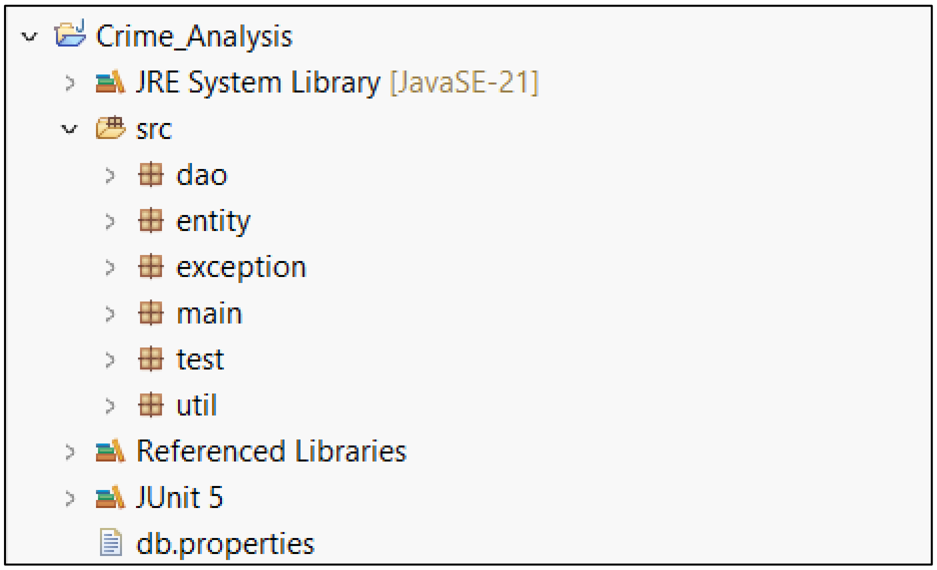
Case Study: Crime Analysis and Reporting System (C.A.R.S.)

1. **The following Directory structure is to be followed in the application.**
    - **entity**
        - Create entity classes in this package. All entity class should not have any business logic.
    - **dao**
        - Create Service Provider interface to showcase functionalities.
        - Create the implementation class for the above interface with db interaction.
    - **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - **util**
        - Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
        - Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).
    - **main**
        - Create a class MainModule and demonstrate the functionalities in a menu driven application.



```
Crime_Analysis
  JRE System Library [JavaSE-21]
  src
    dao
    entity
    exception
    main
    test
    util
  Referenced Libraries
  JUnit 5
  db.properties
```

2. Coding

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)

➢ EVIDENCE

```
package entity;

public class Evidence {
   private int evidenceID;
   private String description;
   private int incidentID; // Foreign Key from the incident table
   private String locationFound;

   // Default Constructor
   public Evidence() {}

   // Parameterized Constructor
   public Evidence(int evidenceID , String description, String locationFound,int incidentID) {
      this.evidenceID = evidenceID;
      this.incidentID = incidentID;
      this.locationFound = locationFound;
      this.description = description;
   }

   // Getters and Setters
   public int getEvidenceID() {
      return evidenceID;
   }

   public void setEvidenceID(int evidenceID) {
      this.evidenceID = evidenceID;
   }

   public String getLocationFound() {
      return locationFound;
   }

   public void setLocationFound(String locationFound) {
      this.locationFound = locationFound;
   }

   public String getDescription() {
      return description;
   }

   public void setDescription(String description) {
      this.description = description;
   }

   public int getIncidentID() {
      return incidentID;
   }

   public void setIncidentID(int incidentID) {
        this.incidentID = incidentID;
    }

            @Override
            public String toString() {
                        return "Evidence [evidenceID=" + evidenceID + ", description=" + description + ", incidentID=" +
incidentID
                                                + ", locationFound=" + locationFound + "]";
            }
}
```

