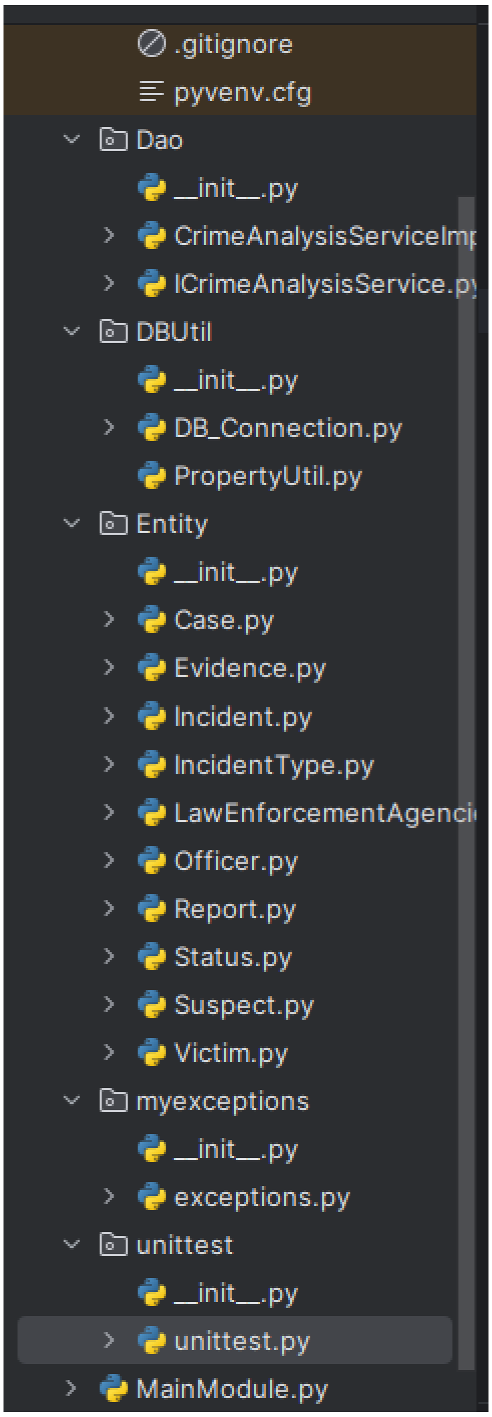
CASE STUDY

💡 Case Study name : Crime Analysis and Reporting System

💡 Pradum singh

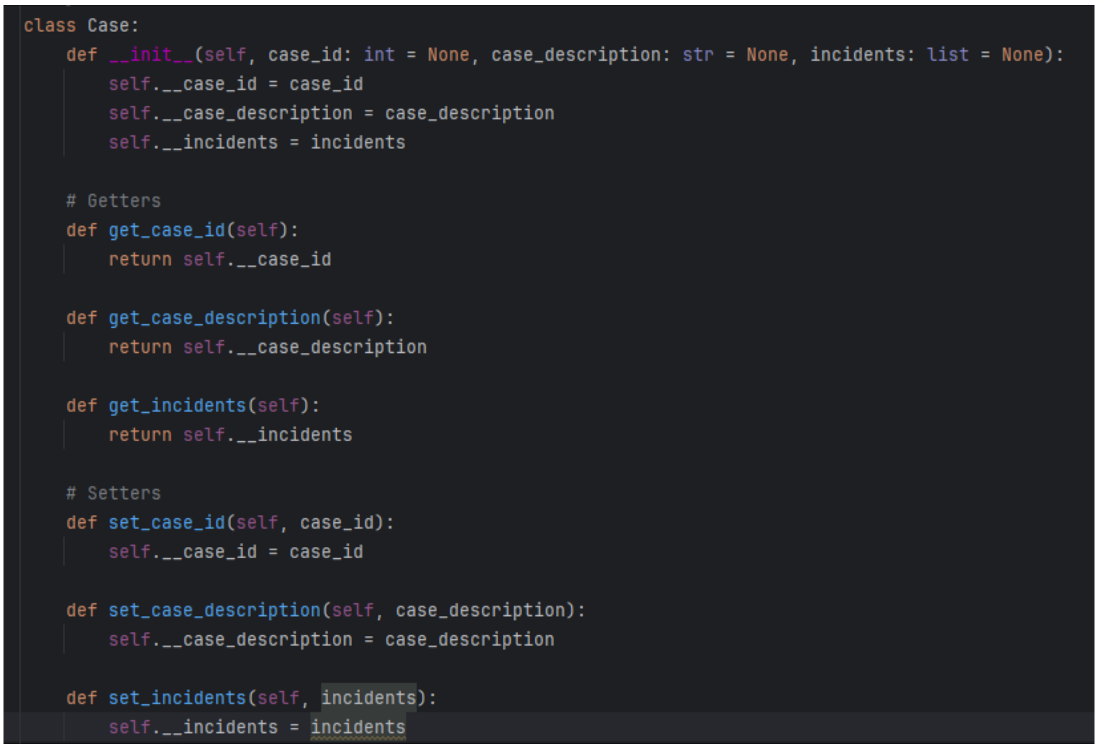
- ***Directory Structure***



```
.gitignore
pyvenv.cfg
Dao
    __init__.py
    CrimeAnalysisServiceImp
    ICrimeAnalysisService.py
DBUtil
    __init__.py
    DB_Connection.py
    PropertyUtil.py
Entity
    __init__.py
    Case.py
    Evidence.py
    Incident.py
    IncidentType.py
    LawEnforcementAgenci
    Officer.py
    Report.py
    Status.py
    Suspect.py
    Victim.py
myexceptions
    __init__.py
    exceptions.py
unittest
    __init__.py
    unittest.py
MainModule.py
```

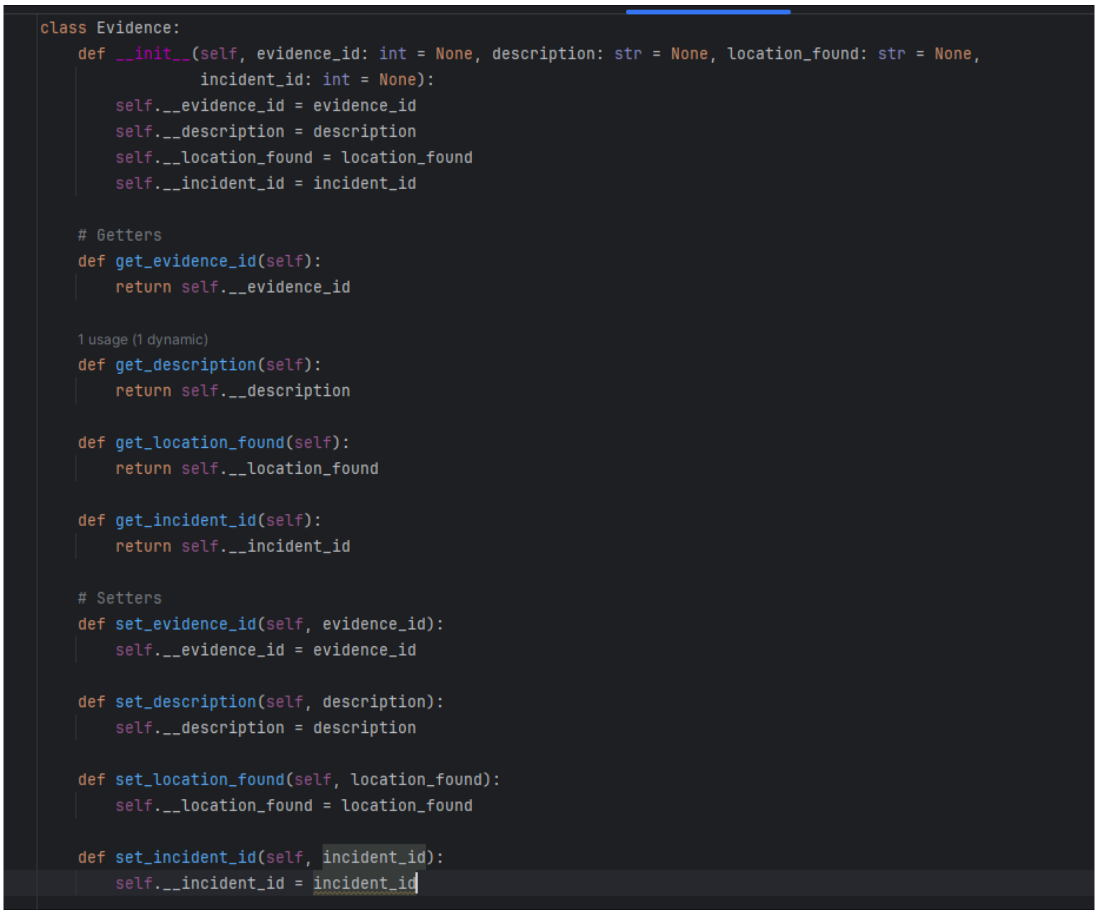
- ◦

- ***Entity***
    - ***Case***



```python
class Case:
    def __init__(self, case_id: int = None, case_description: str = None, incidents: list = None):
        self.__case_id = case_id
        self.__case_description = case_description
        self.__incidents = incidents

    # Getters
    def get_case_id(self):
        return self.__case_id

    def get_case_description(self):
        return self.__case_description

    def get_incidents(self):
        return self.__incidents

    # Setters
    def set_case_id(self, case_id):
        self.__case_id = case_id

    def set_case_description(self, case_description):
        self.__case_description = case_description

    def set_incidents(self, incidents):
        self.__incidents = incidents
```

