CS162 Solo Project Weekly Report: Week 05

CS162, Intro. to Computer Science II
Semester 2, AY2023/24

Solo Project

Full Name	Student ID
NGUYEN Hoang The Kiet	23125023

Lecturer: DINH Ba Tien (PhD)
TAs: HO Tuan Thanh (MSc), NGUYEN Le Hoang Dung (MSc)

Ho Chi Minh City, March 31, 2024

Tasks by Week

Week 05

1. (Section 1.1) Stop (the `stop` module), (Subsection ⓐ) `StopQuery`
2. (Section 1.2) Variant (the `variant` module), (Subsection ⓑ) `VariantQuery`
3. (Section 2) Querying List of Objects

Contents

List of Figures

List of Listings

Chapter 1

Elements of a Bus Network

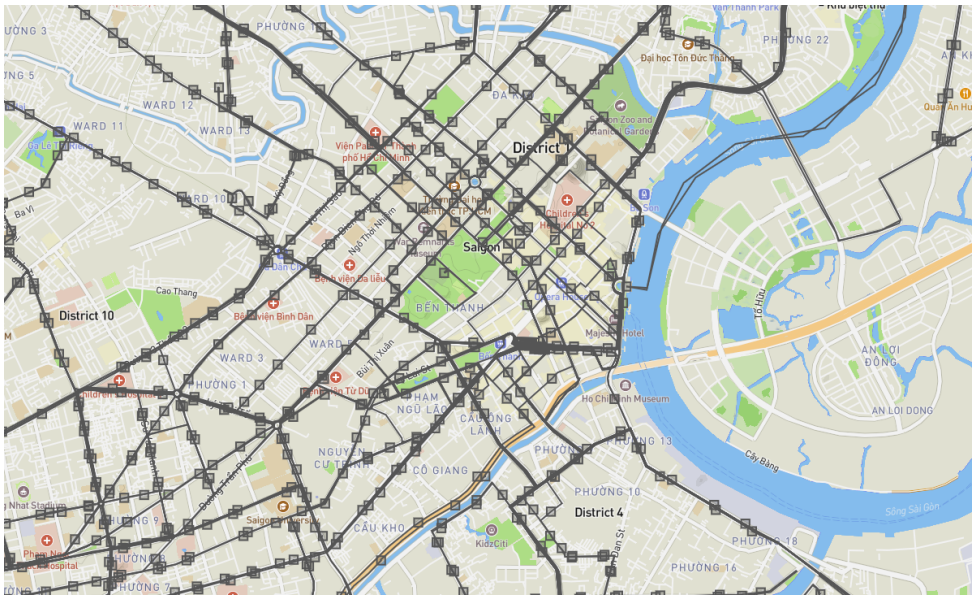

Figure 1.1: Elements of a Bus Network: The public bus network in Ho Chi Minh City. Squares represent bus stops, paths represent bus routes

1.1 | Stop (the stop module)

*Stop*s are the basic elements of a bus network. It defines places where buses stop temporarily to let passengers get on or off the bus.

ⓐ | Properties

- ***property*** `stop_id: int`
- ***property*** `code: str`
- ***property*** `name: str`
- ***property*** `stop_type: str`
- ***property*** `zone: str`
- ***property*** `ward: str`
- ***property*** `address_no: str`
- ***property*** `street: str`
- ***property*** `support_disability: bool`
- ***property*** `status: str`
- ***property*** `latitude: float`
  property `longtitude: float`
  Coordinates of the bus stop in WGS-84 coordinate system.
- ***property*** `coord`
  Coordinates of the bus stop in VN-2000 coordinate system.
- ***property*** `search: list[str]`
  List of tokens that can be used to search for the bus stop
- ***property*** `routes: list[str]`
  List of route names crossing the bus stop

ⓑ | Methods

- `to_string() -> str`
  Display information of a `Stop` in a string format.
  This function is called by `__repr__()`.
- ***static*** `from_dict(obj: dict) -> Stop`
  Create `Stop` from a Python dictionary.
- `to_dict() -> dict`
  Convert `Stop` to a Python dictionary.
- ***static*** `from_json(file: str) -> Stop`
  Create `Stop` from JSON file. The function loads a JSON file into a Python dictionary, then call `from_dict()`.

