TEAM Manual Book

# Introduction

Since the beginning of modern transportation, fossil fuel has always been the main energy source. Recently, people gradually recognize that an awful lot of consumption of fossil fueling could be a threat to our next generation or us. All kinds of new energy vehicles, such as electricity or fuel cell vehicles, are becoming more and more popular.

Different from fossil fuel vehicles, new energy vehicles (generally pure battery vechile) need more time to fueling, comparing to gasoline or diesel, whose charging time could almostly be omitted. So how to choose charging location, how to plan new charging station location and some other concerned problems are hot spots of transportation field.

TEAM emulational platform offers a universal solution to this question. For different cities, user could use python script template we offered to standardize different source data, then to compute future projection about charging station, new energy vehicle quantity of sale

This manual is written for giving some instructions to anyone new to this simulational platform. We will use Chicago as an example to detail all the Data input, simulate mechanisms, output, etc.

# Platform Outline

# Input Module

It’s clear that we couldn’t find source datas with identical format for different cities. So, TEAM core part needs user to offer a standardized data set, which should include geography data, travel survey, current new energy vechile detail information. We offer a python script for our users to process original data using Chicago as an example. User can easily format their source data with our python script.

Figure 2‑1 shows input module’s structure, All the source data could be divided to two parts, XML and shapefile. For processing these data, preprocess part includes some tools which could be used to add or adjust attributes of original data and convert different source data into a unified data format. The blacked cached part is for accerlerating simulation initialization. If the xml.file hasn’t been changed, Team will use cached data to initialize agent, this could increase speed of the initialization.

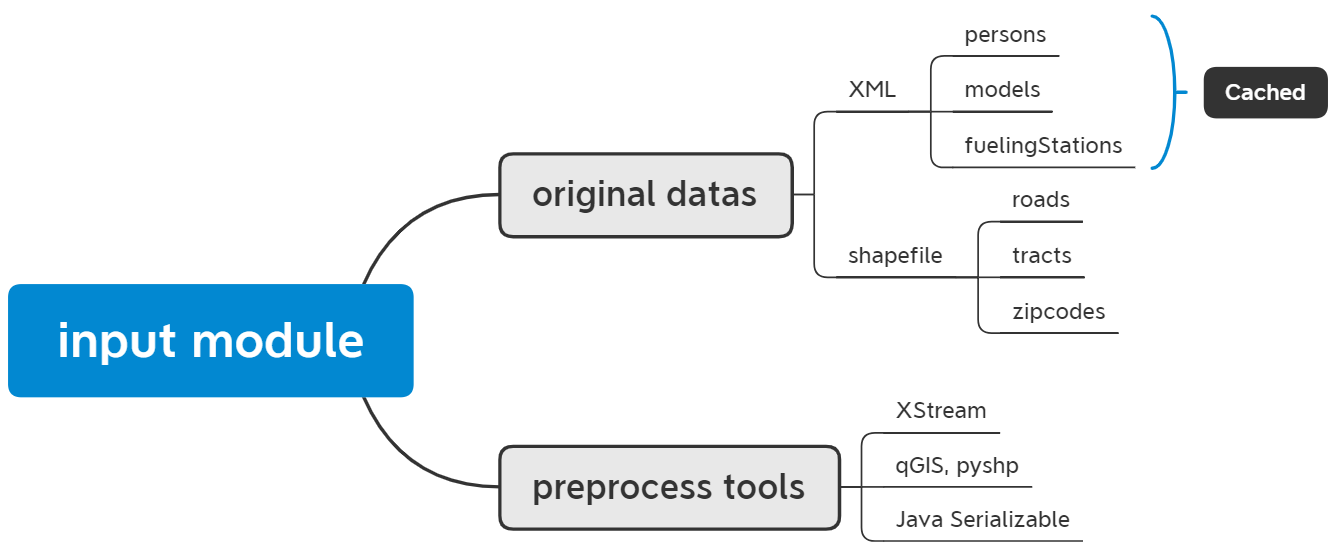


Figure ‑ Input Module Structure

## Xml, according the structure

TEAM uses xml file to store travel survey data, including persons.xml, models.xml, fuelingstations.xml. In the example of TEAM, all the original source datas are csv file, it may come from some official agencies, or some non-profitable organizations.