- **INCIDENT**

```
package entity;
import java.time.LocalDate;

public class Incident {
//instance variable
private int incidentID;
private String incidentType;
private LocalDate incidentDate;
private String location;
private String description;
private String status;
private int victimID;
private int suspectID;
private int officerID;

// Default Constructor
public Incident() {}

// Parameterized Constructor
public Incident(int incidentID, String incidentType, LocalDate incidentDate, String location, String description, String
status, int victimID, int suspectID, int officerID) {
this.incidentID = incidentID;
this.incidentType = incidentType;
this.incidentDate = incidentDate;
this.location = location;
this.description = description;
this.status = status;
this.victimID = victimID;
this.suspectID = suspectID;
this.officerID = officerID;
}

// Getter and Setter
public int getIncidentID() {
return incidentID;
}

public void setIncidentID(int incidentID) {
this.incidentID = incidentID;
}

public String getIncidentType() {
```

```
return incidentType;
}

public void setIncidentType(String incidentType) {
this.incidentType = incidentType;
}

public LocalDate getIncidentDate() {
return incidentDate;
}

public void setIncidentDate(LocalDate incidentDate) {
this.incidentDate = incidentDate;
}

public String getLocation() {
return location;
}

public void setLocation(String location) {
this.location = location;
}

public String getDescription() {
return description;
}

public void setDescription(String description) {
this.description = description;
}

public String getStatus() {
return status;
}

public void setStatus(String status) {
this.status = status;
}

public int getVictimID() {
return victimID;
}

public void setVictimID(int victimID) {
this.victimID = victimID;
}

public int getSuspectID() {
return suspectID;
}

public void setSuspectID(int suspectID) {
this.suspectID = suspectID;
}

public int getOfficerID() {
return officerID;
```

```
}

public void setOfficerID(int officerID) {
this.officerID = officerID;
}

@Override
public String toString() {
return "Incident [incidentID=" + incidentID + ", incidentType=" + incidentType + ", incidentDate="
+ incidentDate + ", location=" + location + ", description=" + description + ", status=" + status
+ ", victimID=" + victimID + ", suspectID=" + suspectID + ", officerID=" + officerID + "]";
}
}
```

➢ LAWENFORCEMENTAGENCY

```
package entity;

public class LawEnforcementAgency {
    private int agencyID;
    private String agencyName;
    private String jurisdiction;
    private String contactInfo;

    // Constructors
    public LawEnforcementAgency() {}

    public LawEnforcementAgency(int agencyID, String agencyName, String jurisdiction, String contactInfo) {
        this.agencyID = agencyID;
        this.agencyName = agencyName;
        this.jurisdiction = jurisdiction;
        this.contactInfo = contactInfo;
    }

    // Getters and Setters
    public int getAgencyID() {
        return agencyID;
     }
    public void setAgencyID(int agencyID) {
        this.agencyID = agencyID;
    }

    public String getAgencyName() {
        return agencyName;
    }
    public void setAgencyName(String agencyName) {
        this.agencyName = agencyName;
    }

    public String getJurisdiction() {
        return jurisdiction;
    }
    public void setJurisdiction(String jurisdiction) {
        this.jurisdiction = jurisdiction;
    }
    public String getContactInfo() {
        return contactInfo;
    }
    public void setContactInfo(String contactInfo) {
        this.contactInfo = contactInfo;
    }

            @Override
            public String toString() {
                        return "LawEnforcementAgency [agencyID=" + agencyID + ", agencyName=" + agencyName + ",
jurisdiction="
                                                + jurisdiction + ", contactInfo=" + contactInfo + "]";
            }
}
```

