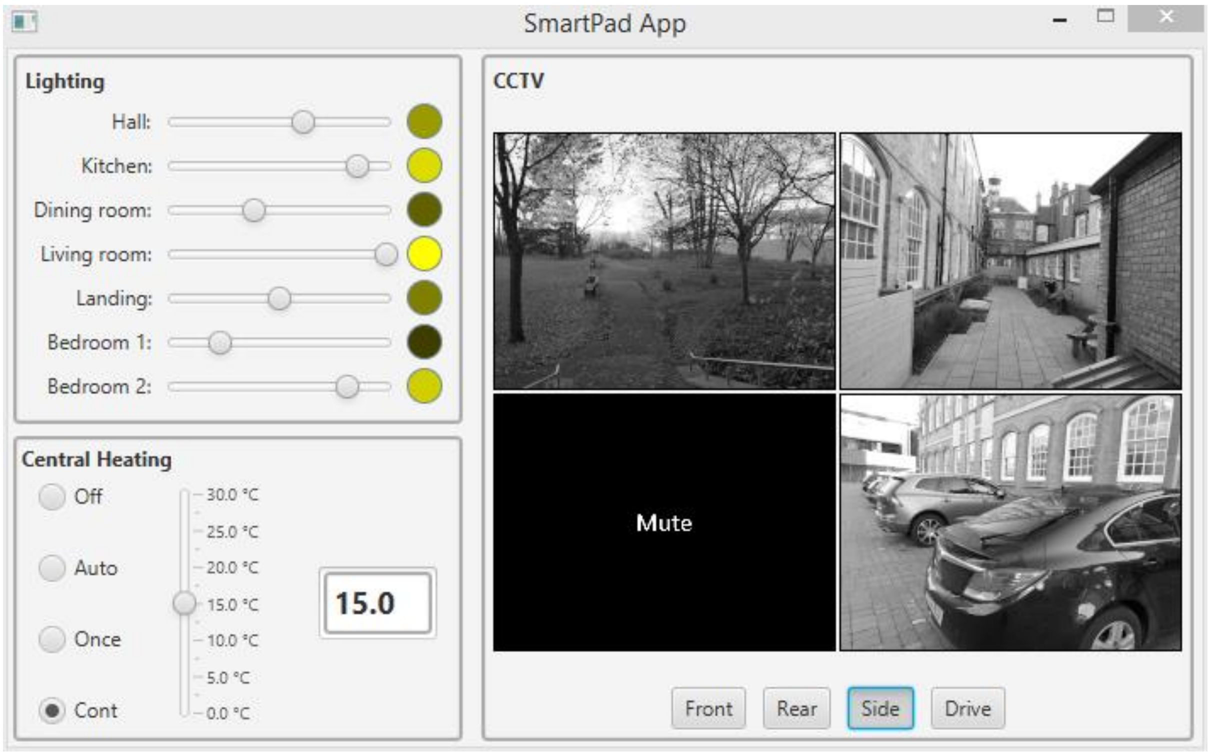


Figure 6: Event interaction with the smart home controller application

The images are provided in the Task3 project template that is provided within the assignment download file.

Viva Voce Assessment

Please note carefully that his assignment is an *individual* assessment.

Following submission and marking of assignments, and as part of the module assessment moderation procedure, a cross-section of students will be required to attend a follow up viva voce assessment with the teaching team. These appointments will be arranged after first phase (provisional) marking has been completed.