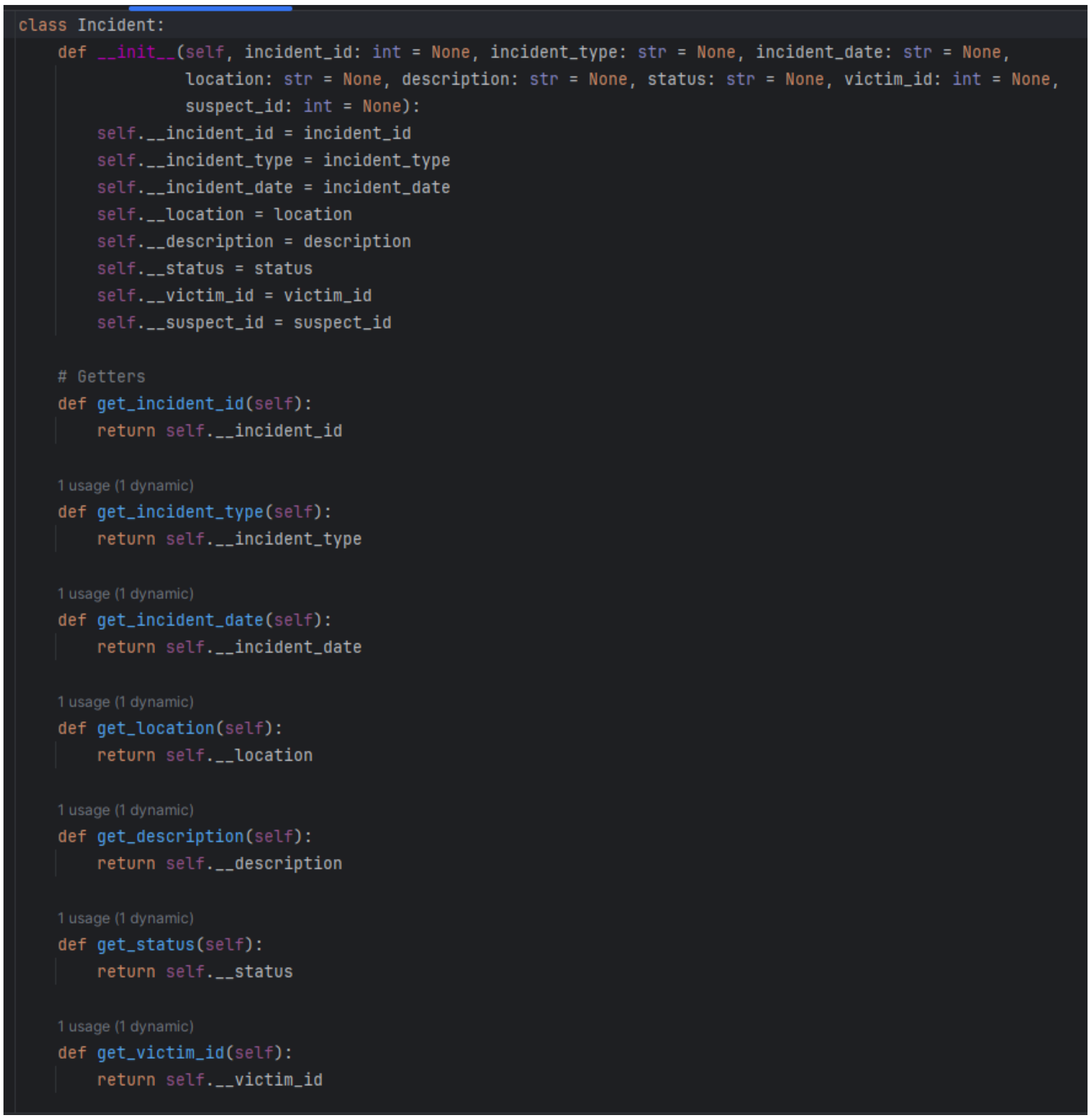
- ***<u>Evidence</u>***



```python
class Evidence:
    def __init__(self, evidence_id: int = None, description: str = None, location_found: str = None,
                 incident_id: int = None):
        self.__evidence_id = evidence_id
        self.__description = description
        self.__location_found = location_found
        self.__incident_id = incident_id

    # Getters
    def get_evidence_id(self):
        return self.__evidence_id

    1 usage (1 dynamic)
    def get_description(self):
        return self.__description

    def get_location_found(self):
        return self.__location_found

    def get_incident_id(self):
        return self.__incident_id

    # Setters
    def set_evidence_id(self, evidence_id):
        self.__evidence_id = evidence_id

    def set_description(self, description):
        self.__description = description

    def set_location_found(self, location_found):
        self.__location_found = location_found

    def set_incident_id(self, incident_id):
        self.__incident_id = incident_id
```

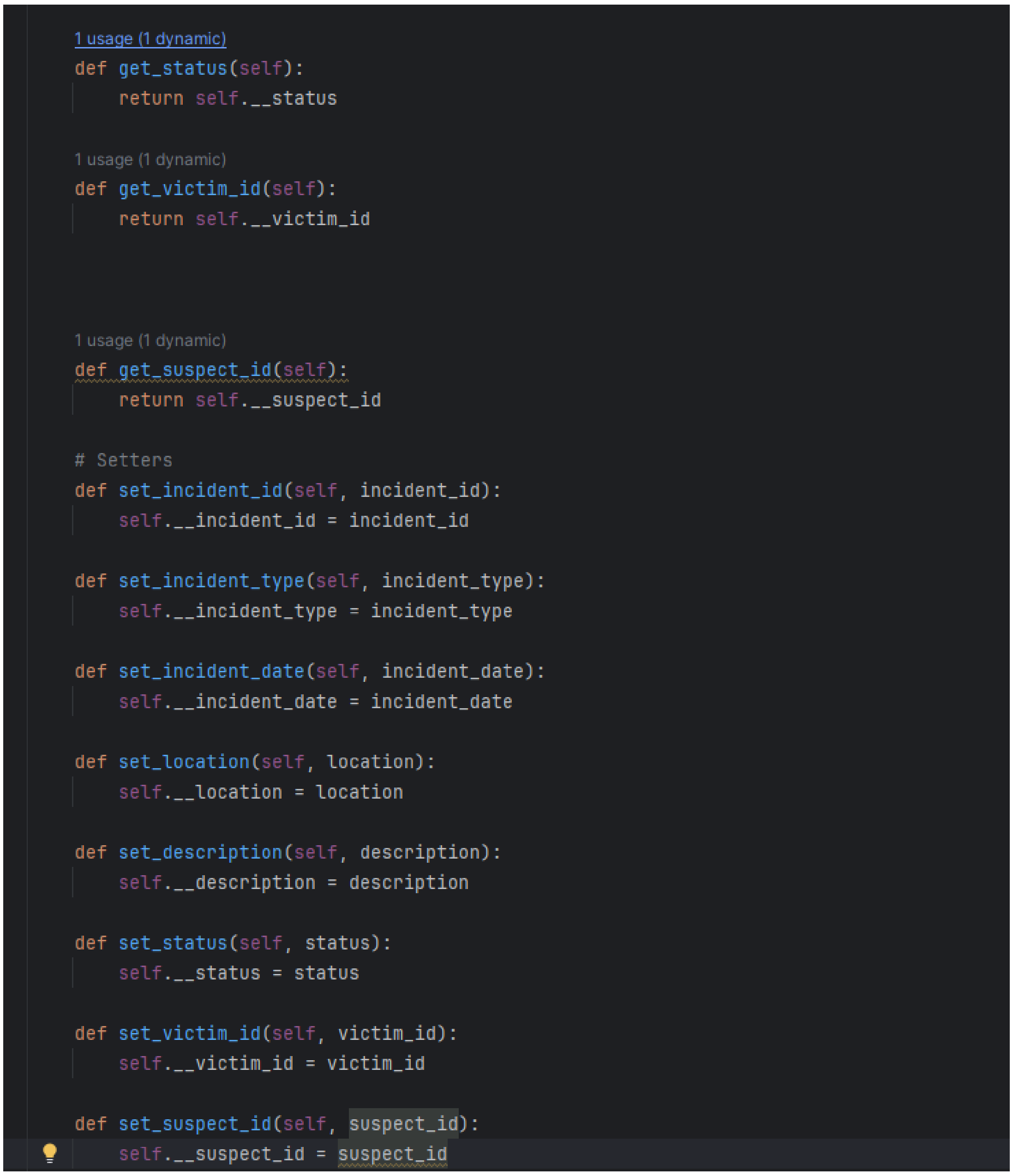
- ***<u>Incident</u>***

▪



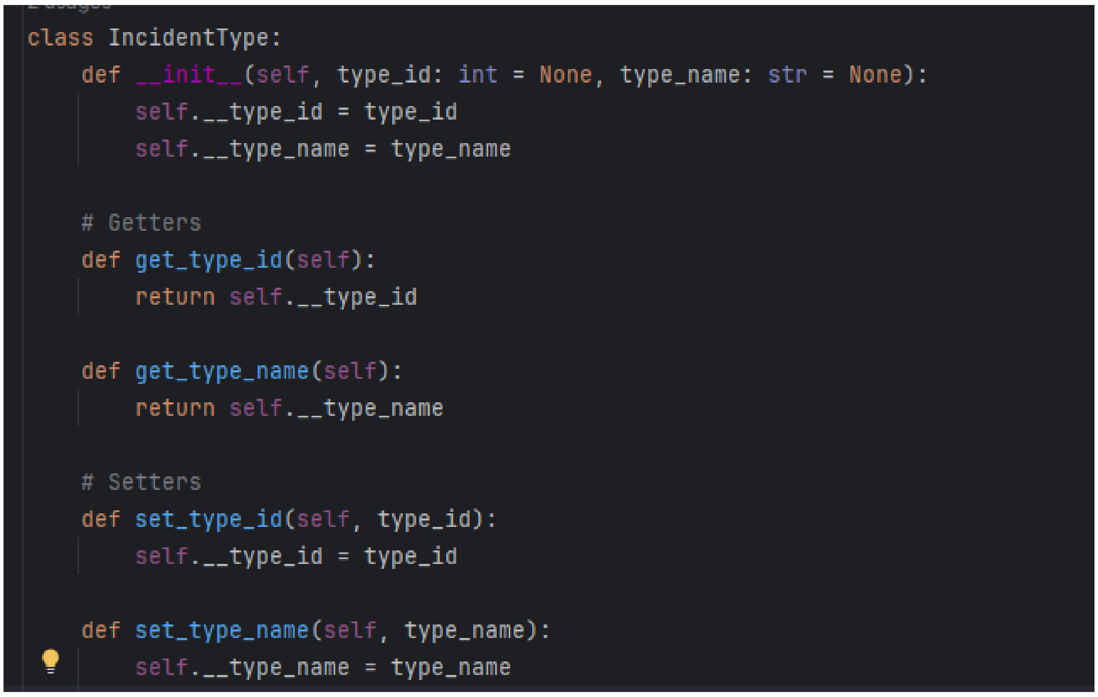
```
class Incident:
    def __init__(self, incident_id: int = None, incident_type: str = None, incident_date: str = None,
                 location: str = None, description: str = None, status: str = None, victim_id: int = None,
                 suspect_id: int = None):
        self.__incident_id = incident_id
        self.__incident_type = incident_type
        self.__incident_date = incident_date
        self.__location = location
        self.__description = description
        self.__status = status
        self.__victim_id = victim_id
        self.__suspect_id = suspect_id

    # Getters
    def get_incident_id(self):
        return self.__incident_id

    1 usage (1 dynamic)
    def get_incident_type(self):
        return self.__incident_type

    1 usage (1 dynamic)
    def get_incident_date(self):
        return self.__incident_date

    1 usage (1 dynamic)
    def get_location(self):
        return self.__location

    1 usage (1 dynamic)
    def get_description(self):
        return self.__description

    1 usage (1 dynamic)
    def get_status(self):
        return self.__status

    1 usage (1 dynamic)
    def get_victim_id(self):
        return self.__victim_id
```