➢ OFFICER

```
package entity;

public class Officer {
    private int officerID;
    private String firstName;
    private String lastName;
    private String badgeNumber;
    private String rank;
    private String contactInfo;
    private int agencyID; // Foreign Key

    // Constructors
    public Officer() {}

    public Officer(int officerID, String firstName, String lastName, String badgeNumber, String rank, String contactInfo,
int agencyID) {
        this.officerID = officerID;
        this.firstName = firstName;
        this.lastName = lastName;
        this.badgeNumber = badgeNumber;
        this.rank = rank;
        this.contactInfo = contactInfo;
        this.agencyID = agencyID;
    }

    // Getters and Setters
    public int getOfficerID() {
        return officerID;
    }
    public void setOfficerID(int officerID) {
        this.officerID = officerID;
    }

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getBadgeNumber() {
        return badgeNumber;
    }
    public void setBadgeNumber(String badgeNumber) {
        this.badgeNumber = badgeNumber;
    }

    public String getRank() {
        return rank;
    }
    public void setRank(String rank) {
        this.rank = rank;
    }

    public String getContactInfo() {
        return contactInfo;
    }
    public void setContactInfo(String contactInfo) {
        this.contactInfo = contactInfo;
    }

    public int getAgencyID() {
        return agencyID;
    }
    public void setAgencyID(int agencyID) {
        this.agencyID = agencyID;
    }

        @Override
        public String toString() {
                return "Officer [officerID=" + officerID + ", firstName=" + firstName + ", lastName=" + lastName
                                + ", badgeNumber=" + badgeNumber + ", rank=" + rank + ", contactInfo=" +
contactInfo + ", agencyID="
                                + agencyID + "]";
        }
}
```

➢ REPORT

```
package entity;
import java.time.LocalDate;

public class Report {
    private int reportID;
    private int incidentID; // Foreign Key
    private int reportingOfficer; // Foreign Key
    private LocalDate reportDate;
    private String reportDetails;
    private String status;

    // Constructors
    public Report() {}

    public Report(int reportID, int incidentID, int reportingOfficer, LocalDate reportDate, String reportDetails, String
status) {
        this.reportID = reportID;
        this.incidentID = incidentID;
        this.reportingOfficer = reportingOfficer;
        this.reportDate = reportDate;
        this.reportDetails = reportDetails;
        this.status = status;
    }

    // Getters and Setters
    public int getReportID() {
        return reportID;
    }
    public void setReportID(int reportID) {
        this.reportID = reportID;
    }

    public int getIncidentID() {
        return incidentID;
    }
    public void setIncidentID(int incidentID) {
        this.incidentID = incidentID;
    }

    public int getReportingOfficer() {
        return reportingOfficer;
    }
    public void setReportingOfficer(int reportingOfficer) {
        this.reportingOfficer = reportingOfficer;
    }

    public LocalDate getReportDate() {
        return reportDate;
    }
    public void setReportDate(LocalDate reportDate) {
        this.reportDate = reportDate;
    }

    public String getReportDetails() {
        return reportDetails;
    }
    public void setReportDetails(String reportDetails) {
        this.reportDetails = reportDetails;
    }

    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }

	@Override
	public String toString() {
		return "Report [reportID=" + reportID + ", incidentID=" + incidentID + ", reportingOfficer=" + 
reportingOfficer
				+ ", reportDate=" + reportDate + ", reportDetails=" + reportDetails + ", status=" + 
status + "]";
	}
}
```

➢ VICTIM

```
package entity;
import java.time.LocalDate;

public class Victim {
	//instance variable
	private int victimID;
	private String firstName;
	private String lastName;
	private LocalDate dateOfBirth;
	private String gender;
	private String contactInfo;

	//default constructor
	public Victim()  {}

	//parameterized constructor
	public Victim(int victimID,String firstName,String lastName,LocalDate dateOfBirth,String gender,String 
contactInfo) {
		this.victimID=victimID;
		this.firstName=firstName;
		this.dateOfBirth=dateOfBirth;
		this.gender=gender;
		this.contactInfo=contactInfo;
	}

	//getter and setter
	public int getVictimID() {
		return victimID;
	}
	public void setVictimID(int victimID) {
		this.victimID=victimID;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName=firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName=lastName;
```

```
            }
            public LocalDate getDateOfBirth() {
                        return dateOfBirth;
            }
            public void setDateOfBirth(LocalDate dateOfBirth) {
                        this.dateOfBirth=dateOfBirth;
            }
            public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getContactInfo() {
        return contactInfo;
    }
    public void setContactInfo(String contactInfo) {
        this.contactInfo = contactInfo;
    }

            @Override
            public String toString() {
                        return "Victim [victimID=" + victimID + ", firstName=" + firstName + ", lastName=" + lastName
+ ", dateOfBirth="
                                    + dateOfBirth + ", gender=" + gender + ", contactInfo=" + contactInfo + "]";
            }

}
```