- to_json(file: str) -> None
  Export Stop information to JSON file. The function calls to_dict() to convert Stop to a dictionary, then dumps it into a JSON file.

```
>>> from stop import Stop
>>> stop = Stop.from_json('stop.json')
>>> stop.to_string()
================[Nguyen Van Linh]=================
| StopID:              7182                       |
| Code:                Q7 BD2                     |
| Name:                Nguyen Van Linh            |
| Type:                O son                      |
| Zone:                Quan 7                     |
| Ward:                Phuong Tan Phong           |
| Address No.:         R1-49                      |
| Street:              Bui Bang Doan              |
| Support Disability:  True                       |
| Status:              Dang khai thac             |
| Lng:                 106.708499                 |
| Lat:                 10.729471                  |
| Search tokens:       NVL, R, BBD                |
| Routes:              D2                         |
==================================================
>>> stop.routes.append('D3')
>>> stop.to_json('out.json')
>>> stop2 = Stop.from_json('out.json')
>>> stop2
================[Nguyen Van Linh]=================
| StopID:              7182                       |
| Code:                Q7 BD2                     |
| Name:                Nguyen Van Linh            |
| Type:                O son                      |
| Zone:                Quan 7                     |
| Ward:                Phuong Tan Phong           |
| Address No.:         R1-49                      |
| Street:              Bui Bang Doan              |
| Support Disability:  True                       |
| Status:              Dang khai thac             |
| Lng:                 106.708499                 |
| Lat:                 10.729471                  |
| Search tokens:       NVL, R, BBD                |
| Routes:              D2 -> D3                   |
==================================================
```