### Person

Figure 2‑2 comes from persons.xml file. If you are not familiar with xml file. A ten minutes tutorial about it from the internet should be enough. From now on, it’s assumed that our reader already knows the basic tree structure of xml file. In persons.xml file, <persons> root element stands for the set for persons, and every entry is an individual person data. It is composed by ID and a person element; this structure could be easily converted into a Map data structure in Java. ID is a unique identifier for this person, and person element includes all the required data for her or his travel simulation. Every person has an attribute called houseHoldID, which is a unique mark for his or her family, and used to calculate the household buget for buying a new energy vehicle. Another attribute is a zipcode which could be used to select person for different strict. The content in a person element looks like a list. When this xml file is converted into a Java object by XstreamTM, it is a map Object, ID is the key and person is the value. Inside any person value, the task list will be transfer to a Linkedlist, which is a concept from computer science. It means a string of points, every point is task, which can only be touched in the linked order, but you can add or delete the task when a task need to be substituted or deleted when this task refers to a charging task or for any other reason.



Figure ‑ Basic Person XML File Format

Now, it comes to the first key point of TEAM, Task. TEAM defines a travel as a Task, which includes depart time, arrive time, purpose ID, the coordinate of depart location and arrive location. With these data, TEAM could have a complete task for a vehicle. About task, there some crucial details to mention. Arrive time and depart time are not accurate time where task must begin or end. They are requirments or could be interpreated as the earliest time to depart or the latest to arrive. Though, task has these definitions, it’s very common that a vehicle doesn’t arrive in time, and it’s the point we care. Isn’t there enough charging station or is the planning rational enough? TEAM core purpose is to offer some advice about new energy charging station planning or charging planning. When a person starts a daily life, it will move according the task list, meanwhile, it will check the tank from time to time to decide a charging location. These details will be introduced later.

### Models

Models.xml is a file including a bunch of vehicle maker model, these models could be used to initialized vehicle attributes, such as vehicle type, battery capacity, etc.

Figure 2‑3 is an example of model item. Its type is PHEV, plug-in hybrid electric vehicle, which has a battery and a gasoline tank. We use this entry to introduce models.xml structure. Every model item in models.xml is an entry. Its basic structure is same to the previous persons.xml. Inside the entry, ID is the unique identifier of this model item. VehicleType stands for fueling type, including BEV, FCEV, PHEV. According to different vehicleType, the attribute will change. PHEV includes electric fuel and and liquid fuel. BEV only has battery and FCEV, only liquid fuel. For example, BEV has elecFuelingStyles and MPKWH, which belongs an electric charging style, but FCEV not.



Figure ‑ Vehicle Model Data

Take a loot at Figure 2‑3. BatteryCapacity is how much energy the battery could offer. The element includes concrete number and its unit. Please remember it is crucial, TEAM platform needs to process different units from time to volume, from mile to kilowatt hour. Any unit’s error could lead to a complete meaningless work and a very insidious bug. ElecFuelingStyles could be L1 L2 L3, which stands for different charging speed for EV, and L3 also has another name, super dc charger. You might noctice that its element nam is elecFuelingStyles, not elecFuelingStyle. It could be like L1, L2 and L3. It is apparently that an EV could be charged by different charging style. We prefer L3, but sometimes only a L1 outlet is available. So elecFuelingStyles is prioritized list. Any vechile will prefer preceding charging style when it is available. By the way, liquid fuel might not face this problem when it only has one choice, gasoline or diesel, but there may be one day where FCEV could choose different fuel charging style.