➢ **SUSPECT**

```
package entity;
import java.time.LocalDate;

public class Suspect {
    private int suspectID;
    private String firstName;
    private String lastName;
    private LocalDate dateOfBirth;
    private String gender;
    private String contactInfo;

    // Constructors
    public Suspect() {}

    public Suspect(int suspectID, String firstName, String lastName, LocalDate dateOfBirth, String gender, String
contactInfo) {
        this.suspectID = suspectID;
        this.firstName = firstName;
        this.lastName = lastName;
        this.dateOfBirth = dateOfBirth;
        this.gender = gender;
        this.contactInfo = contactInfo;
    }
    // Getters and Setters
    public int getSuspectID() {
        return suspectID;
    }
    public void setSuspectID(int suspectID) {
        this.suspectID = suspectID;
    }

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public LocalDate getDateOfBirth() {
        return dateOfBirth;
    }
    public void setDateOfBirth(LocalDate dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getContactInfo() {
        return contactInfo;
    }
    public void setContactInfo(String contactInfo) {
        this.contactInfo = contactInfo;
    }

          @Override
          public String toString() {
                  return "Suspect [suspectID=" + suspectID + ", firstName=" + firstName + ", lastName=" + lastName
                                  + ", dateOfBirth=" + dateOfBirth + ", gender=" + gender + ", contactInfo=" +
contactInfo + "]";
          }
}
```

3. Service Provider Interface/Abstract class

Keep the interfaces and implementation classes in package dao

Create ICrimeAnalysisService Interface/abstract classs with the following methods

// **Create a new incident**

createIncident();

parameters- **Incident** object

return type Boolean

// **Update the status of an incident**

updateIncidentStatus();

parameters- **Status** object,incidentid

return type Boolean

// **Get a list of incidents within a date range**

getIncidentsInDateRange();

parameters- startDate, endDate

return type Collection of Incident objects

// **Search for incidents based on various criteria**

searchIncidents(IncidentType criteria);

parameters- **IncidentType object**

return type Collection of Incident objects

// **Generate incident reports**

generateIncidentReport();

parameters- **Incident object**

return type Report object

- **CRIMEANALYSISSERVICE**

```
package dao;
import entity.Incident;
import java.util.Collection;
import entity.Report;
import exception.IncidentNumberNotFoundException;

import java.time.LocalDate;


public interface CrimeAnalysisService {
    // Create a new incident
    public boolean createIncident(Incident incident);

    // Update the status of an incident
    public boolean updateIncidentStatus(int incidentId, String status)  throws IncidentNumberNotFoundException;

    // Get a list of incidents within a date range
    public Collection<Incident> getIncidentsInDateRange(LocalDate startDate, LocalDate endDate);

    // Search for incidents based on various criteria (e.g., incident type)
    public Collection<Incident> searchIncidents(String incidentType);

    // Generate incident report
    public Report generateIncidentReport(Incident incident) throws IncidentNumberNotFoundException;

}
```

4. Connect your application to the SQL database:

Write code to establish a connection to your SQL database.

Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

Connection properties supplied in the connection string should be read from a property file.

Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

- **DBConnection**

```
package util;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnUtil {

            private static final String fileName = "db.properties";

            public static Connection getDbConnection() {
               Connection con = null;
               String connString=null;
               try {
                   connString = DBPropertyUtil.getConnectionString(fileName); // Get URL from properties
               } catch (IOException e) {
                   System.out.println("Connection string could not be retrieved.");
                   e.printStackTrace();
               }
```

```
                    if (connString != null) {
                     try {
                          con = DriverManager.getConnection(connString); // Get actual Connection object
                     }
                     catch (SQLException e) {
                       System.out.println("Database connection failed.");
                       e.printStackTrace();
                    }
                    }
                    return con;
                 }
              }
```

- **PropertyUtil**

```
package util;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class DBPropertyUtil {

                     //this method takes the filename which contains db connection like
                     //user name, pwd, port number, protocol and db name as an argument
                     //and returns a connection
                     public static String getConnectionString(String fileName)throws IOException {
                               String connStr=null;
                               Properties props=new Properties();
                               FileInputStream fis=new FileInputStream(fileName);
                               props.load(fis);

                               String user = props.getProperty("user");
                               String password = props.getProperty("password");
                    String port = props.getProperty("port");
                    String database = props.getProperty("database");
                    String protocol = props.getProperty("protocol");
                    String system = props.getProperty("system");


connStr=protocol+"//"+system+":"+port+"/"+database+"?user="+user+"&password="+password;
                               return connStr;
                     }

              }
```

5. Service implementation

Create a Service class CrimeAnalysisServiceImpl in package dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the getConnection() method in DBConnection class