Listing 1.1: Elements of a Bus Network: Example of a Stop object

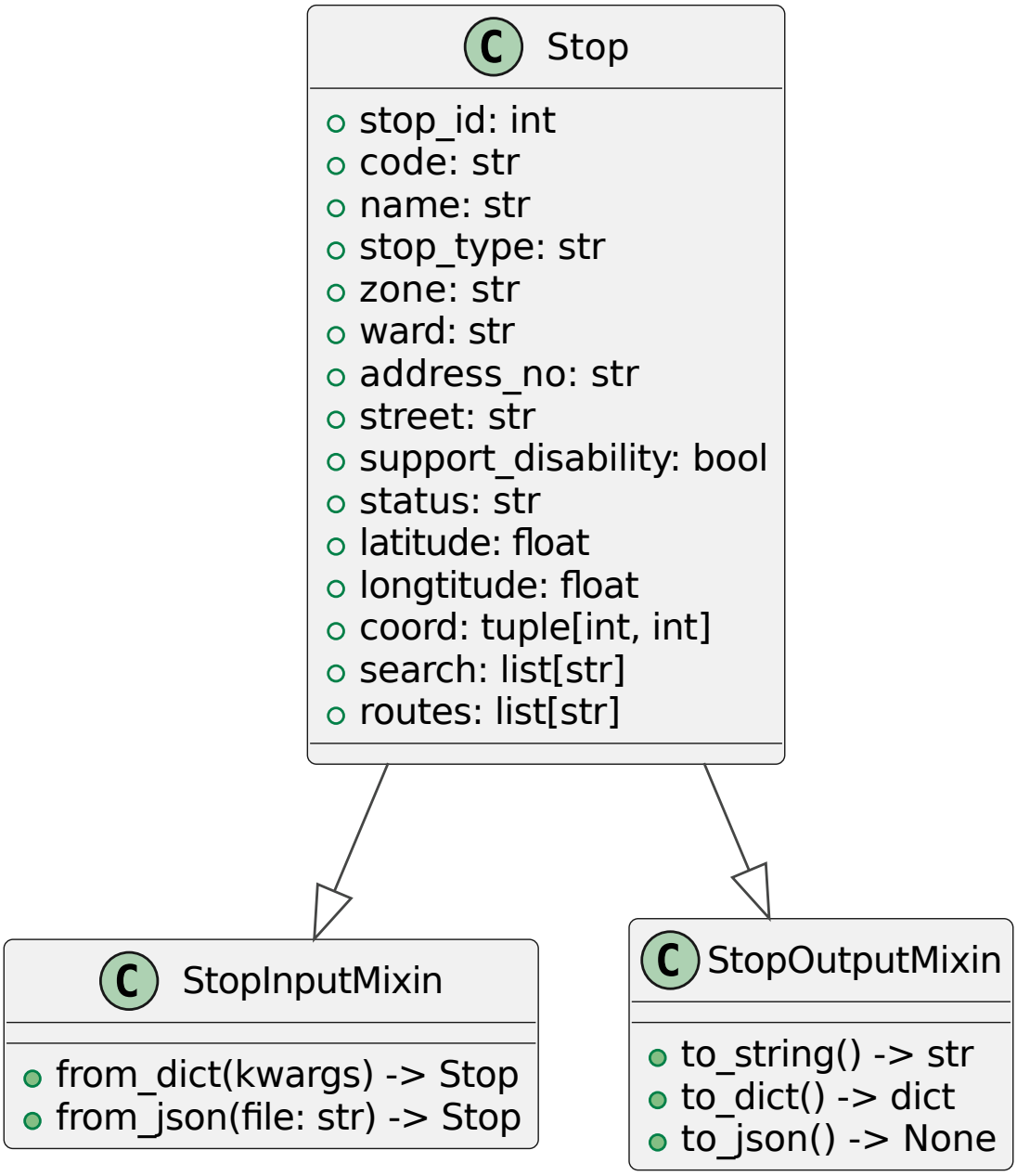

Figure 1.2: Elements of a Bus Network: Stop class diagram

1.2 | Variant (the variant module)

*Variant*s represent bus routes in a specific direction. A bus route has two variants, outbound and inbound. Some bus routes may only have one variant.

ⓐ | Properties

- ***property*** `route_id: int`
  property `route_var_id: int`
  property `route_ids: tuple[int, int]`
  IDs of a `Variant`.
  A route has two variants, each shares the same `route_id` but different `route_var_id`. `route_ids` is a tuple of `route_id` and `route_var_id`.
- ***property*** `number: str`
  Number of the bus variant. Must be of `str` type to store values such as `61-1, D2`.
- ***property*** `name: str`
  property `short_name: str`
  Name/Short name of the bus variant.
- ***property*** `start_stop: str`
  property `end_stop: str`
  Starting and ending stops of a variant
- ***property*** `distance: float`
  property `running_time: float`
  Distance and running time of the bus variant.

ⓑ | Methods

- `to_string() -> str`
  Display information of a `Variant` in a string format.
  This function is called by `__repr__()`.
- ***static*** `from_dict(obj: dict) -> Variant`
  Create `Variant` from a Python dictionary.
- `to_dict() -> dict`
  Convert `Variant` to a Python dictionary.
- ***static*** `from_json(file: str) -> Variant`
  Create `Variant` from JSON file. The function loads a JSON file into a Python dictionary, then call `from_dict()`.
- `to_json(file: str) -> None`
  Export `Variant` information to JSON file. The function calls `to_dict()` to convert `Variant` to a dictionary, then dumps it into a JSON file.

```
>>> from variant import Variant
>>> var = Variant.from_json('var.json')
>>> var
========[Route D2, Paris Baguette -> Cressent mall]========
| RouteNo:        D2                                       |
| StartStop:      Paris Baguette                           |
| EndStop:        Cressent mall                            |
| Name:           Luot di                                  |
| ShortName:      Luot di                                  |
| Distance:       3677.0000000000005                       |
| RunningTime:    14                                       |
| RouteIds:       (212, 1)                                 |
============================================================
>>> var.route_id
212
>>> var.route_var_id
1
>>> var.route_ids
(212, 1)
>>> var.to_dict()
{
    'RouteNo': 'D2',
    'StartStop': 'Paris Baguette',
    'EndStop': 'Cressent mall',
    'Name': 'Luot di'
    'ShortName': 'Luot di'
    'Distance': 3677.0000000000005,
    'RunningTime': 14,
    'RouteIds': (212, 1)
}
```