▪ ++



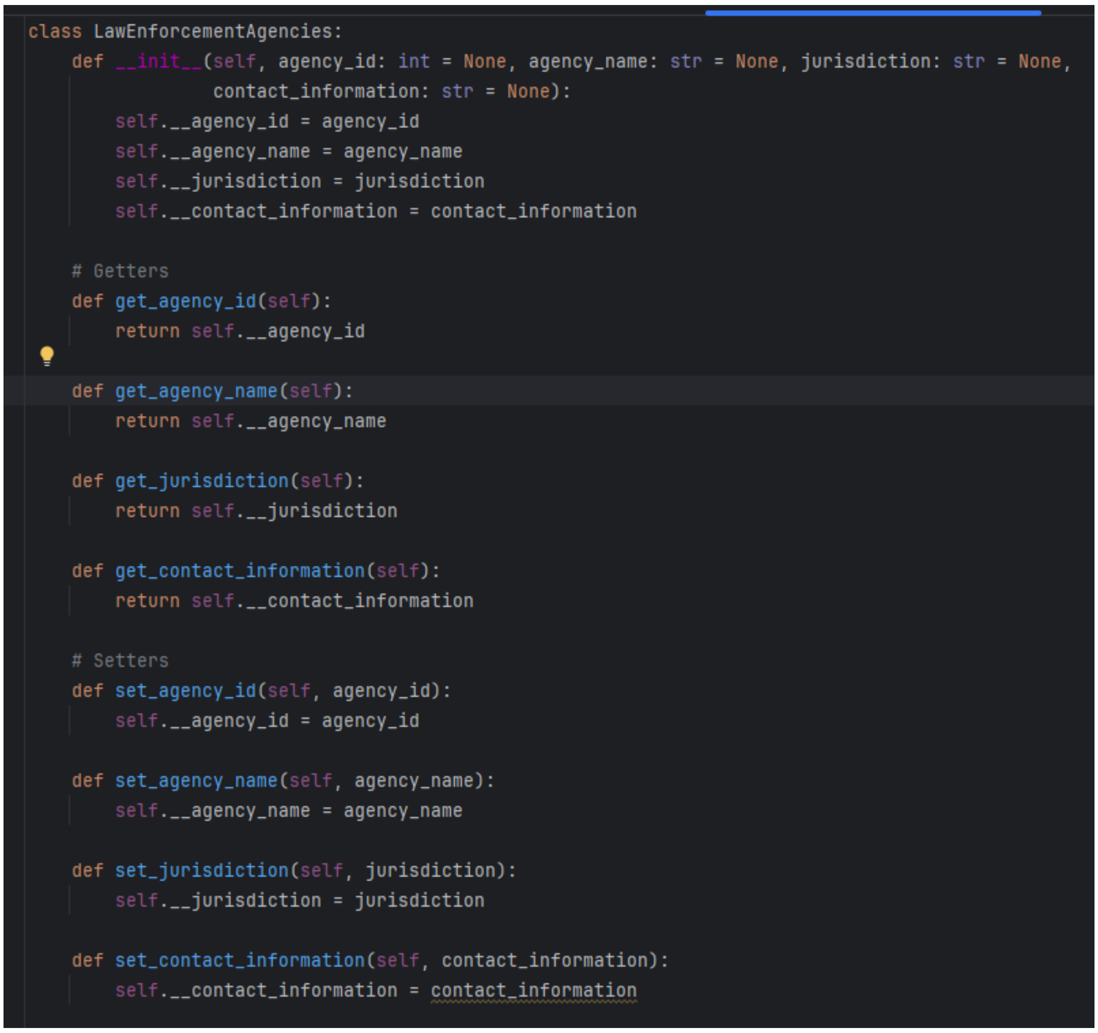
```
1 usage (1 dynamic)
def get_status(self):
    return self.__status

1 usage (1 dynamic)
def get_victim_id(self):
    return self.__victim_id



1 usage (1 dynamic)
def get_suspect_id(self):
    return self.__suspect_id

# Setters
def set_incident_id(self, incident_id):
    self.__incident_id = incident_id

def set_incident_type(self, incident_type):
    self.__incident_type = incident_type

def set_incident_date(self, incident_date):
    self.__incident_date = incident_date

def set_location(self, location):
    self.__location = location

def set_description(self, description):
    self.__description = description

def set_status(self, status):
    self.__status = status

def set_victim_id(self, victim_id):
    self.__victim_id = victim_id

def set_suspect_id(self, suspect_id):
    self.__suspect_id = suspect_id
```

- *IncidentType*



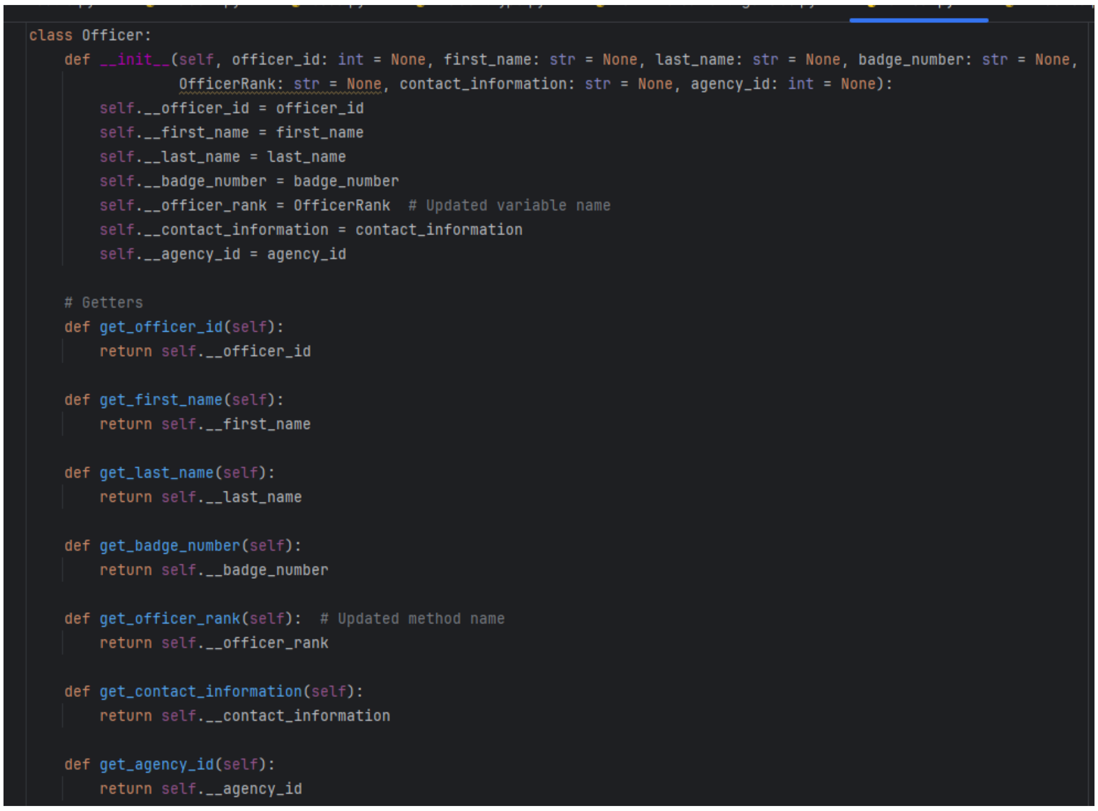
```python
class IncidentType:
    def __init__(self, type_id: int = None, type_name: str = None):
        self.__type_id = type_id
        self.__type_name = type_name

    # Getters
    def get_type_id(self):
        return self.__type_id

    def get_type_name(self):
        return self.__type_name

    # Setters
    def set_type_id(self, type_id):
        self.__type_id = type_id

    def set_type_name(self, type_name):
        self.__type_name = type_name
```

- ***LawEnforcementAgencies***



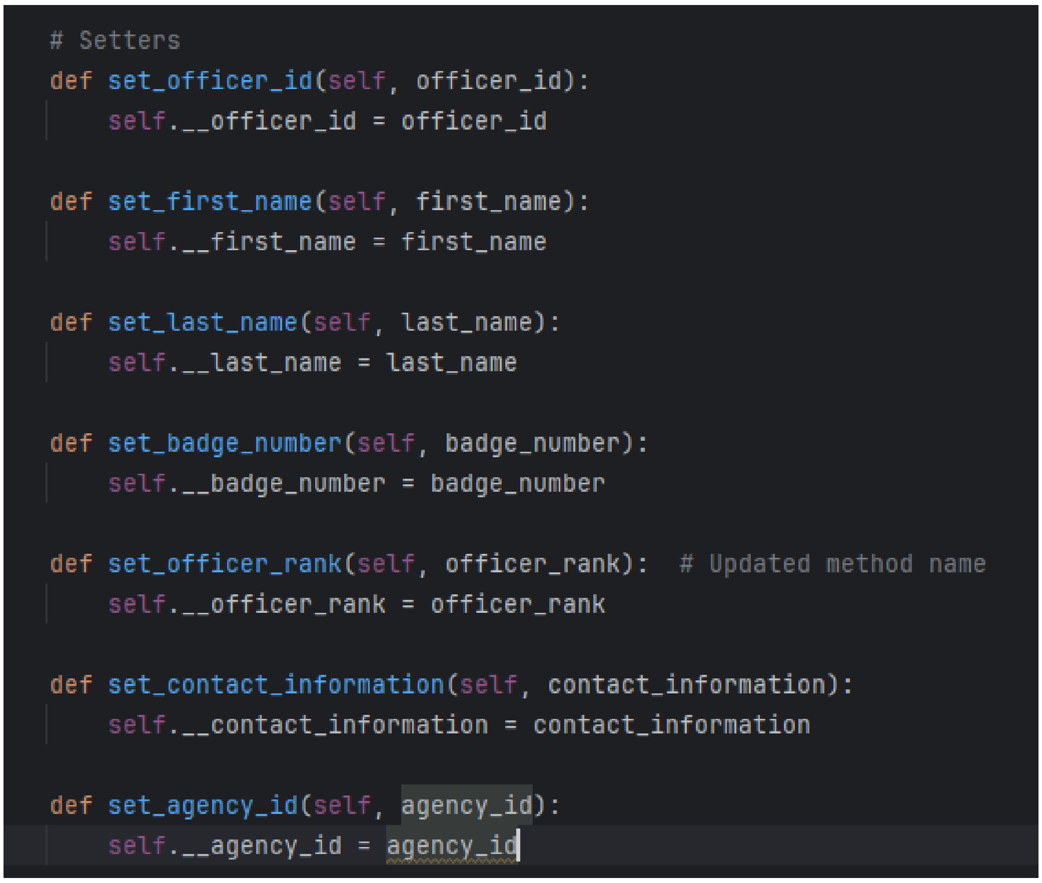
```python
class LawEnforcementAgencies:
    def __init__(self, agency_id: int = None, agency_name: str = None, jurisdiction: str = None,
                 contact_information: str = None):
        self.__agency_id = agency_id
        self.__agency_name = agency_name
        self.__jurisdiction = jurisdiction
        self.__contact_information = contact_information

    # Getters
    def get_agency_id(self):
        return self.__agency_id

    def get_agency_name(self):
        return self.__agency_name

    def get_jurisdiction(self):
        return self.__jurisdiction

    def get_contact_information(self):
        return self.__contact_information

    # Setters
    def set_agency_id(self, agency_id):
        self.__agency_id = agency_id

    def set_agency_name(self, agency_name):
        self.__agency_name = agency_name

    def set_jurisdiction(self, jurisdiction):
        self.__jurisdiction = jurisdiction

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information
```