Provide implementation for all the methods in the interface/abstract clsass

➢ CRIMEANALYSISSERVICEIMPL

```
package dao;
import entity.Incident;
import entity.Report;
import java.util.Collection;
import java.util.List;
import java.util.ArrayList;
import java.sql.*;
import java.time.LocalDate;
import exception.IncidentNumberNotFoundException;
import util.DBConnUtil;


public class CrimeAnalysisServiceImpl implements CrimeAnalysisService {

private static Connection connection;

public CrimeAnalysisServiceImpl()  throws SQLException {
connection =DBConnUtil.getDbConnection();
}

@Override
public boolean createIncident(Incident incident) {
// DB logic to insert incident
try {
PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Incidents (IncidentID, IncidentType,
IncidentDate, Location, Description, Status, VictimID, SuspectID, OfficerID) " +
"VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");

pstmt.setInt(1, incident.getIncidentID());
pstmt.setString(2, incident.getIncidentType());
pstmt.setDate(3, java.sql.Date.valueOf(incident.getIncidentDate()));
pstmt.setString(4, incident.getLocation());
pstmt.setString(5, incident.getDescription());
pstmt.setString(6, incident.getStatus());
pstmt.setInt(7, incident.getVictimID());
pstmt.setInt(8, incident.getSuspectID());
pstmt.setInt(9, incident.getOfficerID());

int rows = pstmt.executeUpdate();
return rows > 0;

}
catch (SQLException e)
{
e.printStackTrace();
}
return false; // placeholder
}

@Override
public boolean updateIncidentStatus(int incidentId, String status) throws IncidentNumberNotFoundException {
// DB logic to update status
try {
PreparedStatement pstmt = connection.prepareStatement("UPDATE Incidents SET Status = ? WHERE
IncidentID = ?");
```

```
pstmt.setString(1, status);
pstmt.setInt(2, incidentId);

int rows = pstmt.executeUpdate();

if (rows == 0) {
throw new IncidentNumberNotFoundException("Incident with ID " + incidentId + " not found.");
}

return true;

} catch (SQLException e) {
e.printStackTrace();
}
return false;
}

@Override
public Collection<Incident> getIncidentsInDateRange(LocalDate startDate, LocalDate endDate) {
// DB logic to fetch by date range
List<Incident> incidents = new ArrayList<>();
try {
PreparedStatement pstmt = connection.prepareStatement("SELECT * FROM Incidents WHERE IncidentDate
BETWEEN ? AND ?");
pstmt.setDate(1, java.sql.Date.valueOf(startDate));
pstmt.setDate(2, java.sql.Date.valueOf(endDate));

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {
Incident incident = new Incident(
rs.getInt("IncidentID"),
rs.getString("IncidentType"),
rs.getDate("IncidentDate").toLocalDate(),
rs.getString("Location"),
rs.getString("Description"),
rs.getString("Status"),
rs.getInt("VictimID"),
rs.getInt("SuspectID"),
rs.getInt("OfficerID")
);
incidents.add(incident);
}

} catch (SQLException e) {
e.printStackTrace();
}
return incidents;
}

@Override
public Collection<Incident> searchIncidents(String incidentType) {
// DB logic to fetch by incident type
List<Incident> incidents = new ArrayList<>();
try {
PreparedStatement pstmt = connection.prepareStatement("SELECT * FROM Incidents WHERE IncidentType =
?");
```

```
pstmt.setString(1, incidentType);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {
Incident incident = new Incident(
rs.getInt("IncidentID"),
rs.getString("IncidentType"),
rs.getDate("IncidentDate").toLocalDate(),
rs.getString("Location"),
rs.getString("Description"),
rs.getString("Status"),
rs.getInt("VictimID"),
rs.getInt("SuspectID"),
rs.getInt("OfficerID")
);
incidents.add(incident);
}

} catch (SQLException e) {
e.printStackTrace();
}

return incidents;

}

@Override
public Report generateIncidentReport(Incident incident) {
// DB logic to generate and return report

String reportDetails = "Incident Type: " + incident.getIncidentType() + ", Description: " +
incident.getDescription();

try {
PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Reports (IncidentID, ReportingOfficer,
ReportDate, ReportDetails, Status) VALUES (?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);

pstmt.setInt(1, incident.getIncidentID());
pstmt.setInt(2, incident.getOfficerID()); // assuming officer who reported
pstmt.setDate(3, java.sql.Date.valueOf(LocalDate.now()));
pstmt.setString(4, reportDetails);
pstmt.setString(5, "Draft");

int rows = pstmt.executeUpdate();

if (rows > 0) {
ResultSet rs = pstmt.getGeneratedKeys();
if (rs.next()) {
int reportId = rs.getInt(1);
return new Report(reportId, incident.getIncidentID(), incident.getOfficerID(), LocalDate.now(), reportDetails,
"Draft");
}
}

} catch (SQLException e) {
e.printStackTrace();
```

```
}
return null;
}
}
```

6. Exception Handling

Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

IncidentNumberNotFoundException: throw this exception when user enters an invalid patient number which doesn’t exist in db

```
package exception;

public class IncidentNumberNotFoundException extends Exception {
            public IncidentNumberNotFoundException(String message) {
    super(message);
            }

}
```