Listing 1.1: Elements of a Bus Network: Example of a `Variant` object

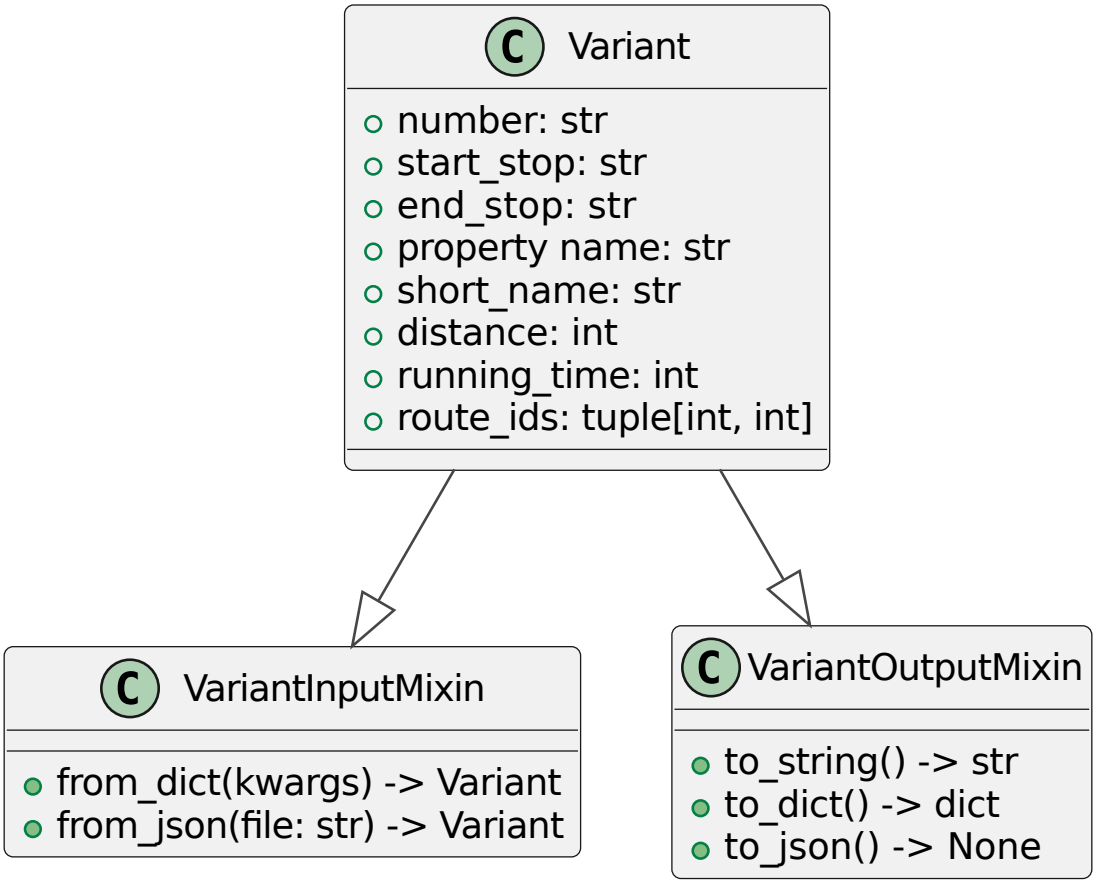

Figure 1.3: Elements of a Bus Network: Variant class diagram

1.3 | Path (the path module)

*Path*s represent the specific path of a bus variant in the form of a LineString.

ⓐ | Properties

- ***property*** `route_id: int`
  property `route_var_id: int`
  property `route_ids: tuple[int, int]`
  IDs of the route that the `Path` represents.
  A route has two variants, each shares the same `route_id` but different `route_var_id`. `route_ids` is a tuple of `route_id` and `route_var_id`.
- ***property*** `coords: list[tuple[float, float]]` Coordinates of a `LineString` representing the path of the bus variant.

ⓑ | Methods

- `polysides() -> Iterable[tuple[tuple[float, float], tuple[float, float]]]`
  Return an `Iterable` of tuples of coordinates, representing each segments of the `LineString`.
- `to_string() -> str`
  Display information of a `Path` in a string format.
  This function is called by `__repr__()`.
- ***static*** `from_dict(obj: dict) -> Path`
  Create `Path` from a Python dictionary.
- `to_dict() -> dict`
  Convert `Path` to a Python dictionary.
- ***static*** `from_json(file: str) -> Path`
  Create `Path` from JSON file. The function loads a JSON file into a Python dictionary, then call `from_dict()`.
- `to_json(file: str) -> None`
  Export `Path` information to JSON file. The function calls `to_dict()` to convert `Path` to a dictionary, then dumps it into a JSON file.