### FuelingStations

FuelingStations is composed of all the required electrical or liquid fueling station data, coordinate, charging styles, available number. Every time, a vehicle come to a station for charging or leave, if there are available charging interfaces, relevant number will vary once.

Figure 2‑4 is a concrete example of a charging station, includes its coordinate and all the available charging interface number.



Figure 2‑4 Fueling Station XML File Structure

OriginalFuelingStyles includes six kinds of charging outlets. We assume that every kind of fuel is enough, and the only limition is the charging outlet number. It’s reational.

## Shapfile

The shapefile format is a geospatial vector data format for geographic information system (GIS) software. It is developed and regulated by Esri as a mostly open specification for data interoperability among Esri and other GIS software products. The shapefile format can spatially describe vector features: points, lines, and polygons, representing, for example, water wells, rivers, and lakes. Each item usually has attributes that describe it, such as name or temperature.

In TEAM project, simulation need basic city road, tract, zipcode geography geometry to show all the city’s landscape, vechile travel track, etc. Considering different source shapefile, python script could be used to adjust the shp file attributes. For example, in our example of Chicago simulation, roads shapefile from the government doesn’t include real geographic length, but longitude and latitude are offered. Example python script uses coordinate data to calculate real geographic length. This procedure needs a third-party library shapefile.py.



Above is a code snippet of adding geographic length attribute. The coordinate data are used to compute the real geography length. This document won’t detail these stuffs. If our reader is not familiar with python, maybe one day tutorial is enough. We just use some basic python grammar.

# Program Structure

## Repast introduction

The Repast Suite is a family of advanced, free, and open source agent-based modeling and simulation platforms that have been under continuous development for over 15 years, <https://repast.github.io/>.

TEAM is built on this platform. Agent-based simulation is a kind of scientific research method for individual behaviror simulation. Every vehicle, every person, even every road could be an agent and we can set its individual behaviror. You may be curious about why road could be an agent. Our TEAM project could simulate a complete city transport system, so for a road, the congestion of a road will change according to the number of vechicle, weather, etc.

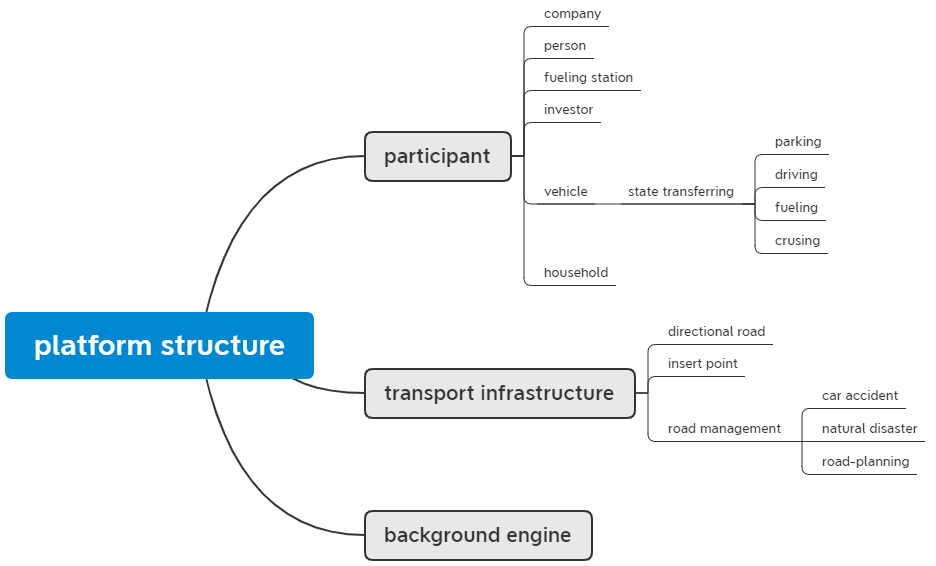


Figure ‑ Program Basic Structure

Figure 3‑1 is basic structure of TEAM simulate platform. Among different agents, participant includes TNC companies, persons, households, investors fueling stations and vehicles whose data come from XML survery data. Vehicle is the most crucial agent type in this simulate platform, it bears the core movment procedure. Roads, tracts and zipcodes coming from shape file also could be called agent, but they don’t play that role much, so we choose to define these ones as basic geography infrastures. Background engine is like a manager, which plays the role of organize resouces and plan procedures, etc. All above content will be be detailed in following sections;

