1 Requirements

The second part of the project entails expanding the current game mechanics to allow transporting a variety of goods among cities and factories on large maps with dynamic economy. The player should be able to choose among several transportation options, manage his infrastructure and vehicles to which one can assign complex routes. The principal aim of this part of the project is to employ the concepts of object-oriented programming to differentiate various categories of objects (such as vehicle types, goods, stations, or factories). The following sections give some guidelines on how to do this, which can be adapted to the particularities of your project.

Economic simulation will ensure that the world will remain active even without player involvement. However, the choices of the player will matter for the development of the economy. The required extensions can be split into several categories, which are treated separately in subsections:

- Different cargo types.
- Different vehicle types.
- Stations.
- Factories.
- City consumption.
- New Maps.

1.1 Different Cargo Types

So far, the game only allows for passenger transport. Several new types of goods are to be added in the expansion. The exact definition of types of goods is left to your imagination, however we suggest that you support at least some goods from the following categories:

- Dry bulk cargo (e.g. ores, grain, coal, cement, etc.)
- Liquid cargo (e.g. oil, water, milk etc.)
- Boxed/Container cargo (e.g. most consumer goods)
- Individual cargo (e.g. vehicles, livestock)

Each type of goods will have a predefined average/starting/default price, and a upper and lower price range limits (possibly functions of the default price). The goods are produced by factories (discussed later). Each city keeps a store of each type of goods and maintains current prices for which the goods can purchased/sold in the city (see also the city consumption section). Additionally, goods will spread even without being transported by the player. If a city has an abundance of a certain type of goods, the goods will start to move to neighbouring cities in which there is a demand for that type of goods.

1.2 Different Connections and Vehicle Types

Currently, the game only allows the player to buy trains to transport passengers. In this part, we wish to differentiate multiple types of transport. The maps (see below) will allow different types of connections between cities. For instance, the cities of London and Cambridge might be connected by rail and road, whereas Dover and Calais might be connected only by sea. Different types of connections would be served by different vehicle types. Moreover, the connections can have parameters depending on their type: a road may have a speed limit, and a naval connection may be limited to ships of a certain size.

At least the following vehicle types are required:
• Trains – Modular, they can be composed of different combinations of engines and carriages. The engines should be split into at least two categories (electric/diesel) and carriages should be specific to good categories (or individual goods types). Each engine/carriage should have it’s maximum speed and weight. Engines will also have tractive effort, which determines how heavy trains they can pull (at what speeds). Carriages will additionally have capacity (affecting their loaded weight).

• Buses/Trucks – Cargo category (goods type) specific. Should have a maximum speed and capacity.

• Ships – At least one ship should be available for the main categories: bulk carriers, container ships and tankers. Ships should have maximum speed and capacity as well as (beam) size to determine which waterways can be traversed.

• Aircraft – Aircraft ignore city connections, they simply connect two cities having an airport (see the section on Stations). Each aircraft should have capacity, cruising speed and cruising altitude. Additionally, different aircraft may require different runway lengths depending on their weight (affected by load).

Vehicles will also have running costs that the player has to pay. Travel speed and running cost are subject to route conditions (e.g. pulling heavy trains is more expensive, sailing upstream is more expensive, planes need to climb to their target altitude regardless of route length, etc.).

As opposed to current version, where routes have only two destinations, at least some vehicle types should be able to follow routes spanning several cities (not necessarily hamiltonian) provided the correct type of stations (see below) and infrastructure (inter-city connections) are available.

1.3 Stations

Up till now it was possible to just send a train between cities without requirements on infrastructure. From now on vehicles (trains/ships/planes) will be required to use appropriate infrastructure to access a city (train stations/ports/airports, collectively called stations).

Stations can have individual limits on how many vehicles a player can send to a city, or the size of allowed vehicles (e.g. length of platforms, length of runway). These restrictions can be alleviated by expanding the stations (different kind of expansion). The cost of running a station will be related to its performance (bigger stations cost more to build and operate).

Some stations (ports, airports) may already exist on the map, owned by the cities, they can be used by the player for a fee, but have to be purchased before expanding.

1.4 City Consumption

Cities will now have a demand for consumer goods (e.g., food but not iron ore). The cities will consume their demanded goods on designated ticks (economy ticks), not necessarily every tick of the game (vehicles should be able to move faster). Cities only consume goods if they are available in their store.

If demand was not met in the city for some type of goods, the price of the goods will increase in the city (limited by the price range). On the other hand, if demand was fully satisfied, the price should decrease. (And possibly, the goods should leave to neighbouring towns that have higher demand.)

The demand of the city is shared with the demand of the factories in the city (more below), i.e. a city may not demand wood at all or only in very limited quantities, but a city with a lumber mill will demand much more wood, depending on the lumber mill consumption (with full effect on price changes).