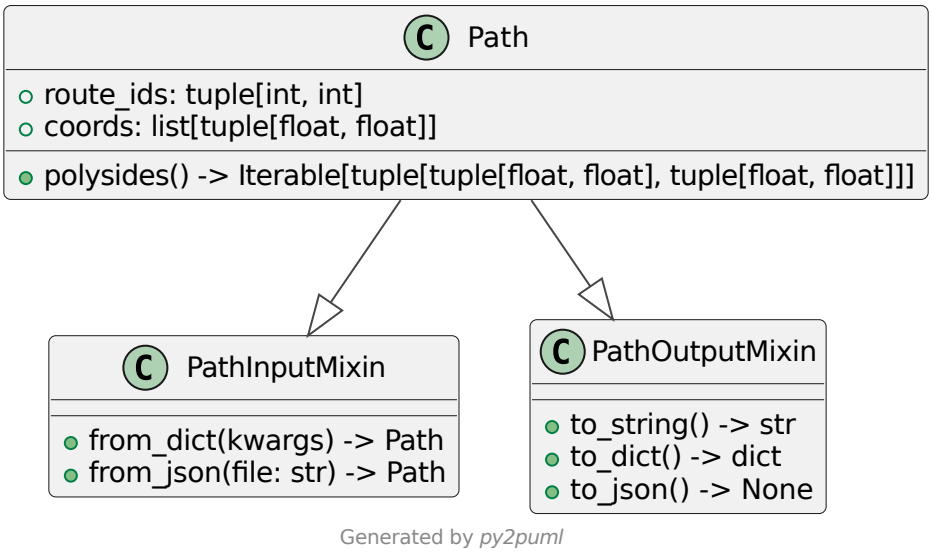

Figure 1.4: Elements of a Bus Network: Path class diagram

```
>>> from path import Path
>>> import math
>>> path = Path.from_json('path.json')
>>> path.route_ids
(212, 1)
>>> len(path.coords)
30
>>> sum(math.dist(p1, p2) for (p1, p2) in path.polysides())
3664.914478222333
```

Listing 1.1: Elements of a Bus Network: Example of a `Path` object

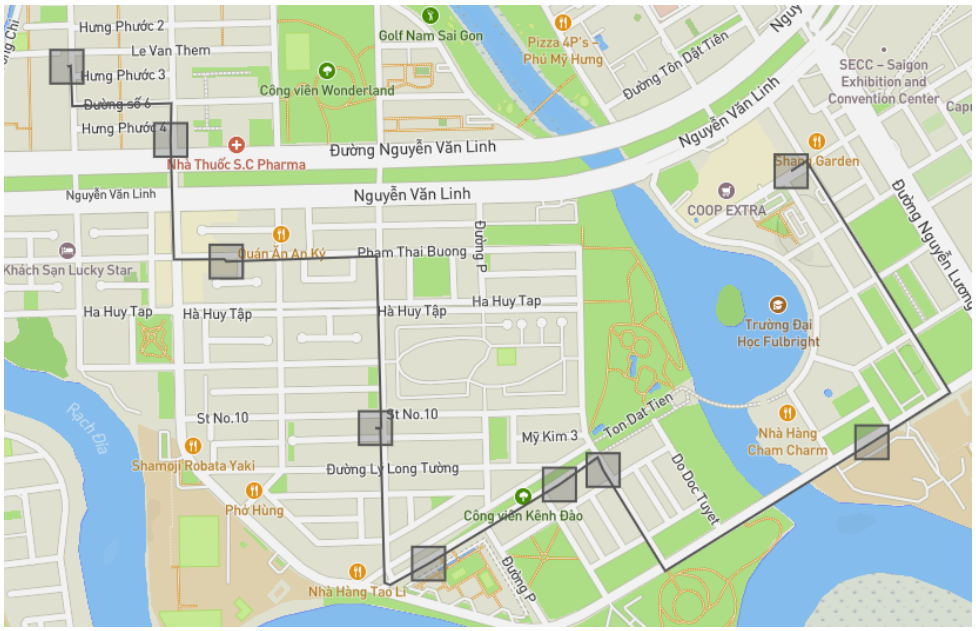

Figure 1.5: Elements of a Bus Network: The variant from *Paris Baguette* to *Cressent mall* has a total distance of approx. 3665km, whilst the distance in database is 3677km. Their relative difference is 0.33%.

