Telerik Academy

Logistics App

Project Description

Design and implement a **Logistics** console application.

The application will be used by employees of a large Australian company aiming to expand its activities to the freight industry. The app will be used to manage the delivery of packages between hubs in major Australian cities. An employee of the company must be able to record the details of a delivery package, create or search for suitable delivery routes, and inspect the current state of delivery packages, transport vehicles and delivery routes.

Functional Requirements

The application **must** support the following operations:

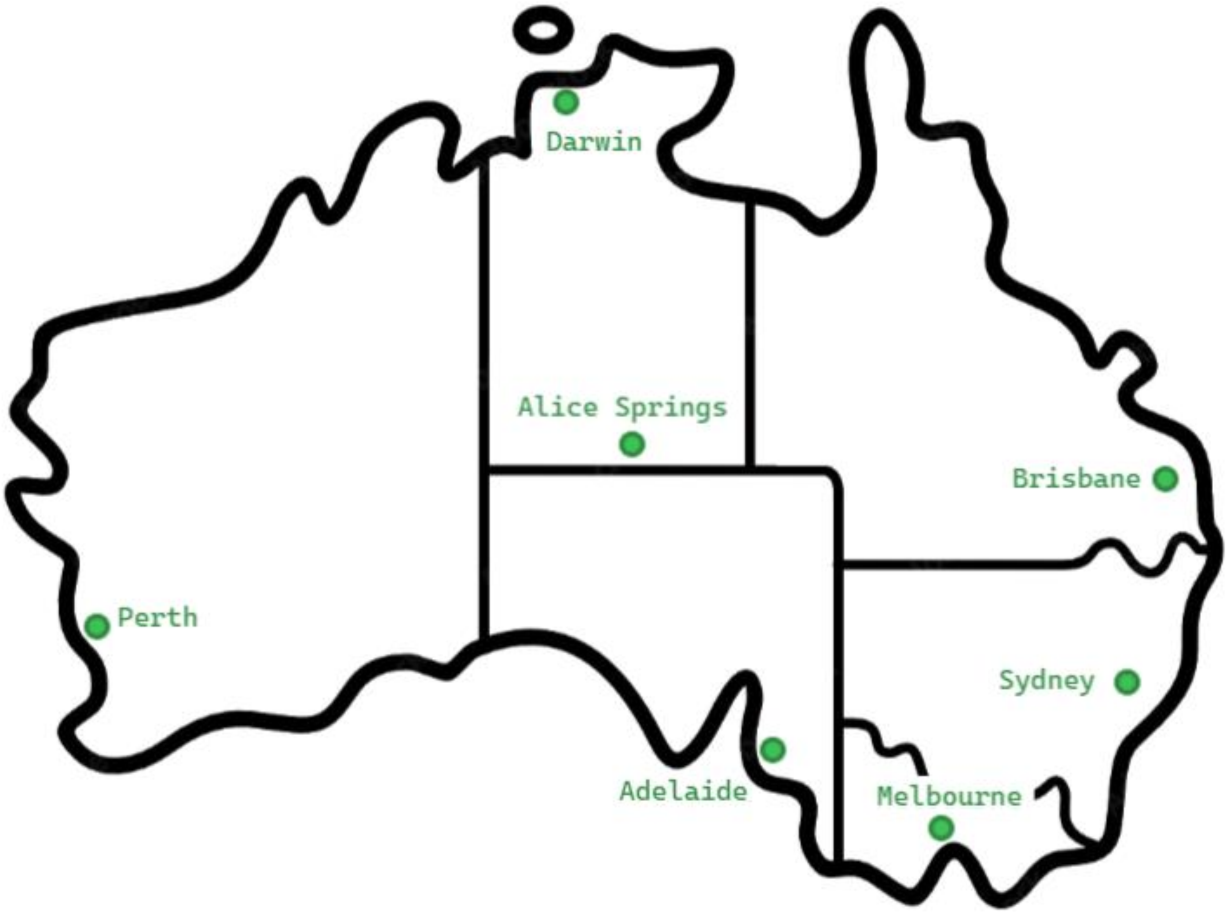
1. Creating a delivery package – unique id, start location, end location and weight in kg, and contact information for the customer.
2. Creating a delivery route – should have a unique id, and a list of locations (at least two).
   a. The first location is the starting location – it has a departure time.
   b. The other locations have expected arrival time.
3. Search for a route based on package's start and end locations.
4. Updating a delivery route – assign a free truck to it.
5. Updating a delivery route – assign a delivery package.
6. View a information about routes, packages and trucks.