## Participant

Repast core mechanism could be described as a bunch of agents which are commanded by repast runtime regularly. In TEAM simulation, all agents could be divided into several categories: TNC company, individual person, fueling station and vehicle. These four kinds of agents will communicate and interact each other, then organize a complete simulated result.

One of features in this simulatant platform is we use state machine tranfering to implement agent behaviors. For example, a vehicle could be in one of three or four status: parking, driving, fueling or cruising if it’s a TNC vehicle. In Figure 3‑1, any behavior of a vehicle could be interpreted a transferring from one status to another. If the depart time is due, vehicle starts a new travel, so it could be implemented as parking status to driving status. Other agent included in participants also has similar procedure, but simpler compared to vechile agent.

Individual persons have tasks to finish, for private vehicle owner, TEAM will randomly designate a vechile as his or her private vehicle. The private vehicle agent will move according to the tasks.

Fueling station offers charging service for any new energy or common fossil energy vehicle. Initialization will add fueling station to the context. User could adjust fueling station’s position, charging interface number, In the future update, Investor should add fueling station dynamicly. The charging experience will directly affect investor decision.

Vehicle is another crucial part of this project, for now, we have two kinds of vehicles, TNC or private. Private vehicle’s tasks are the same to its owner. From the first task of one day, a vehicle will set the first task in the linkedList of tasks as its current Task. When the depart time of this task is due, then begin this traveling. When this vechile arrived destination, it will get next task in the tasks structure. For common circumstance, it almost takes up all the time. But there some special case related to fueling. These contents will be detailed in charging station estimator section.

We use a state machine in this process the status transferring, a private vehicle has three status: parking, driving, fueling. In any status of them, it could transfer to one of the others. So, we exploit Java “switch case” statement.

The survey of Chicago could be seen as a database. The common routine is severa

## Transport Infrasture

Transport Infrasture includes road, fuelingStation, tract, zipcode. Tract and zipcode belong to boundary type. It only offers a geography boundary for simulation. All the simulation data could be analysized inside a specific boundary, and it’s useful for investor to do some planning or predication.

Road plays an important role in this platform. All the road informations are extracted from a shapefile. Before a road agent is added, all the required attributes should be insert into a corresponding feature, which stands for an individual road in the shapefile.

There some important information about road. In real life, most of roads, except some countryside lanes, are directional. But it’s very common that the corresponding road shapefile doesn’t have the direction information. For this circumstance, we add some self-adapting method. If the road already offers the direction of this road, it’s Ok. If not, the road will be seen as a bidirectional road. Based on the original road, a clone road wil be added to the context, and these two road agent attributes will be set propriately according the original one. A directional road could be useful to simulate tide traffic question.

## Background Engine

We set a Background Engine to manage context, geography, agents, etc. Repast Simphony’s basic program structure is composed with context, and several projections, which are road network and geography for TEAM. Repast indeed offers an interface to initialize context and projections. It’s like a main function in C or C++ program language. It’s clear enough, but not appropriate for TEAM, since it need a global runtime engine to orgranize all the required information. For user, Background Engine is like a global manager who is accessible to every context, projection, agent.

So, we use a Background Enginee to initialize all the required information, such as agents, infrastructures and so on. Considering the global feature of Engine, the projection of future and the adjustment of simulation also are implemented in Background Engine. It’s handy. All the detail will be introduced in the next chapter.