Chapter 2

Querying List of Objects

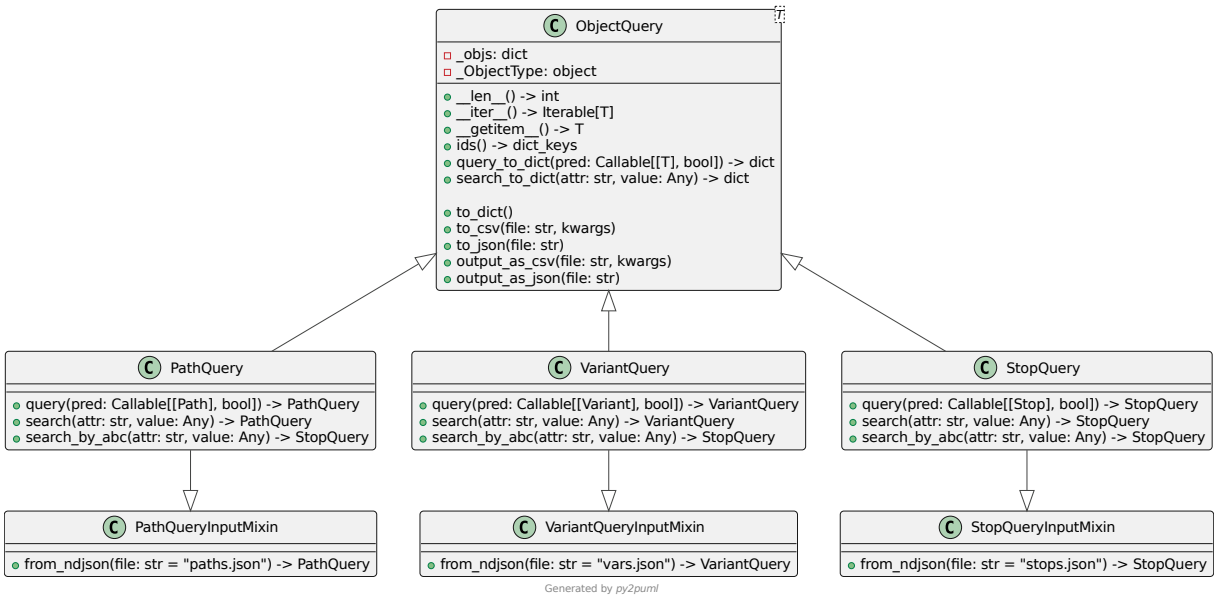

Figure 2.1: Querying List of Objects: ObjectQuery base class and derived classes for specific querying data types

2.1 | A generic object querying type: `ObjectQuery`

ⓐ | Properties

- `_objs: dict`
  Items in query are stored in a dictionary with their associated IDs.
- `_ObjectType: object`
  Type of items.
- ***property*** `ids: list[int]`
  Return ids of items.
- ***property*** `values: list[Any]`
  Return values of items.

ⓑ | Methods

- `query_to_dict(pred: Callable) -> dict`
  Query all items satisfying the predicate `pred`, return as a dictionary.
- `search_to_dict(attr: str, value: Any) -> dict`
  Search for all items with attributes matching `value`, return as a dictionary.

```
def search_to_dict(self, attr: str, value: Any) -> dict:
    if not hasattr(self._ObjectType, attr):
        raise AttributeError

    return self.query_to_dict(lambda obj: getattr(obj, attr) == value)
```

- `to_dict() -> dict`
  Returns internal state `_objs`.
- `to_json(file: str) -> None`
  Export `ObjectQuery` information to JSON file. The function dumps the dictionary `_objs` into a JSON file.
- `to_csv(file: str) -> None`
  Export `ObjectQuery` information to CSV file. The function converts the dictionary `_objs` into a `pandas` table before exporting to CSV.
- `output_as_json = to_json`
  `output_as_csv = to_csv`
  Aliases for `to_json` and `to_csv` functions.