The application **should** support the following operations:

1. Save the application state to the file system

The company owns the following transport vehicles:

Vehicle ids	**Name**	**Capacity (kg)**	**Max range (km)**	**Number of vehicles**
1001-1010	Scania	42000	8000	10
1011-1025	Man	37000	10000	15
1026-1040	Actros	26000	13000	15



Use the following distances in **km**

	SYD	MEL	ADL	ASP	BRI	DAR	PER
SYD		877	1376	2762	909	3935	4016
MEL	877		725	2255	1765	3752	3509
ADL	1376	725		1530	1927	3027	2785
ASP	2762	2255	1530		2993	1497	2481
BRI	909	1765	1927	2993		3426	4311
DAR	3935	3752	3027	1497	3426		4025
PER	4016	3509	2785	2481	4311	4025	

Use cases

Use case #1

A customer visits the company office in Sydney on Oct 8th. They bring a package that needs to be delivered to Melbourne. An employee of the company records the customer's contact info, weighs the package at 45kg and then checks for a suitable delivery route. The system reports that there are two routes:

- Brisbane (Oct 10th 06:00h) → Sydney (Oct 10th 20:00h) → Melbourne (Oct 11th 18:00h)
- Sydney (Oct 12th 06:00h) → Melbourne (Oct 12th 20:00h) → Adelaide (Oct 13th 15:00h)

Both routes' trucks have free capacity, and the employee suggests the first one, as the package will arrive one day earlier. The customer agrees and the employee uses the system to add the delivery package to the first route and to update the package's expected arrival time to Oct 11th 18:00h.

Use case #2

Many packages with total weight of 23000kg have gathered in the hub in Alice Springs and an employee of the company uses the system to create a route that leaves on Sep 12th 06:00h with the following stops:

- Alice Springs → Adelaide → Melbourne → Sydney → Brisbane

The system determines the route distance to 4041km and calculates estimated arrival times for each of the locations based on a predefined average speed of 87km/h. The employee then finds a free truck that meets the required range and capacity and proceeds to bulk assign the packages to the newly created route by using the route id and the packages' ids.

Use case #3

A manager at the company uses the system to find information about all delivery routes in progress. The system responds with information that contains each route's stops, delivery weight, and the expected current stop based on the time of the day.

Use case #4

A supervising employee uses the system to view information about each package that is not yet assigned to a delivery route. The system responds with a list of packages containing their IDs and locations

Use case #5

A customer contacts the office to request information about their package. The customer provides the id that they received when the package was created, and an employee enters the package id in the system. It responds with detailed information which is then emailed to the customer.

Technical Requirements

- Follow the **OOP** programming principles:
  - Encapsulate your objects.
  - Apply information hiding where necessary.
  - Decide between inheritance and composition properly.
  - Use polymorphism properly.
- Follow guidelines for writing **readable code**:
  - Adequate naming of variables, functions, classes, methods, and attributes.
  - Well-formatted and consistent code.
  - Well-structured and readable logic.
- Implement proper user input **validation** and display meaningful user messages.
- Implement proper **error handling**.
- Prefer using list comprehension where readability will be improved
- **Cover the core functionality with unit tests.**
- Use **Git** to keep your source code and for team collaboration.

Teamwork Guidelines

Please see the Teamwork Guidelines document.