## Operational Structure

Team model could be depicted as Figure 3‑2, Every blue bubble is an agent type of this system, and red dash line stands for behavior from one entity to another. TEAM platform implements all the agents, then using schedule property of repast to realize relationship among them.

Figure 3‑3 shows a complete transferring state machine for vechile. From above introduction, we know vehicle includes TNC vechicle and private vechicle. This illustration includes these catergories.

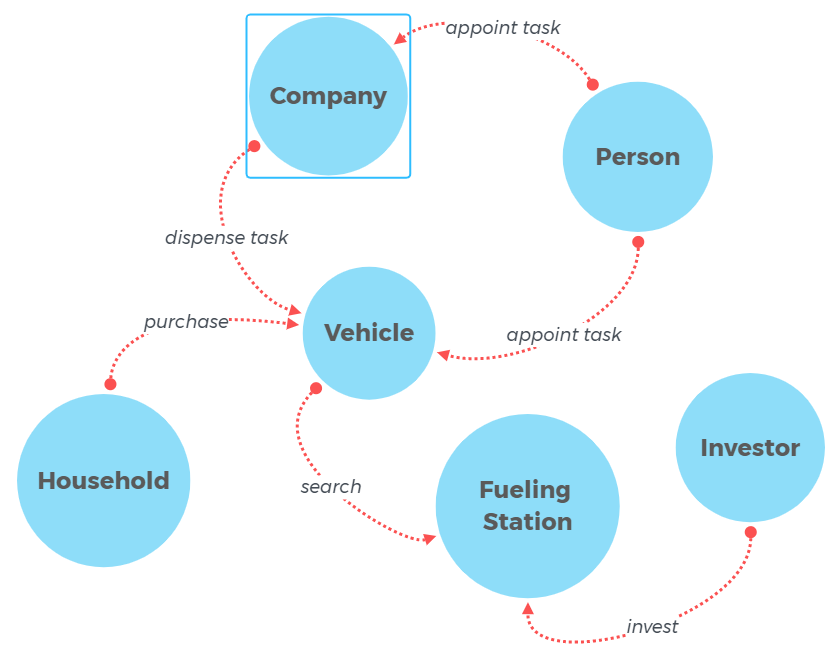


Figure 3‑2 Opertional Structure

Parking status could be transferred any other one if the condition is satisfied. When it’s a private car, the condition of time is due means the time is up for depart time, or this should have departed before. Then the status changes to Driving. When this travel is finished, or we can say this vehicle has arrived its destination, this vehicle enters another status, fueling, when this destination is a charging station, or parking when it’s a real task destination. If it is a on-route charging, this vehicle will transfer to driving when the energy is full or it’s time to leave, if not, this vehicle couldn’t finish this travel on time. If the fueling station is a depart place or a destination, after fueling, vehicle may transfer to parking status if the depart time is not due. Cruising is a special status of TNC vehicle. It is supposed that a vehicle is always driving or cruising except begin of working or off-duty. There are many other transferrings in this illustration, I believe it’s straightforward enough.