2.2 | Inheritence from `ObjectQuery` for specific object types

ⓐ | StopQuery

```
class StopQuery(ObjectQuery, StopQueryInputMixin):
    def __init__(self, objs):
        super.__init__(objs=objs, ObjectType=Stop)

    def query(self, pred: Callable[[Stop], bool]):
        return StopQuery(self.query_to_dict(pred))

    def search(self, attr: str, value: Any):
        return StopQuery(self.search_to_dict(attr, value))

    # aliases
    search_by_abc = search
```

Listing 2.1: Querying List of Objects: `StopQuery` class

ⓑ | VariantQuery

```
class VariantQuery(ObjectQuery, VariantQueryInputMixin):
    def __init__(self, objs):
        super.__init__(objs=objs, ObjectType=Variant)

    def query(self, pred: Callable[[Variant], bool]):
        return VariantQuery(self.query_to_dict(pred))

    def search(self, attr: str, value: Any):
        return VariantQuery(self.search_to_dict(attr, value))

    # aliases
    search_by_abc = search
```

Listing 2.2: Querying List of Objects: `VariantQuery` class

ⓒ | PathQuery

```
class PathQuery(ObjectQuery, PathQueryInputMixin):
    def __init__(self, objs):
        super.__init__(objs=objs, ObjectType=Path)

    def query(self, pred: Callable[[Path], bool]):
        return PathQuery(self.query_to_dict(pred))

    def search(self, attr: str, value: Any):
        return PathQuery(self.search_to_dict(attr, value))

    # aliases
    search_by_abc = search
```

Listing 2.3: Querying List of Objects: `PathQuery` class

2.3 | Example use of ObjectQuery-derived class

```
>>> stops = StopQuery.from_ndjson()
>>> variants = VariantQuery.from_ndjson()
>>> paths = PathQuery.from_ndjson()
>>> print('There are {} stops, {} variants and {} paths.'\
...       .format(len(stops), len(variants), len(paths)))
There are 4397 stops, 297 variants and 297 paths.
>>> stop_id = 1234
>>> print(stops.search('stop_id', stop_id).values)
[==================[Nga 3 Cu Cai]==================
| StopID:              1234                       |
| Code:                HHM 053                    |
| Name:                Nga 3 Cu Cai               |
| Type:                Nha cho                    |
| Zone:                Huyen Hoc Mon              |
| Ward:                Xa Xuan Thoi Dong          |
| Address No.:         43/1 (ke 33/5A), 7/2A      |
| Street:              Quoc lo 22                 |
| Support Disability:  True                       |
| Status:              Dang khai thac             |
| Lng:                 106.603324                 |
| Lat:                 10.860602                  |
| Search tokens:       N3CC, 43/1(33/5A),7/2A, Ql22
| Routes:              122 -> 13 -> 62-5 -> 70-3 -> 74 -> 85 -> 94
==================================================]
>>> variants.query(lambda var: var.distance <= 1680)
===========[Route HS-73, Rung Sac -> tieu Hoc An Nghia]===========
| RouteNo:           HS-73                                      |
| StartStop:         Rung Sac                                   |
| EndStop:           tieu Hoc An Nghia                          |
| Name:              Luot di                                    |
| ShortName:         Truong Tieu Hoc An Nghia                   |
| Distance:          1680.0                                     |
| RunningTime:       20                                         |
| RouteIds:          (293, 1)                                   |
==============================================================
===========[Route HS-73, tieu Hoc An Nghia -> Rung Sac]===========
| RouteNo:           HS-73                                      |
| StartStop:         tieu Hoc An Nghia                          |
| EndStop:           Rung Sac                                   |
| Name:              Luot ve                                    |
| ShortName:         Rung Sac                                   |
| Distance:          1616.0                                     |
| RunningTime:       20                                         |
| RouteIds:          (293, 2)                                   |
==============================================================
```

Bibliography