- ***Officer***



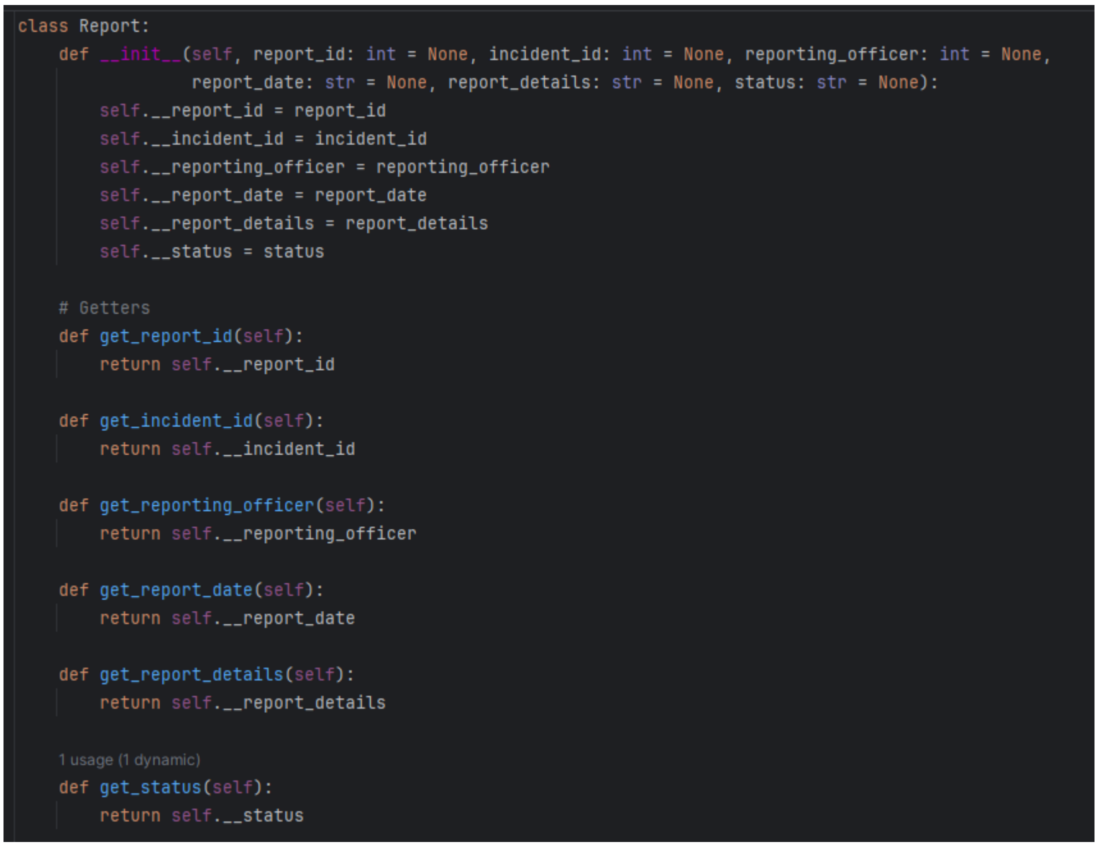
```python
class Officer:
    def __init__(self, officer_id: int = None, first_name: str = None, last_name: str = None, badge_number: str = None,
                 OfficerRank: str = None, contact_information: str = None, agency_id: int = None):
        self.__officer_id = officer_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__badge_number = badge_number
        self.__officer_rank = OfficerRank  # Updated variable name
        self.__contact_information = contact_information
        self.__agency_id = agency_id

    # Getters
    def get_officer_id(self):
        return self.__officer_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_badge_number(self):
        return self.__badge_number

    def get_officer_rank(self):  # Updated method name
        return self.__officer_rank

    def get_contact_information(self):
        return self.__contact_information

    def get_agency_id(self):
        return self.__agency_id
```

- ++



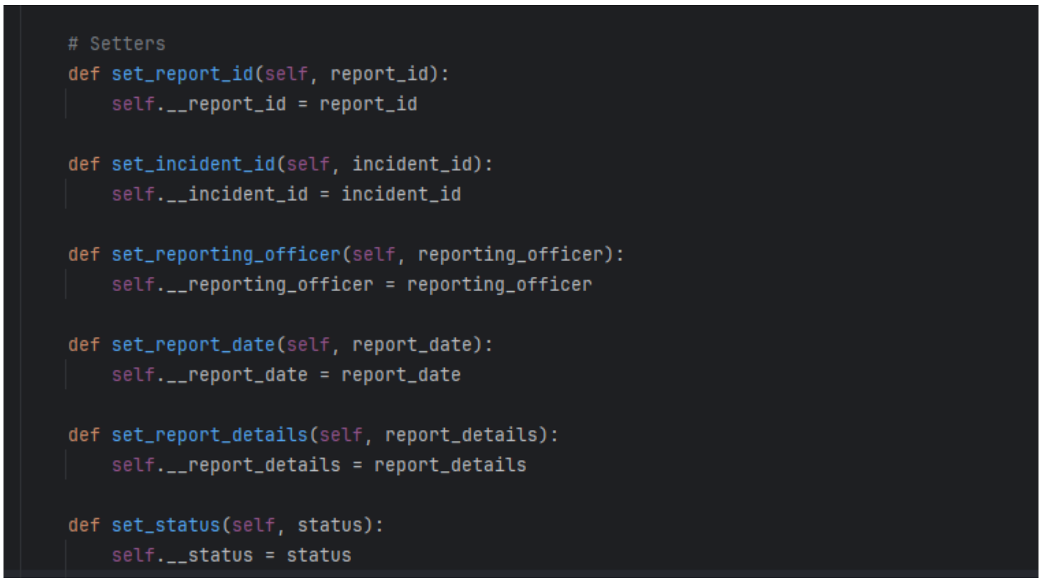
```
# Setters
def set_officer_id(self, officer_id):
    self.__officer_id = officer_id

def set_first_name(self, first_name):
    self.__first_name = first_name

def set_last_name(self, last_name):
    self.__last_name = last_name

def set_badge_number(self, badge_number):
    self.__badge_number = badge_number

def set_officer_rank(self, officer_rank):  # Updated method name
    self.__officer_rank = officer_rank

def set_contact_information(self, contact_information):
    self.__contact_information = contact_information

def set_agency_id(self, agency_id):
    self.__agency_id = agency_id
```

- ***Report***



```
class Report:
    def __init__(self, report_id: int = None, incident_id: int = None, reporting_officer: int = None,
                 report_date: str = None, report_details: str = None, status: str = None):
        self.__report_id = report_id
        self.__incident_id = incident_id
        self.__reporting_officer = reporting_officer
        self.__report_date = report_date
        self.__report_details = report_details
        self.__status = status

    # Getters
    def get_report_id(self):
        return self.__report_id

    def get_incident_id(self):
        return self.__incident_id

    def get_reporting_officer(self):
        return self.__reporting_officer

    def get_report_date(self):
        return self.__report_date

    def get_report_details(self):
        return self.__report_details

    1 usage (1 dynamic)
    def get_status(self):
        return self.__status
```

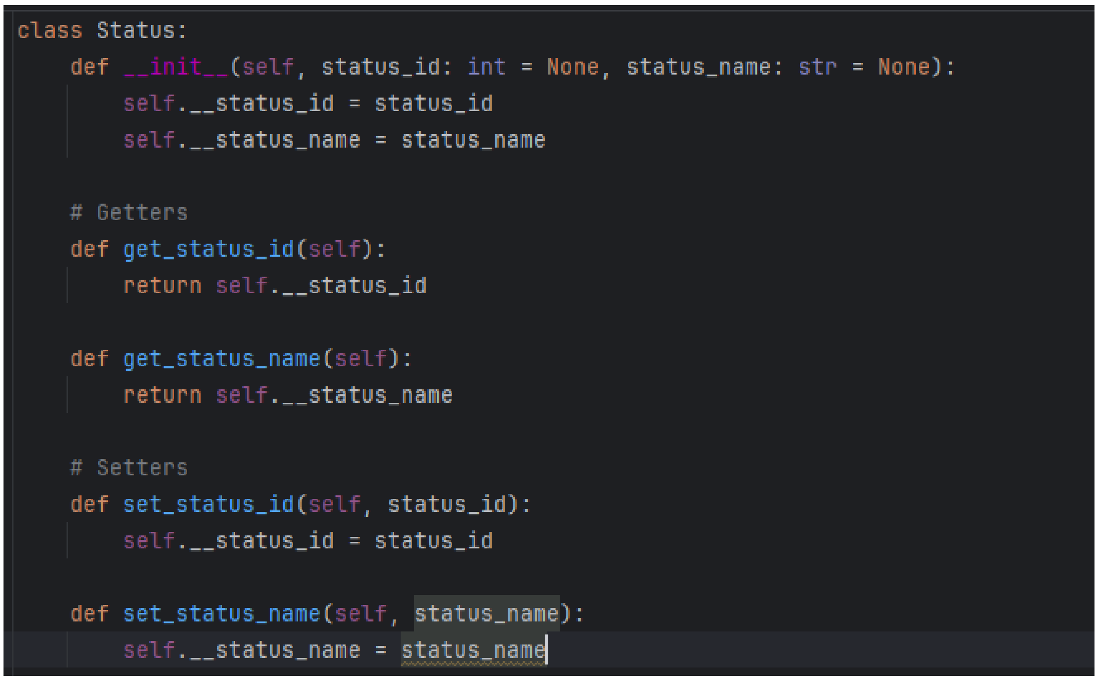
- ++



```python
    # Setters
    def set_report_id(self, report_id):
        self.__report_id = report_id

    def set_incident_id(self, incident_id):
        self.__incident_id = incident_id

    def set_reporting_officer(self, reporting_officer):
        self.__reporting_officer = reporting_officer

    def set_report_date(self, report_date):
        self.__report_date = report_date

    def set_report_details(self, report_details):
        self.__report_details = report_details

    def set_status(self, status):
        self.__status = status
```