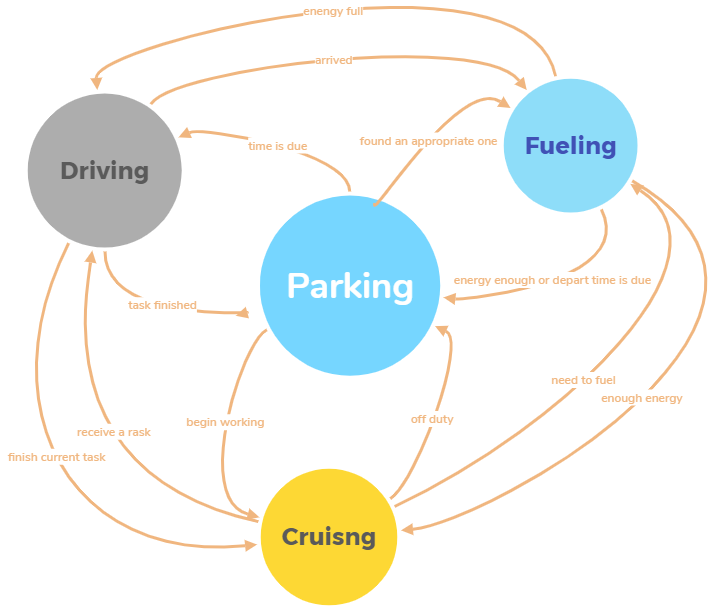


Figure ‑ Vehicle Transferring

# Crucial Part

New energy vechile’s main problem is charging. Previous chapter has mentioned it. Not like fossil vehicle, new energy vechicle’s mainstream product, pure electric vehicle, needs lots of time to charging. According some reports about the habit of new energy vehicle owner, commonly, they will roughly schedule a charging station in advance. For ones who are used to omitting the charging time, planning might affect the attitude of consumer to EV. In this simulation platform, we present a strategy that it could help driver decide which charging station is appropriate for current status, including, remain mileage, destination, idiosyncrasy.

## Charging Station Estimator

New energy vehicle is not like fossil fuel vechile where fueling time could be omitted. For new energy vehicle, charging time often is the most annoying problem. It needs a quite long time to charge compared to fossil fuel, even super charging of Telsa, the charging time also couldn’t be omitted.

How to get an appropriate charging station becomes a question. We assume that for new energy vehicle, driver need change their habit about fueling. They need to plan it in advance. Before a trip, driver should have a plan about when and where to fuel. Even it sounds complicated, I believe it will be finished in the future.

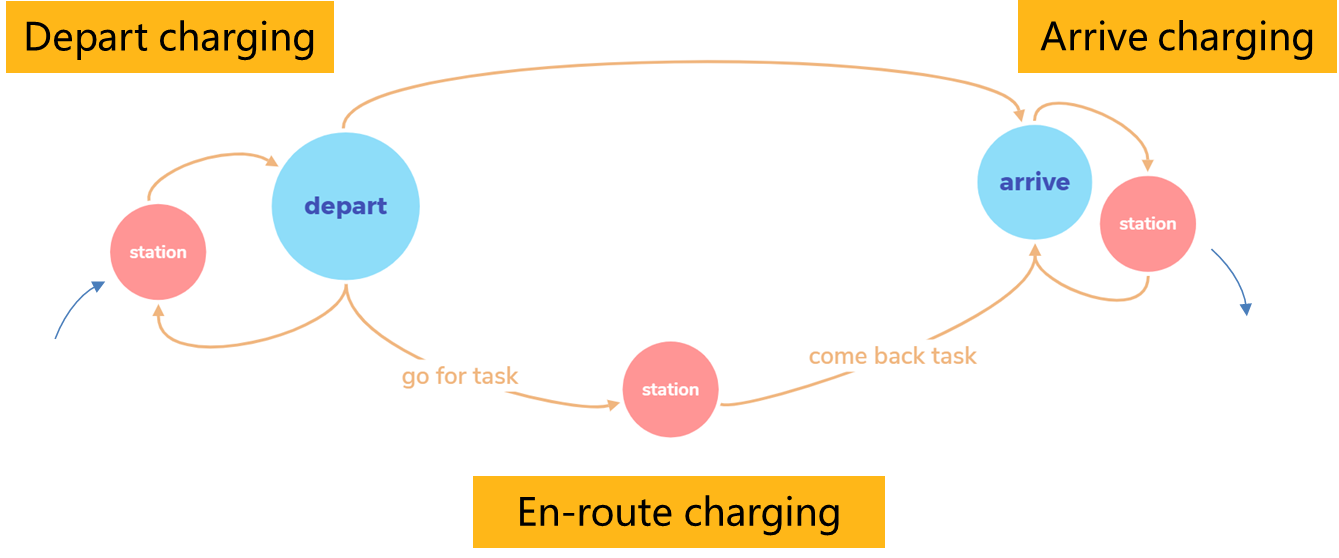


Figure ‑ Charging Strategy

In our simulation, we let our vehicle agent plan it fueling. The basic idea is illustrated in Figure 4‑1. For any driver, we check the first three trips (it could be changed), if the vehicle couldn’t finish one of them, then driver need to search an appropriate charging station. There are two basic method to find

