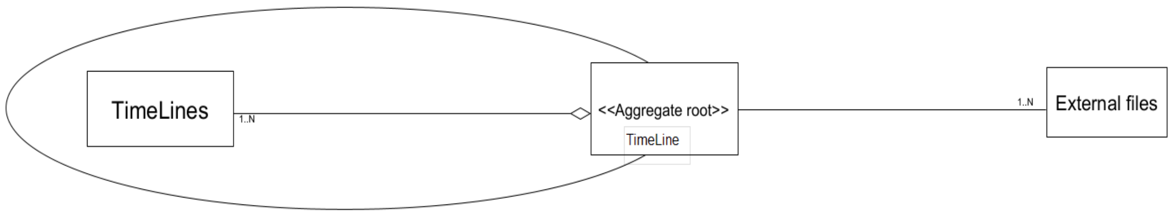
Lab 8 aggregateOne

The two aggregate pattern is evident in the models.py file. this file is in the directory.
coronavirus-tracker-api/app/**models.py**/
Project chosen: project 1 Coronavirus API

Two of the aggregate are present in these parts:

Aggregate 1: this aggregate pattern is present in the codestarAI/**coronavirus-tracker-api**/app/**models.py**/



File. This aggregate is a formation of three different objects that are collocated in a way such in which an aggregation forms. This aggregation works based on the root object TimeLine that which is called by another object TimeLines. This root object can be accessed by the external files which is any other file in the Coronavirus project outside this directory. The circle encompasses the aggregate boundary in which the aggregate is holds place. Any object inside this circle can be accessed by the root but cannot be accessed directly outside the boundry from another object. An external object from this circle must access this internal object via referencing the aggregate root. The cardinality must remain 1..N for both the TimeLine and External file for the aggregate boundary/aggregate to hold place. This reasoning also applies for aggregate 2.