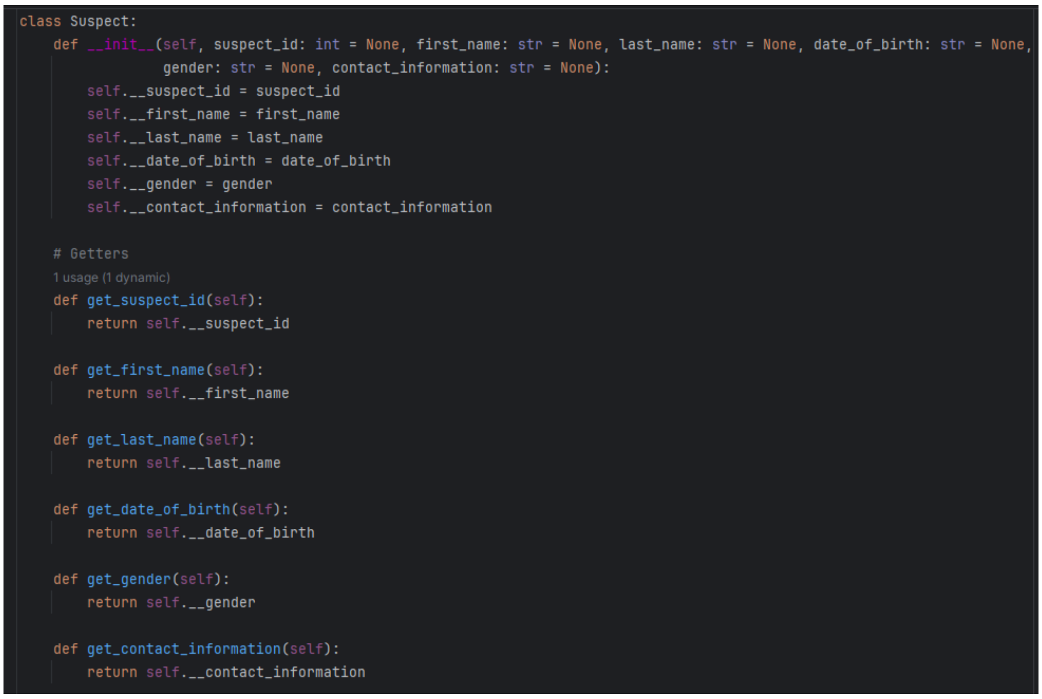
-

- ***Status***



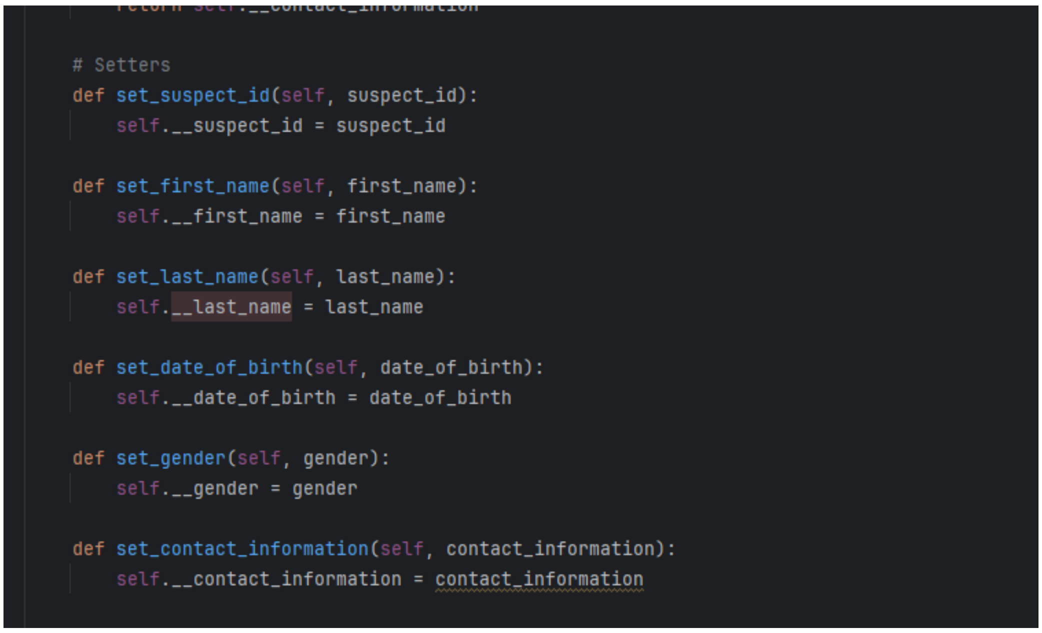
```python
class Status:
    def __init__(self, status_id: int = None, status_name: str = None):
        self.__status_id = status_id
        self.__status_name = status_name

    # Getters
    def get_status_id(self):
        return self.__status_id

    def get_status_name(self):
        return self.__status_name

    # Setters
    def set_status_id(self, status_id):
        self.__status_id = status_id

    def set_status_name(self, status_name):
        self.__status_name = status_name
```

- ***Suspect***



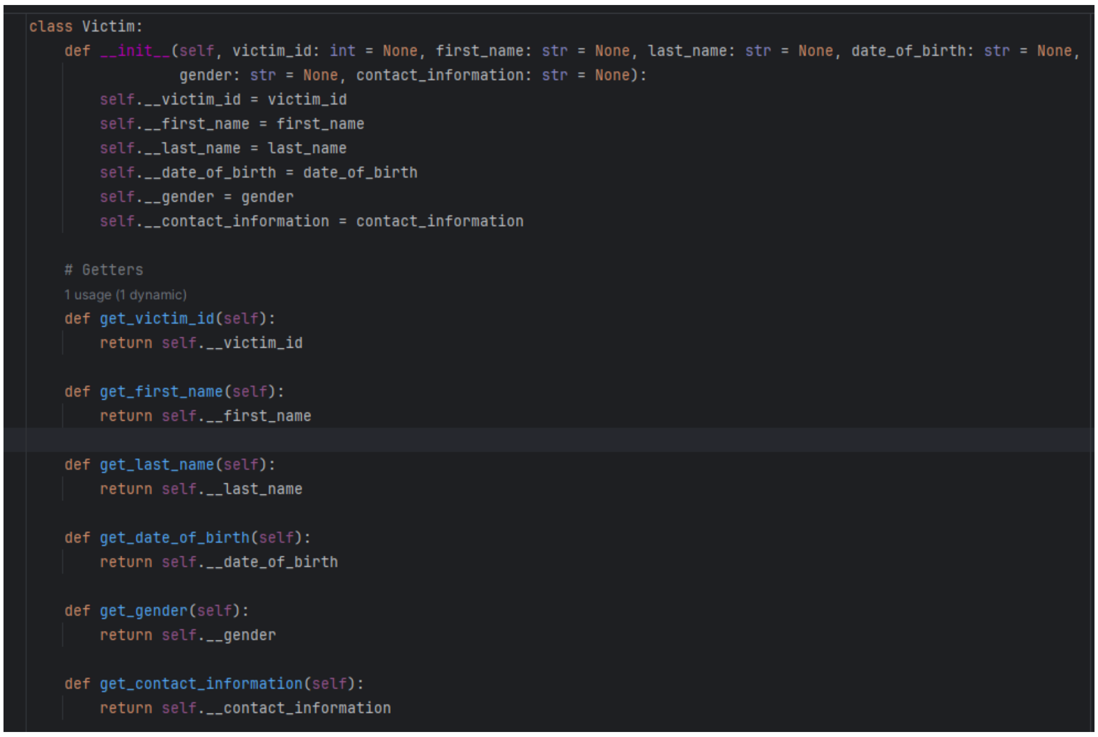
```
class Suspect:
    def __init__(self, suspect_id: int = None, first_name: str = None, last_name: str = None, date_of_birth: str = None,
                 gender: str = None, contact_information: str = None):
        self.__suspect_id = suspect_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__date_of_birth = date_of_birth
        self.__gender = gender
        self.__contact_information = contact_information

    # Getters
    1 usage (1 dynamic)
    def get_suspect_id(self):
        return self.__suspect_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_date_of_birth(self):
        return self.__date_of_birth

    def get_gender(self):
        return self.__gender

    def get_contact_information(self):
        return self.__contact_information
```

- ++

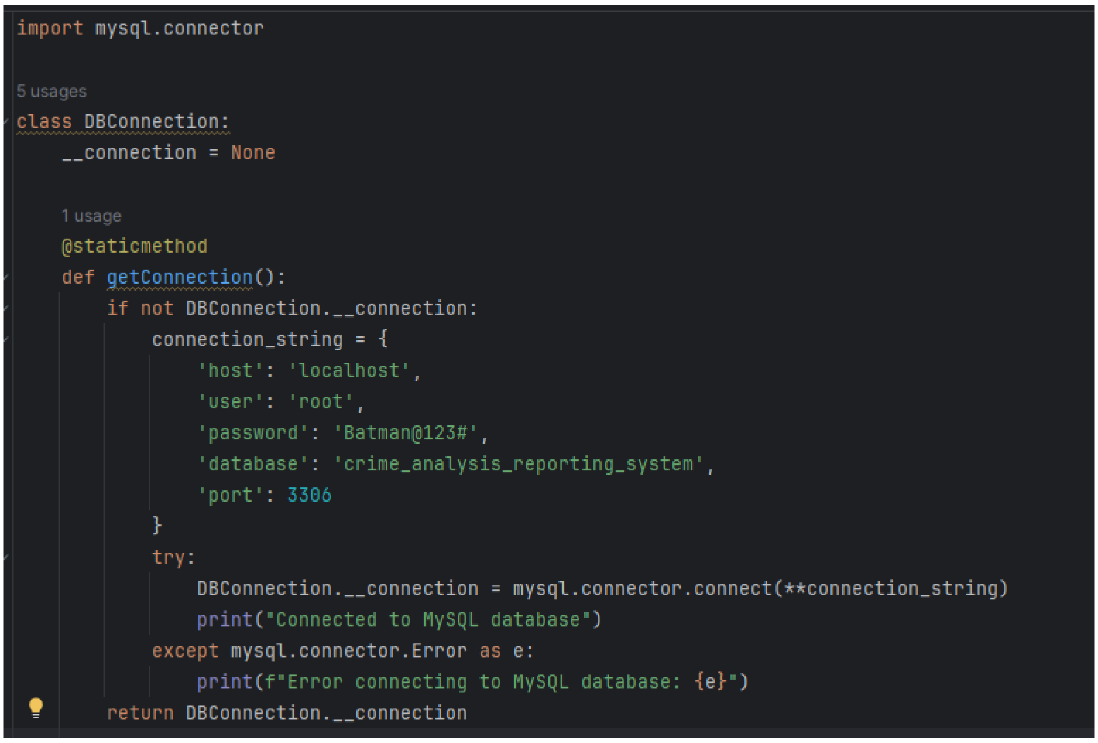


```
    # Setters
    def set_suspect_id(self, suspect_id):
        self.__suspect_id = suspect_id

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_date_of_birth(self, date_of_birth):
        self.__date_of_birth = date_of_birth

    def set_gender(self, gender):
        self.__gender = gender

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information
```

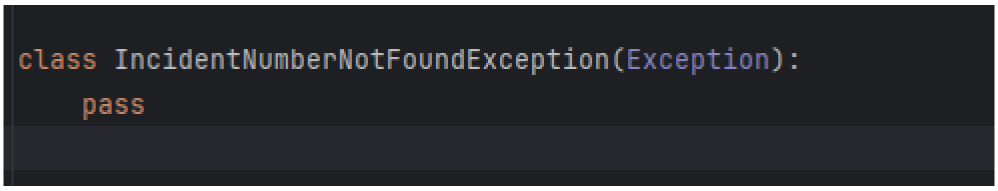
- ***Victim***


```python
class Victim:
    def __init__(self, victim_id: int = None, first_name: str = None, last_name: str = None, date_of_birth: str = None,
                 gender: str = None, contact_information: str = None):
        self.__victim_id = victim_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__date_of_birth = date_of_birth
        self.__gender = gender
        self.__contact_information = contact_information

    # Getters
    1 usage (1 dynamic)
    def get_victim_id(self):
        return self.__victim_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_date_of_birth(self):
        return self.__date_of_birth

    def get_gender(self):
        return self.__gender

    def get_contact_information(self):
        return self.__contact_information
```