1. At the depart place or arrive place, there is a station and the distance is smaller than the max walking distance.
2. On-route charging, this method chooses a station between starting place and destination.

For any station satisfied the requirement of time and geography, we will construct an Estinator for this station. It will estimate all the attributes the station offered, then get a utility number. Also, a discrete choose model will be introduced to choose the final one, not only dependant on utility number.

## Projection and agent update

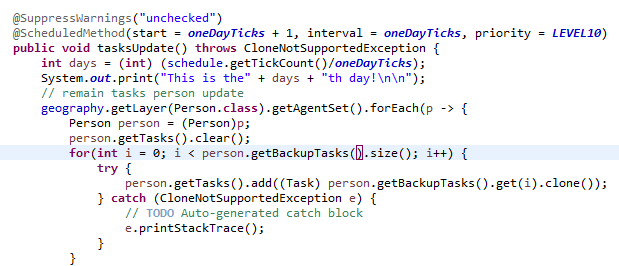


Figure ‑ Agent update

TEAM chooses a representative day as the simulation original data for one year. After a year’s simulation, more person and more vehicle should be added to the simulation. Background Engine will do this job like Figure 4‑2, user could add their method of person update to add agents about the next year’s simulation. In Figure 4‑2, the annotation above the function is a mark for this function, which could tell repast runtime that when and how often to run this method, and the priotity. There is a key point that the new added agent needs to be scheduled manly.

# Appendix

1. How to use a cached DijkstraShortestPath algotithum?

In TEAM simulation platform, the search of the shortest path needs to take carefully, thanks to its complex algorithum. Repast’s shortest path class employ the edu.uci.ics.jung.algorithms.shortestpath.DijkstraShortestPath class, but not very propriate. It has two constructor function, Figure 5‑1. Repast use the first one, and it didn’t use the cached feature.

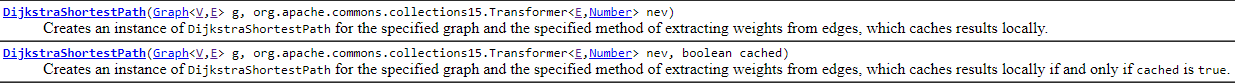


Figure 5‑1 DijkstraShortest Constructor

1. Data Class introduction

For the calculation of the quantity with unit，we offer a class named Data, which could help user to finish this interesting task. Any data with unit could use this Class as its type. The instance has a number and a unit name. All the suppored units has been organized as Enum set, which could be seen in the source code. For common plus and minus, the calculation could be finished if the two unit of the operand is identical. If they belong the same group, but not same, such as Second and Hour, you need to transfer it manly. The class offer a function for this. For multiplication and division, the key point is if we don’t transfer either unit, if the result unit is in the units Enum set. If not, then there will be an exception, and user need to transfer one of the operands or add a unit in the unit Enum set.

1. TeamShapefileLoader

It’s a customed class whose prototype comes from repast library. The prototype class could load geometry from shapefile and generate corresponding agent from the attributes in the feature. But，if we need some special operation for the geometry, it is powerless. For example, the basic geometry for tract and zipcode is multipolygon, but we need a single polygon. We offer a universial solution for this, for the customed shapefileLoader, an anonymous interface object that could be set outside is a parameter for the constructor. Client code could add customed code the geometry processing. For now, tract and zipcode need the transform from multipolygon to polygon, and road need a quite complicated operation which includes bidirectional road to directional road.