# 06-08163 / 06-08166 – Software Workshop Java [(A)]

## Exercise 2: Route Finder

### Instructions for students:
Fill in this form with your name and registration number.

When you have completed each exercise component, tick the corresponding box in the column marked ‘Claimed Done’. You need not have completed all the work in order to have your exercise solution assessed by the demonstrators, but you must have submitted the work that you claim to have completed. You get a mark if you have made a reasonable effort, even if you have not completed everything. Ask a demonstrator to come and have a look at your work. The demonstrator will ask you questions on the work that you claim to have done.

### Your details:
Student Registration Number:  
Surname:  
Forename(s):  

### Marking scheme:
Example routes are shown on the reverse of this form, to aid testing.

<table>
<thead>
<tr>
<th>Unit of Work</th>
<th>[ ] Not Started</th>
<th>[ ] In Progress</th>
<th>[ ] Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[1] Accessing the database</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 1: Does the program work correctly when it is given two arguments that are station names?</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does the program respond sensibly if given an incorrect number of arguments or arguments that are not valid station names?</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>[2] Design</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A clear, neat diagram showing all the classes used, with accurate relationships between them.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A clear, concise narrative, which explains the design and includes an explanation of the route finding algorithm used.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choice of classes: Do they correspond to natural concepts or actors. Are they named using nouns?</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choice of classes: Low coupling between classes and high cohesion of each class.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>[3] Code</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The code implements the design effectively.</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Methods do one task. Variable names are sensible. Classes begin with Uppercase letters, variables / methods with lowercase.</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use of database connections. Are all connections are being closed (or possibly reused)?</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>[4] Javadoc</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The code displays clear Javadoc.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Javadoc has been compiled into a meaningful set of API web pages.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>[5] Functionality</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 2: Constructing Routes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The program can at least produce a route satisfying the requirements, even if only in some cases.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The program reliably produces all routes (or all optimal routes) satisfying the requirements.</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The program only produces optimal routes satisfying the requirements (extra credit).</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 3: Dijkstra's Algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The program codes Dijkstra's Algorithm. The code is clear and it is possible to see that it encapsulates Dijkstra's Algorithm.</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>It produces the route that arrives at the destination at the earliest possible time within the parameters specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part 4: Web Access</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student can explain how servlets work. The servlet works correctly. The response page includes a new form, so that the user can enter a new query.</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>[6] Additional Credit</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 points if the viva is conducted on Tuesday, 2 points for Thursday and 0 points for Friday.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total mark awarded:</strong></td>
<td>110</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Demonstrator name:  
Date of viva:  

Demonstrator signature:  
Student signature:
These are some examples the demonstrator may use to check your work. Remember that your program should behave correctly with any parameters, not just the examples shown here.

**RouteFinder1**
```
gromit java RouteFinder1 Birmingham Glasgow
Birmingham (6.55) to Glasgow (11.26)
```
```
gromit java RouteFinder1 Birmingham Edinburgh
Birmingham (9.05) to Edinburgh (13.36)
```
```
gromit java RouteFinder1 London Birmingham
London (5.44) to Birmingham (8.15)
```
```
gromit java RouteFinder1 Colchester Birmingham
No direct connection found
```
```
gromit java RouteFinder1 Liverpool Glasgow
No direct connection found
```
```
gromit java RouteFinder1 Paris London
Paris doesn’t seem to be a station name.
No direct connection found
```
```
gromit java RouteFinder1 Paris Munich
Paris doesn’t seem to be a station name.
Munich doesn’t seem to be a station name.
No direct connection found
```
```
gromit java RouteFinder1
Usage: java RouteFinder1 <origin> <destination>
```

**RouteFinder2**
```
gromit java RouteFinder2 Leeds Sheffield 9 12
Leeds (9.08) to Manchester (10.06), Manchester (10.19) to Sheffield (11.08)
Number of database results extracted: 37
```
```
gromit java RouteFinder2 Colchester Sheffield 9 20
Colchester (9.01) to London (9.55), London (9.55) to Manchester (12.36), Manchester (12.43) to Sheffield (14.37)
Number of database results extracted: 11569
```
```
gromit java RouteFinder2 Colchester Sheffield 9 20 3
Number of database results extracted: 11569
Colchester (9.01) to London (9.55), London (9.55) to Manchester (12.36), Manchester (12.43) to Sheffield (14.37)
Colchester (10.01) to London (10.52), London (10.55) to Manchester (13.36), Manchester (13.43) to Sheffield (15.36)
Colchester (11.01) to London (11.52), London (11.55) to Manchester (14.36), Manchester (14.43) to Sheffield (15.36)
Colchester (12.01) to London (12.54), London (12.55) to Manchester (15.36), Manchester (15.43) to Sheffield (16.35)
Colchester (13.01) to London (13.54), London (13.55) to Manchester (16.36), Manchester (16.43) to Sheffield (17.37)
Colchester (14.01) to London (14.52), London (14.55) to Manchester (17.36), Manchester (17.43) to Sheffield (18.36)
Colchester (15.01) to London (15.52), London (15.55) to Manchester (18.36), Manchester (18.43) to Sheffield (19.4)
```
```
gromit java RouteFinder2 Colchester Sheffield 9 13 3
No itinerary possible
Number of database results extracted: 270
```
```
gromit java RouteFinder2 Colchester Sheffield 9 22 1
No itinerary possible
Number of database results extracted: 6469
```

**RouteFinder3**
```
gromit java RouteFinder3 Leeds Sheffield 9 12
Leeds (9.08) to Manchester (10.06), Manchester (10.19) to Sheffield (11.08)
Number of database results extracted: 20
```
```
gromit java RouteFinder3 Colchester Sheffield 9 20
Colchester (9.01) to London (9.55), London (9.55) to Manchester (12.36), Manchester (12.43) to Sheffield (13.35)
Number of database results extracted: 636
```
```
gromit java RouteFinder3 Colchester Sheffield 9 13
null
Number of database results extracted: 60
```
```
gromit java RouteFinder3 London Newcastle 7 10
London (7.0) to Newcastle (9.54)
Number of database results extracted: 36
```
```
gromit java RouteFinder3 Warwick Leeds 7 16
Warwick (7.1) to Birmingham (7.46), Birmingham (8.0) to Manchester (9.46), Manchester (9.46) to Leeds (10.43)
Number of database results extracted: 338
```
```