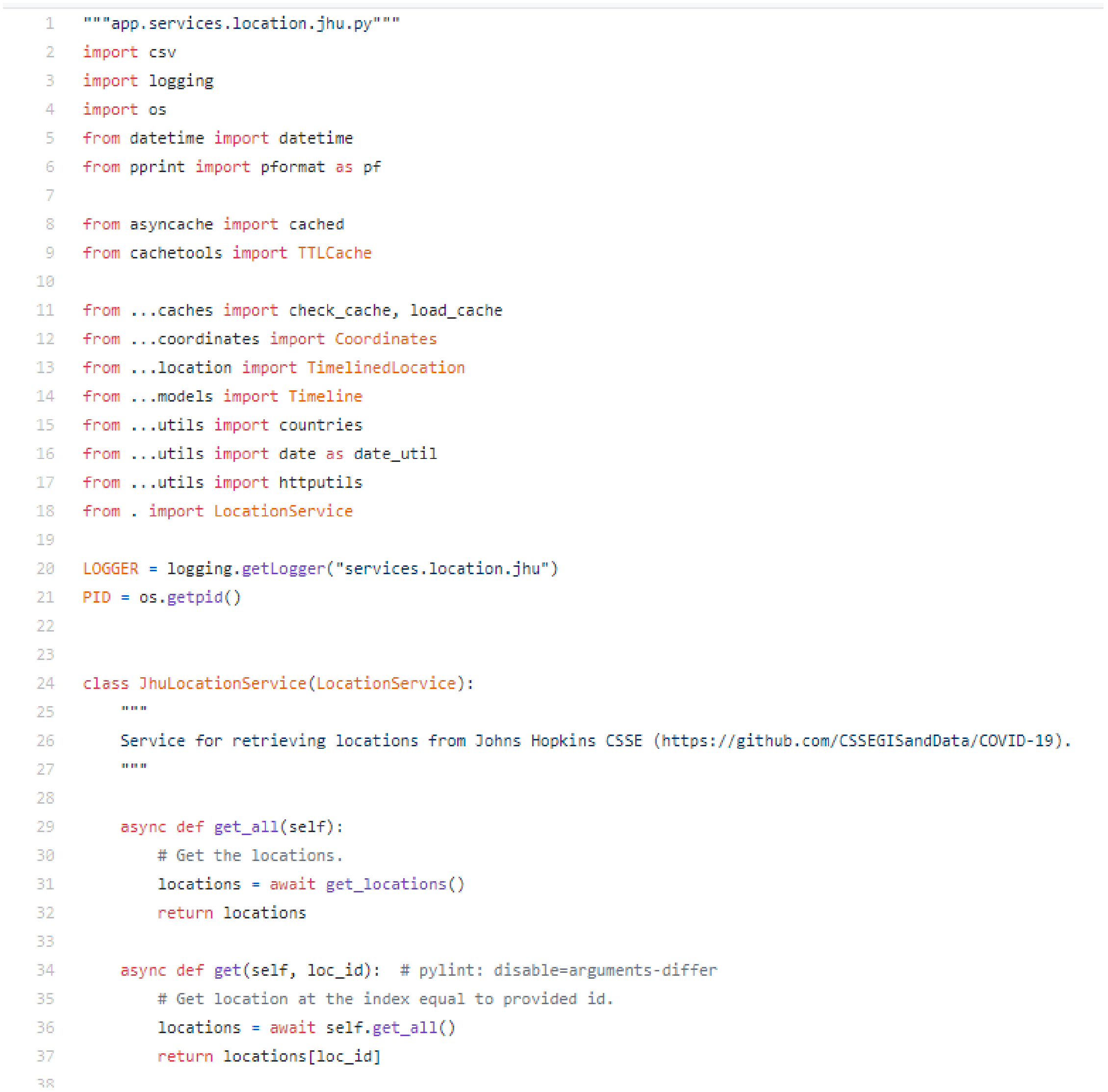
Aggregate 1 code:

```
class Timeline:
  def __init__(self, timeline):
   self.timeline = timeline

class Timelines:
  def __init__(self, confirmed, deaths, recovered):
   self.confirmed = confirmed
   self.deaths = deaths
   self.recovered = recovered

  if __name__ == "__main__":
   confirm = Timeline({'Sat':10, 'Wed':20})
   death = Timeline({'Sat':10, 'Wed':20})
   recovered = Timeline({'Sat':10, 'Wed':20})
   timelines_1 = Timelines(confirm, death, recovered)
```

This code describes the aggregation process. The class TimeLines creates an instance of the class TimeLine and are associated with one another be can still exists without the other. By the confirm = TimeLine() method and death = TimeLine() plus recovered = TimeLine() method calling the TimeLine class these methods make a direct reference to the root object, which it is associated with but can still exist separately without. As for the external file object, this object makes references with the aggregated root by this entire section of the Aggregate 1 code being imported in another file. Using this aggregates functionality. For example this is demonstrated here:



```
 1  """app.services.location.jhu.py"""
 2  import csv
 3  import logging
 4  import os
 5  from datetime import datetime
 6  from pprint import pformat as pf
 7
 8  from asyncache import cached
 9  from cachetools import TTLCache
10
11  from ...caches import check_cache, load_cache
12  from ...coordinates import Coordinates
13  from ...location import TimelinedLocation
14  from ...models import Timeline
15  from ...utils import countries
16  from ...utils import date as date_util
17  from ...utils import httputils
18  from . import LocationService
19
20  LOGGER = logging.getLogger("services.location.jhu")
21  PID = os.getpid()
22
23
24  class JhuLocationService(LocationService):
25      """
26      Service for retrieving locations from Johns Hopkins CSSE (https://github.com/CSSEGISandData/COVID-19).
27      """
28
29      async def get_all(self):
30          # Get the locations.
31          locations = await get_locations()
32          return locations
33
34      async def get(self, loc_id):  # pylint: disable=arguments-differ
35          # Get location at the index equal to provided id.
36          locations = await self.get_all()
37          return locations[loc_id]
38
```

If you look at line 14 this line of code imports this entire method TimeLine which is coming from the models directory allowing this file to use aspects/functionality of the timeLine code. This file is present in the

coronavirus-tracker-api/blob/app/services/location/jhu.py

File.