1.5 Factories

Distributed among the cities are production facilities (collectively called factories). Factories are the only way to produce goods. Factories are of different types depending on the production chain (goods produced). The factory types you include should be related to the goods you support.

Production line comprises zero or more input goods (in respective quantities), required to produce one or more output goods (in respective quantities). Moreover, the production will have additional costs (labour, facility maintenance, etc.) that are statically determined per factory type. Production happens concurrently with the city consumption (on economy ticks). For example, an economy tick may look like this:

1. Finish previous production: The factories sell the finished production (started on previous economy tick) to the city stores.

2. Consumption step: Cities and factories will consume the goods they demand, provided they are available in the stores.
3. Goods movement: Leftover goods may embark on a travel to neighbouring cities, provided the demand is larger there.

4. Price calculation: Cities update their price lists depending on the level of demand satisfaction and amount of leftover goods.

5. Start new production: The factories will start new production, from the goods they have obtained in the consumption step.

Factories may take longer than one economy tick to produce their output. In fact, the behaviour of the factories is differentiated according to the goods that they produce. Your simulation should support different factory types such as:

- Farms – Produce large quantities only in short time period each year, however, require labour throughout the year.
- Ranches – They have a set amount of animals that grows when fed.
- Mines – Periodic output of goods.
- Simple factories – Converts inputs directly into outputs.
- Complex factories – Includes several stages of production, each production stage may take one or more economy ticks and later production stages take the outputs of the earlier production stages as inputs. Production stages may be exclusive or parallel, depending on whether they can happen in the same tick (provided that all intermediate goods are available from previous ticks).

Finally, factories are "smart". This means that at the start of their production cycle, they will decide how many goods to produce depending on the current prices of input and output goods and constrained by the size of the factory. This leads to a simple optimisation problem which can be tackled with linear programming.

Note that as mentioned above, the production cycle can be longer than one tick (it takes the whole year for farms), and the prices may change during this time, however, factories have no way of predicting this change.

1.6 Maps
The game should support a range of different maps. Several maps will be provided that your implementation should be able to handle. The maps are formatted as xml files that may look like this:

```xml
<xml version="1.0" encoding="utf−8"/>
<Map name="not_EU" width="100" height="200">
  <City name="London" x="−25" y="−75" population="8869898">
    <Factory type="sugar_refinery" size="5" />
    <Airport size="475" runway_length="3902" />
    <Port size="50" beam_clearance="70" />
  </City>
  <City name="Cambridge" x="25" y="75" population="155806" />
  <Connection upstream="London" downstream="Cambridge">
    <Road length="103" maximum_speed="112" lanes="2" />
    <Rail length="89" maximum_speed="160" tracks="2" electrified="yes" />
    <River length="132" beam_clearance="3" />
    <Canal length="87" beam_clearance="4.5" />
    <Sea />
  </Connection>
</Map>
```

As noticeable from the example, several extensions are added to cities (i.e. factories) and especially the connections between them. These include type of connection (rail, road or water), maximum speeds and restrictions on vehicles. Your implementation should support the provided maps, but you can create your own extensions of the map syntax, as long as they remain backward compatible.
2 Evaluation Criteria

2.1 Report and Presentation
You will have to submit a short (2-3 pages) written report (ideally \LaTeX-generated PDF). In the report you should explain the implementation challenges you encountered and describe your solutions. In case your program has any defects or shortcomings, list them in the report explaining the reason for their existence. Each group will deliver a 10 minute presentation of their project at the end of phase 2.

2.2 Functionality
Naturally, the functionality of your program will play a major role in its evaluation. Bonus features on top of the ones requested in this document may yield bonus points. The quality of graphical user interface (GUI) will also be taken into account. Of course, this is not a UX design course, so it will be evaluated from the functional perspective, however, intuitiveness and ease of use will be taken into account.

2.3 Code Organisation
Your project must be organised hierarchically, e.g. code split into files, classes and methods. Try to separate functionality as much as possible. It is especially advised to separate the data layer (business logic) and the user interface.

2.4 Code Quality
Proper use of OOP and the Scala language are important criteria for evaluation. Pay attention to proper use of inheritance. Avoid code duplication. Prefer functional constructs over nested loops/if statements. Stick to the naming conventions of Scala, if at all possible. Try to name your variables/methods/classes etc. with names that well describe their role in the program. In case you use a construction whose purpose is not obvious from the code, please document this in a comment.

3 Important Dates

- Deadline for submitting the code and the report: \textbf{Tuesday 10/04/2018}.
- The presentation of phase 2: \textbf{Friday 13/04/2018}.