- ***DBUtil (Connecting to the database)***
  - ***DB_Connection***

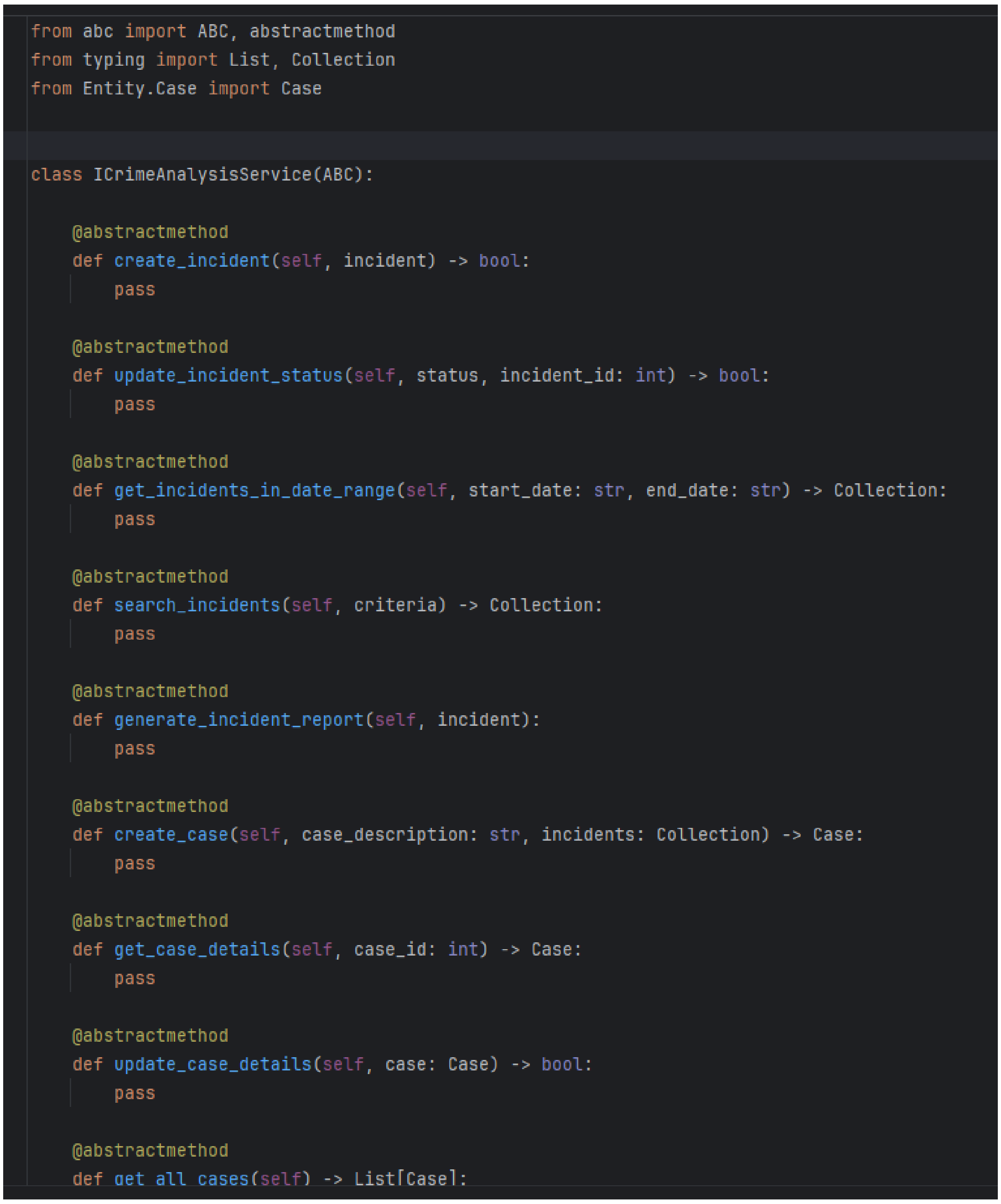

```python
import mysql.connector

5 usages
class DBConnection:
    __connection = None

    1 usage
    @staticmethod
    def getConnection():
        if not DBConnection.__connection:
            connection_string = {
                'host': 'localhost',
                'user': 'root',
                'password': 'Batman@123#',
                'database': 'crime_analysis_reporting_system',
                'port': 3306
            }
            try:
                DBConnection.__connection = mysql.connector.connect(**connection_string)
                print("Connected to MySQL database")
            except mysql.connector.Error as e:
                print(f"Error connecting to MySQL database: {e}")
        return DBConnection.__connection
```

  - ***PropertyUtil[not necessarily needed]***
- ***Exceptions***



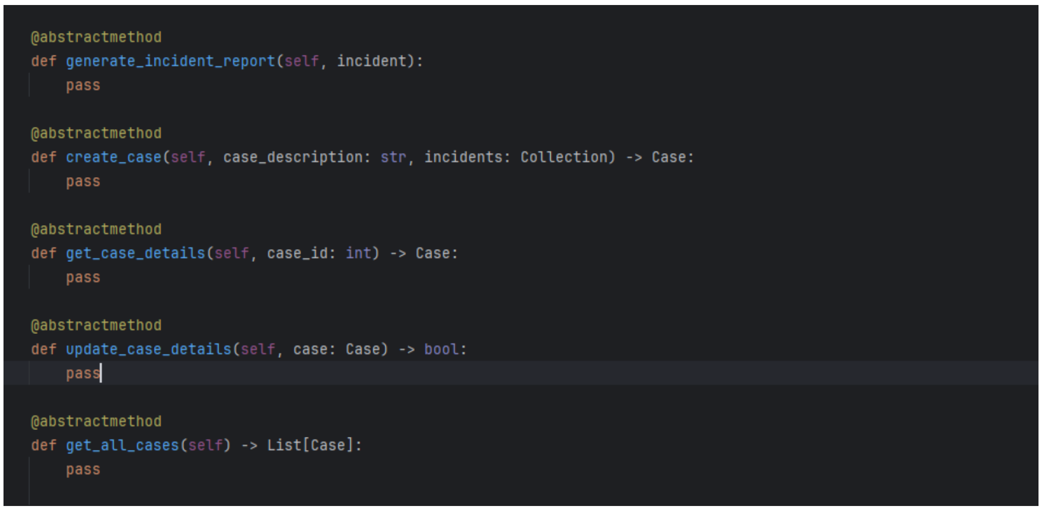
```
class IncidentNumberNotFoundException(Exception):
    pass
```

- ***Service class (data access objects)***
  - ***ICrimeAnalysisService***



```
from abc import ABC, abstractmethod
from typing import List, Collection
from Entity.Case import Case


class ICrimeAnalysisService(ABC):

    @abstractmethod
    def create_incident(self, incident) -> bool:
        pass

    @abstractmethod
    def update_incident_status(self, status, incident_id: int) -> bool:
        pass

    @abstractmethod
    def get_incidents_in_date_range(self, start_date: str, end_date: str) -> Collection:
        pass

    @abstractmethod
    def search_incidents(self, criteria) -> Collection:
        pass

    @abstractmethod
    def generate_incident_report(self, incident):
        pass

    @abstractmethod
    def create_case(self, case_description: str, incidents: Collection) -> Case:
        pass

    @abstractmethod
    def get_case_details(self, case_id: int) -> Case:
        pass

    @abstractmethod
    def update_case_details(self, case: Case) -> bool:
        pass

    @abstractmethod
    def get_all_cases(self) -> List[Case]:
```

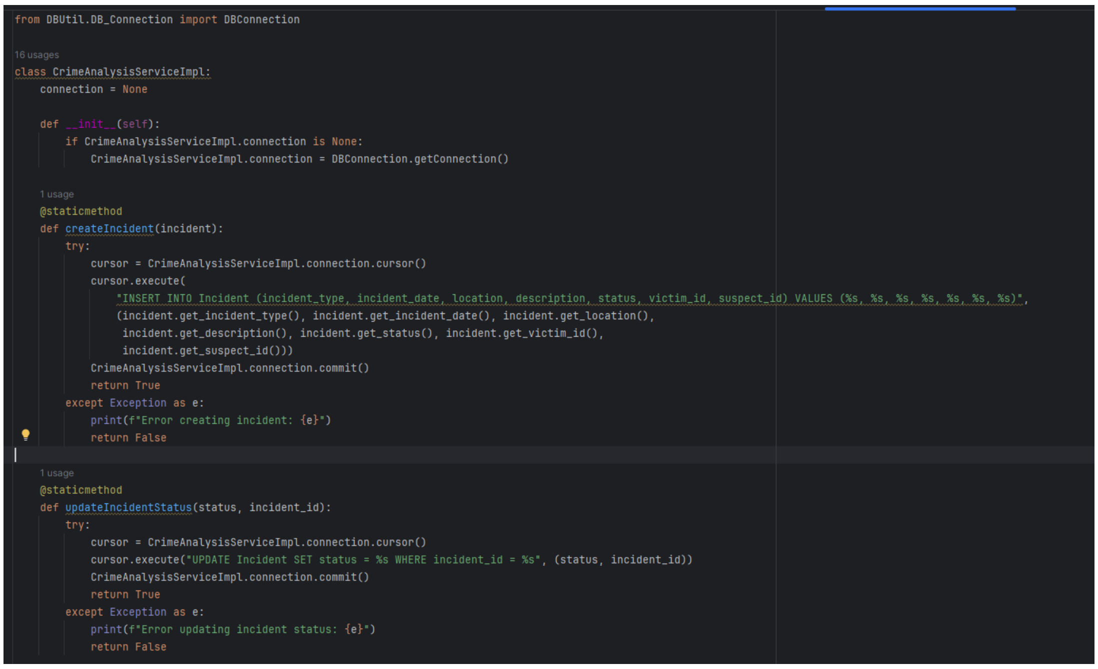
- ++



```python
@abstractmethod
def generate_incident_report(self, incident):
    pass

@abstractmethod
def create_case(self, case_description: str, incidents: Collection) -> Case:
    pass

@abstractmethod
def get_case_details(self, case_id: int) -> Case:
    pass

@abstractmethod
def update_case_details(self, case: Case) -> bool:
    pass

@abstractmethod
def get_all_cases(self) -> List[Case]:
    pass
```