Appendix A

Option 1: list traders

On selecting option 1, the User should be presented with a neatly formatted listing of the trading companies. The data displayed should be the company name, the location, and the services offered by the company. Most importantly, the listing should also provide a unique identifier in order that the User can use this to select a single trading company to view all the details of that company (i.e., option 2).

```
--------------------------------------------------------------------------------------------------------------------------
| ID | Company name          | Location         | Services offered                                                       |
--------------------------------------------------------------------------------------------------------------------------
|  1 | Bob's Builders        | Finchley Central | Brick laying, Scaffolding, Joinery                                     |
|  2 | Plaster Magic         | Brent Cross      | Plastering, Rendering, Fascias, Painting and Decorating                |
|  3 | Bricks and Blocks     | Bounds Green     | Brick laying, Walls, Garage roofs                                      |
|  4 | Decorating Delight    | Highgate         | Wall papering, Plastering, Painting, Decorating                        |
|  5 | Gorgeous Gardens      | Hendon           | Gardening, Landscaping, Composting, Clearance                          |
|  6 | Motor Madness         | Southgate        | General motor maintenance, Engine tuning, Breakdowns, MOT Testing      |
|  7 | Motor bodies          | Oakwood          | Crash recovery, Bodywork, Respraying                                   |
|  8 | Honest Plumbing Ltd.  | Willesden Green  | Plumbing, Water leaks, Pipework                                        |
|  9 | Tim's Tilers          | Totteridge       | Roofing, Tiling, Stack alignment                                       |
| 10 | Sound Satellites      | Edgware          | TV Aerials, Satellite Dishes, Cable TV installation                    |
| 11 | Enfield Engines       | Enfield          | Engine tuning, General motor maintenance, Breakdown assistance         |
| 12 | Brian's Bathrooms     | Barnet           | Plastering, Tiling, Plumbing, Bathroom suites                          |
| 13 | Everything Electrical | Enfield          | Wiring, Electrics, Lighting                                            |
| 14 | Woody Furniture       | Woodside Park    | Carpentry, Custom furniture, Joinery                                   |
| 15 | Sky’s the limit       | Mill Hill East   | Satellite dishes, Cable TV installation, TV repairs                    |
| 16 | Harry the Plumber     | Highgate         | General plumbing, Bathroom fitting                                     |
| 17 | Derek the Decorator   | Barnet           | Painting, Decorating, Wall papering, Plastering                        |
| 18 | Scuff and Scratch     | Finchley         | Motor bodywork and repairs, Resprays, Dents                            |
| 19 | Carl's Carpentry      | Hendon           | Joinery, Carpentry, Woodwork                                           |
| 20 | Graham's Garage       | Totteridge       | MOT inspections, Motor Mechanics, Engine tune-ups                      |
| 21 | Electric Dreams       | Enfield          | Electrical work, Wiring, Electrical testing, Electrical installations  |
| 22 | Busy Bee Cleaners     | Barnet           | Cleaning services, Domestic cleaning, Ironing                          |
| 23 | Pests r' us           | North Finchley   | Wasps, Ants, Mice, Rats, Pigeons, General pest control                 |
| 24 | Barry's Bricks        | Brent Cross      | Brick laying, Building, Construction                                   |
| 25 | Autumm Leaves         | Southgate        | Garden clearance, General gardening                                    |
--------------------------------------------------------------------------------------------------------------------------
```

Option 2: select trader

On selecting option 2, the User should be prompted to enter the unique identifier of a listed trading company (that was displayed when option 1 was chosen). Following this, all the details of that company should be displayed, as illustrated below. This should be achieved by invocation of the relevant *Trader* object's *toString()* method. Console output should be neatly formatted.

```
List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:> 2

Enter trader ID from list [1 - 25] :> 16

-----------------------------------------------------------
Company name:            Harry the Plumber
Location:                Highgate
Services offered:        General plumbing, Bathroom fitting
Number of employees:     1
Daily call out rate (£): 175.0
Other information:       General plumber available 24/7.
-----------------------------------------------------------


List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:>
```

Option 3: search locations

On selecting option 3, the User should be prompted to enter a ‘search string’ to match against the location field of the Trader objects. All searches should be case insensitive. The program should return a listing of trading companies that match using the same neat formatting as the option to list all trading companies, as illustrated below:

```
List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:> 3

Enter search text > Finchley

---------------------------------------------------------------------------------------------------------------------------
| ID | Company name           | Location         | Services offered                                                       |
---------------------------------------------------------------------------------------------------------------------------
|  1 | Bob's Builders         | Finchley Central | Brick laying, Scaffolding, Joinery                                     |
| 18 | Scuff and Scratch      | Finchley         | Motor bodywork and repairs, Resprays, Dents                            |
| 23 | Pests r' us            | North Finchley   | Wasps, Ants, Mice, Rats, Pigeons, General pest control                 |
---------------------------------------------------------------------------------------------------------------------------

List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:>
```

Option 4: search services

This option should work in the same way as option 3, but this time the search should match on the services offered by the trading company, as illustrated below:

```
List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:> 4

Enter search text > plumbing
--------------------------------------------------------------------------------------------------------------------------
| ID | Company name           | Location          | Services offered                                                     |
--------------------------------------------------------------------------------------------------------------------------
| 16 | Harry the Plumber      | Highgate          | General plumbing, Bathroom fitting                                   |
|  8 | Honest Plumbing Ltd.   | Willesden Green   | Plumbing, Water leaks, Pipework                                      |
| 12 | Brian's Bathrooms      | Barnet            | Plastering, Tiling, Plumbing, Bathroom suites                        |
--------------------------------------------------------------------------------------------------------------------------

List traders.......1
Select trader......2
Search locations...3
Search services....4
Exit...............0

Enter choice:> 0
```