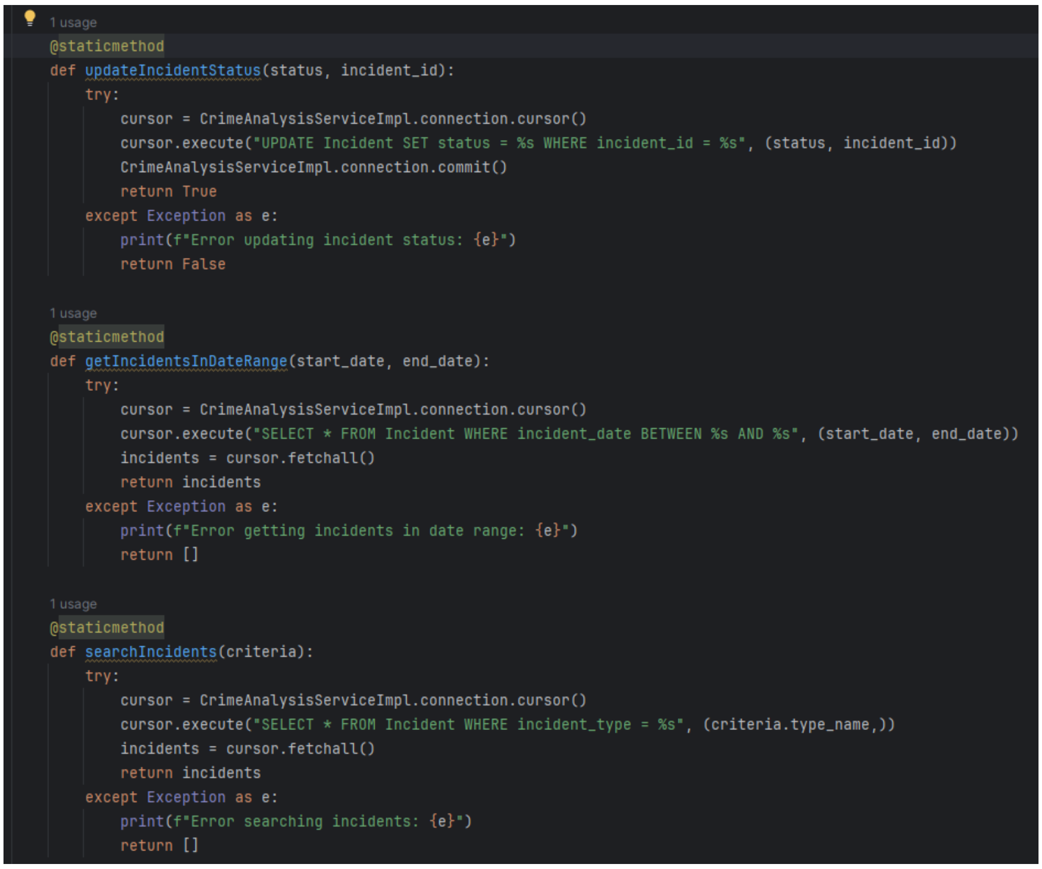
- ■

- ◦ ***CrimeAnalysisServiceImp***



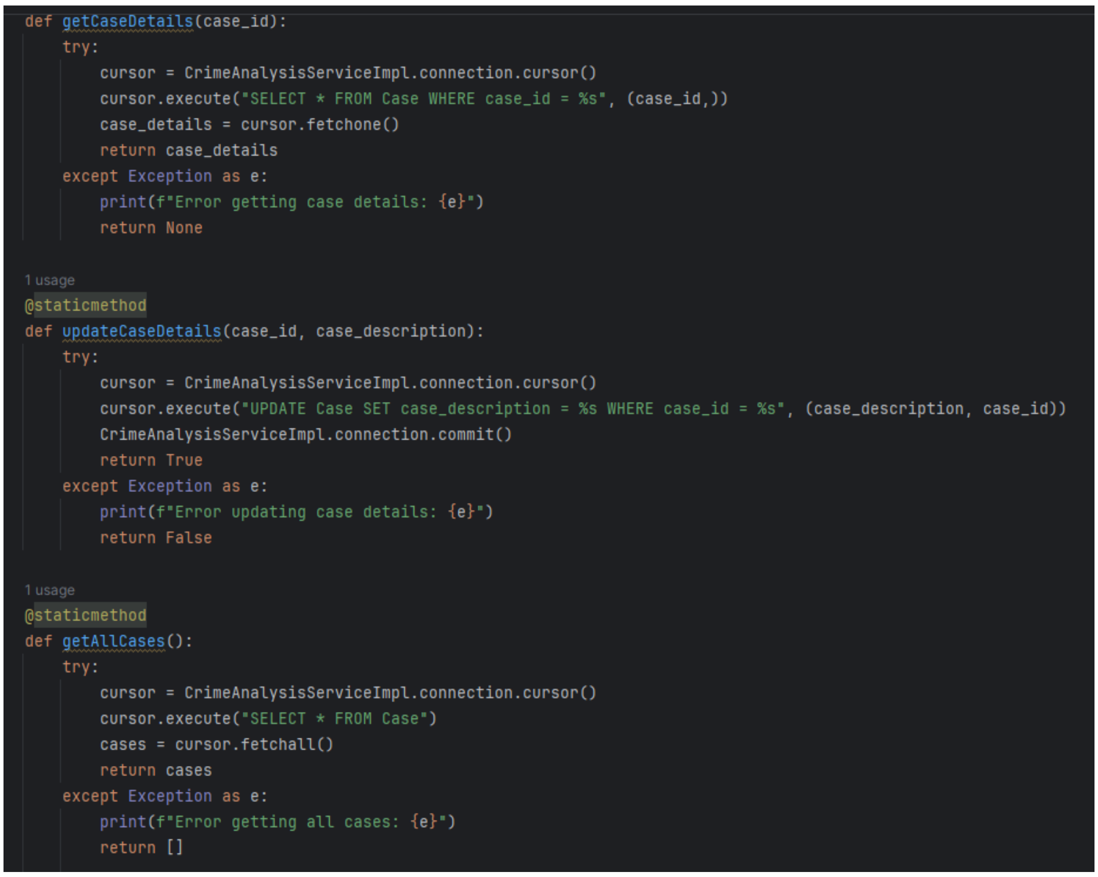
```python
from DBUtil.DB_Connection import DBConnection

16 usages
class CrimeAnalysisServiceImpl:
    connection = None

    def __init__(self):
        if CrimeAnalysisServiceImpl.connection is None:
            CrimeAnalysisServiceImpl.connection = DBConnection.getConnection()

    1 usage
    @staticmethod
    def createIncident(incident):
        try:
            cursor = CrimeAnalysisServiceImpl.connection.cursor()
            cursor.execute(
                "INSERT INTO Incident (incident_type, incident_date, location, description, status, victim_id, suspect_id) VALUES (%s, %s, %s, %s, %s, %s, %s)",
                (incident.get_incident_type(), incident.get_incident_date(), incident.get_location(),
                 incident.get_description(), incident.get_status(), incident.get_victim_id(),
                 incident.get_suspect_id()))
            CrimeAnalysisServiceImpl.connection.commit()
            return True
        except Exception as e:
            print(f"Error creating incident: {e}")
            return False

    1 usage
    @staticmethod
    def updateIncidentStatus(status, incident_id):
        try:
            cursor = CrimeAnalysisServiceImpl.connection.cursor()
            cursor.execute("UPDATE Incident SET status = %s WHERE incident_id = %s", (status, incident_id))
            CrimeAnalysisServiceImpl.connection.commit()
            return True
        except Exception as e:
            print(f"Error updating incident status: {e}")
            return False
```

- ■ ++



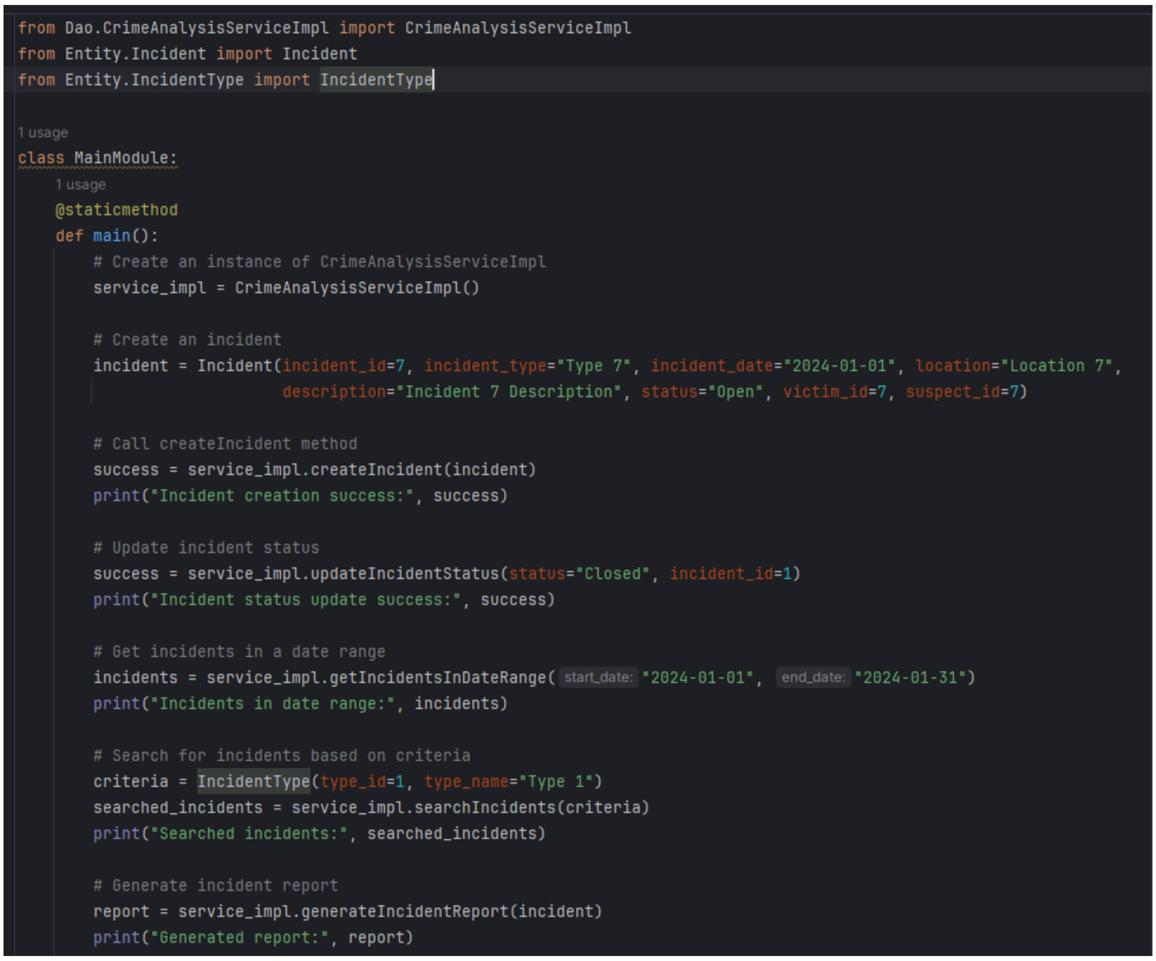
```
1 usage
@staticmethod
def updateIncidentStatus(status, incident_id):
    try:
        cursor = CrimeAnalysisServiceImpl.connection.cursor()
        cursor.execute("UPDATE Incident SET status = %s WHERE incident_id = %s", (status, incident_id))
        CrimeAnalysisServiceImpl.connection.commit()
        return True
    except Exception as e:
        print(f"Error updating incident status: {e}")
        return False

1 usage
@staticmethod
def getIncidentsInDateRange(start_date, end_date):
    try:
        cursor = CrimeAnalysisServiceImpl.connection.cursor()
        cursor.execute("SELECT * FROM Incident WHERE incident_date BETWEEN %s AND %s", (start_date, end_date))
        incidents = cursor.fetchall()
        return incidents
    except Exception as e:
        print(f"Error getting incidents in date range: {e}")
        return []

1 usage
@staticmethod
def searchIncidents(criteria):
    try:
        cursor = CrimeAnalysisServiceImpl.connection.cursor()
        cursor.execute("SELECT * FROM Incident WHERE incident_type = %s", (criteria.type_name,))
        incidents = cursor.fetchall()
        return incidents
    except Exception as e:
        print(f"Error searching incidents: {e}")
        return []
```

- ++

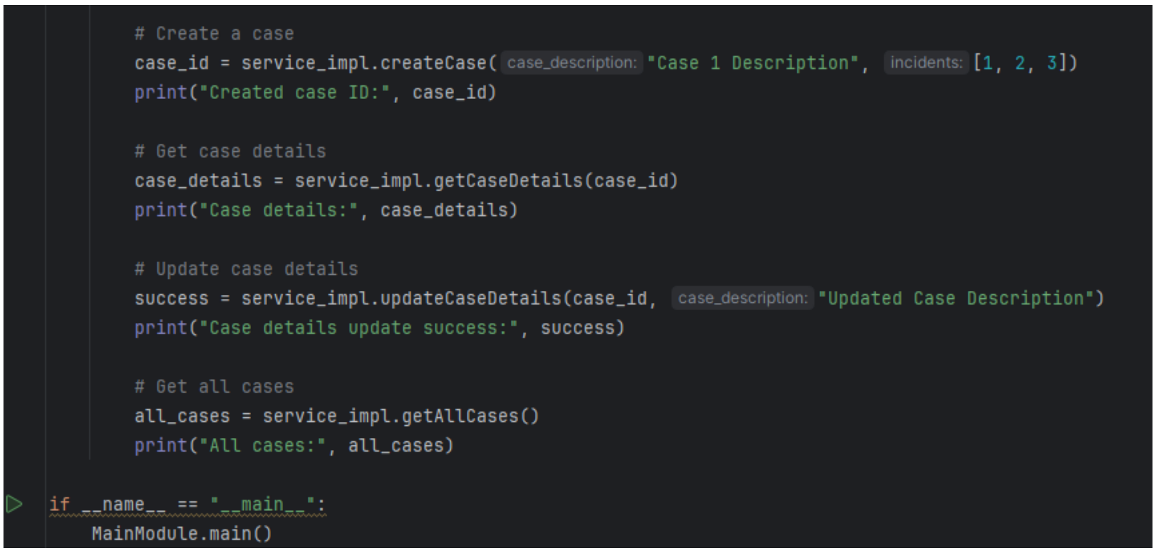


```
def getCaseDetails(case_id):
    try:
        cursor = CrimeAnalysisServiceImpl.connection.cursor()
        cursor.execute("SELECT * FROM Case WHERE case_id = %s", (case_id,))
        case_details = cursor.fetchone()
        return case_details
    except Exception as e:
        print(f"Error getting case details: {e}")
        return None

1 usage
@staticmethod
def updateCaseDetails(case_id, case_description):
    try:
        cursor = CrimeAnalysisServiceImpl.connection.cursor()
        cursor.execute("UPDATE Case SET case_description = %s WHERE case_id = %s", (case_description, case_id))
        CrimeAnalysisServiceImpl.connection.commit()
        return True
    except Exception as e:
        print(f"Error updating case details: {e}")
        return False

1 usage
@staticmethod
def getAllCases():
    try:
        cursor = CrimeAnalysisServiceImpl.connection.cursor()
        cursor.execute("SELECT * FROM Case")
        cases = cursor.fetchall()
        return cases
    except Exception as e:
        print(f"Error getting all cases: {e}")
        return []
```

- ***Main Module***
    -



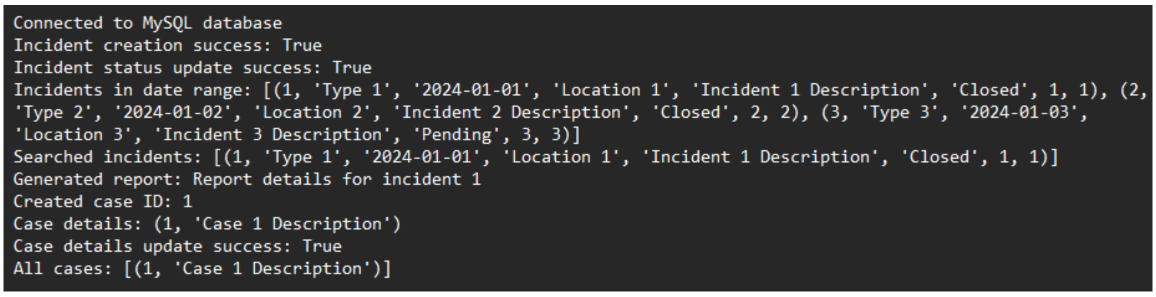
```
from Dao.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
from Entity.Incident import Incident
from Entity.IncidentType import IncidentType

1 usage
class MainModule:
    1 usage
    @staticmethod
    def main():
        # Create an instance of CrimeAnalysisServiceImpl
        service_impl = CrimeAnalysisServiceImpl()

        # Create an incident
        incident = Incident(incident_id=7, incident_type="Type 7", incident_date="2024-01-01", location="Location 7",
                            description="Incident 7 Description", status="Open", victim_id=7, suspect_id=7)

        # Call createIncident method
        success = service_impl.createIncident(incident)
        print("Incident creation success:", success)

        # Update incident status
        success = service_impl.updateIncidentStatus(status="Closed", incident_id=1)
        print("Incident status update success:", success)

        # Get incidents in a date range
        incidents = service_impl.getIncidentsInDateRange( start_date: "2024-01-01",  end_date: "2024-01-31")
        print("Incidents in date range:", incidents)

        # Search for incidents based on criteria
        criteria = IncidentType(type_id=1, type_name="Type 1")
        searched_incidents = service_impl.searchIncidents(criteria)
        print("Searched incidents:", searched_incidents)

        # Generate incident report
        report = service_impl.generateIncidentReport(incident)
        print("Generated report:", report)
```



```
        # Create a case
        case_id = service_impl.createCase( case_description: "Case 1 Description",  incidents: [1, 2, 3])
        print("Created case ID:", case_id)

        # Get case details
        case_details = service_impl.getCaseDetails(case_id)
        print("Case details:", case_details)

        # Update case details
        success = service_impl.updateCaseDetails(case_id,  case_description: "Updated Case Description")
        print("Case details update success:", success)

        # Get all cases
        all_cases = service_impl.getAllCases()
        print("All cases:", all_cases)

if __name__ == "__main__":
    MainModule.main()
```

    - ++

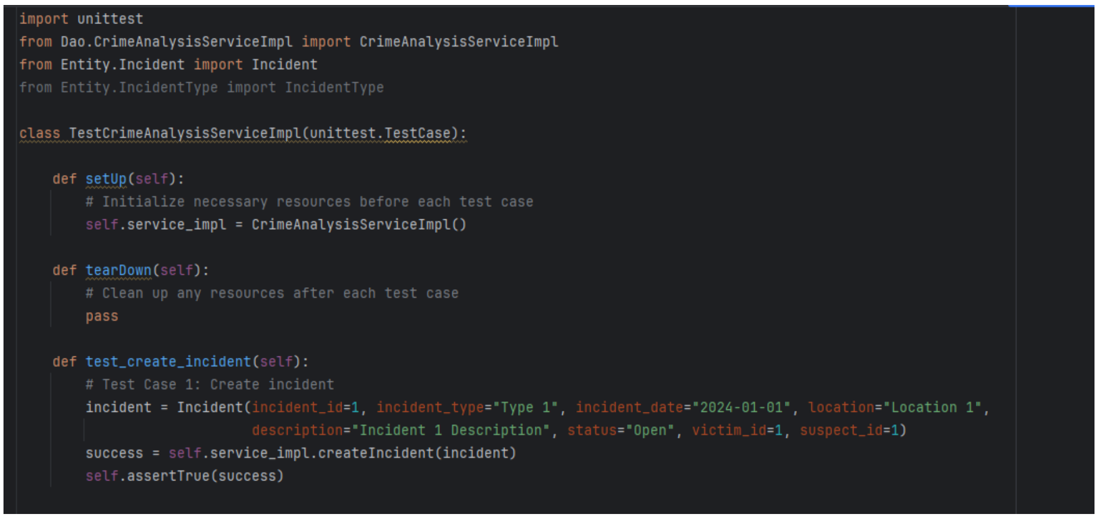
- **_Output_**



```
Connected to MySQL database
Incident creation success: True
Incident status update success: True
Incidents in date range: [(1, 'Type 1', '2024-01-01', 'Location 1', 'Incident 1 Description', 'Closed', 1, 1), (2,
'Type 2', '2024-01-02', 'Location 2', 'Incident 2 Description', 'Closed', 2, 2), (3, 'Type 3', '2024-01-03',
'Location 3', 'Incident 3 Description', 'Pending', 3, 3)]
Searched incidents: [(1, 'Type 1', '2024-01-01', 'Location 1', 'Incident 1 Description', 'Closed', 1, 1)]
Generated report: Report details for incident 1
Created case ID: 1
Case details: (1, 'Case 1 Description')
Case details update success: True
All cases: [(1, 'Case 1 Description')]
```

  - Output:

- **_Creating Additional Test Cases:-_**



```python
import unittest
from Dao.CrimeAnalysisServiceImpl import CrimeAnalysisServiceImpl
from Entity.Incident import Incident
from Entity.IncidentType import IncidentType

class TestCrimeAnalysisServiceImpl(unittest.TestCase):

    def setUp(self):
        # Initialize necessary resources before each test case
        self.service_impl = CrimeAnalysisServiceImpl()

    def tearDown(self):
        # Clean up any resources after each test case
        pass

    def test_create_incident(self):
        # Test Case 1: Create incident
        incident = Incident(incident_id=1, incident_type="Type 1", incident_date="2024-01-01", location="Location 1",
                            description="Incident 1 Description", status="Open", victim_id=1, suspect_id=1)
        success = self.service_impl.createIncident(incident)
        self.assertTrue(success)
```

    - Note : we can add additional test cases for everything else too