7. MainMethod

Create class named MainModule with main method in main package. Trigger all the methods in service implementation class

```
package main;

import java.sql.SQLException;
import java.time.LocalDate;
import java.util.List;
import java.util.Scanner;

import dao.CrimeAnalysisServiceImpl;
import dao.CrimeAnalysisService;
import entity.Incident;
import entity.Report;
import exception.IncidentNumberNotFoundException;

public class mainModule {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      CrimeAnalysisService service;
```

```
try {
   try {
      service = new CrimeAnalysisServiceImpl();
   } catch (SQLException e) {
      System.out.println("oh...know!!!Failed to connect to database: " + e.getMessage());
      return;
   }

   String continueChoice;
   do {
      System.out.println("\n---  Helooo!! welcome to Crime Analysis and Reporting System ---");
      System.out.println("1. Create Incident");
      System.out.println("2. Update Incident Status");
      System.out.println("3. Get Incidents in Date Range");
      System.out.println("4. Search Incidents by Type");
      System.out.println("5. Generate Report");
      System.out.println("6. Exit");
      System.out.print("Enter choice: ");
      int choice = sc.nextInt();
      sc.nextLine(); // Clear buffer

      try {
         switch (choice) {
            case 1:
               System.out.print("Enter Incident Type: ");
               String type = sc.nextLine();
               System.out.print("Enter Date (yyyy-mm-dd): ");
               String dateStr = sc.nextLine();
               System.out.print("Enter Location: ");
               String location = sc.nextLine();
               System.out.print("Enter Description: ");
               String desc = sc.nextLine();
               System.out.print("Enter Status: ");
               String status = sc.nextLine();
               System.out.print("Enter Victim ID: ");
               int vid = sc.nextInt();
               System.out.print("Enter Suspect ID: ");
               int sid = sc.nextInt();
               System.out.print("Enter Officer ID: ");
               int oid = sc.nextInt();
               sc.nextLine();

               Incident inc = new Incident(0, type, LocalDate.parse(dateStr), location, desc, status, vid, sid, oid);
               boolean created = service.createIncident(inc);
               System.out.println(created ? "Incident created successfully." : "Failed to create incident.");
               break;

            case 2:
               System.out.print("Enter Incident ID to update: ");
               int incidentId = sc.nextInt();
               sc.nextLine();
               System.out.print("Enter new status: ");
               String newStatus = sc.nextLine();
               boolean updated = service.updateIncidentStatus(incidentId, newStatus);
               System.out.println(updated ? "Status updated." : "Incident not found.");
               break;
```

```
case 3:
   System.out.print("Enter Start Date (yyyy-mm-dd): ");
   LocalDate start = LocalDate.parse(sc.nextLine());
   System.out.print("Enter End Date (yyyy-mm-dd): ");
   LocalDate end = LocalDate.parse(sc.nextLine());

   if (start.isAfter(end)) {
      System.out.println("Start date must be before end date.");
      break;
   }

   List<Incident> rangeList = (List<Incident>) service.getIncidentsInDateRange(start, end);
   if (rangeList.isEmpty()) {
      System.out.println("No incidents found in this date range.");
   } else {
      rangeList.forEach(System.out::println);
   }
   break;

case 4:
   System.out.print("Enter Incident Type: ");
   String iType = sc.nextLine();
   List<Incident> found = (List<Incident>) service.searchIncidents(iType);
   found.forEach(System.out::println);
   break;

case 5:
   try {
      System.out.print("Enter Incident ID for report: ");
      int rid = sc.nextInt();
      Incident incForReport = new Incident();
      incForReport.setIncidentID(rid);
      System.out.print("Enter Officer ID: ");
      int reportOfficerId = sc.nextInt();
      incForReport.setOfficerID(reportOfficerId);
      sc.nextLine();
      System.out.print("Enter Type: ");
      incForReport.setIncidentType(sc.nextLine());
      System.out.print("Enter Description: ");
      incForReport.setDescription(sc.nextLine());

      Report report = service.generateIncidentReport(incForReport);
      if (report != null)
         System.out.println("Report generated:\n" + report);
      else
         System.out.println("Failed to generate report.");
   } catch (IncidentNumberNotFoundException e) {
      System.out.println(" " + e.getMessage());
   }
   break;

case 6:
   System.out.println("Exiting... Goodbye!");
   return;

default:
```

```
                        System.out.println("Invalid choice.");
                    }
                } catch (Exception e) {
                    System.out.println("Unexpected Error: " + e.getMessage());
                    e.printStackTrace();
                }

                System.out.print("Do you want to continue (yes/no)? ");
                continueChoice = sc.nextLine();

            } while (continueChoice.equalsIgnoreCase("yes"));

        } finally {
            sc.close();
            System.out.println("byebye...........");
        }
    }
}
```

8. Unit Testing

Creating JUnit test cases for a Crime Analysis and Reporting System is essential to ensure the correctness and reliability of your system. Below are some example questions to guide the creation of JUnit test cases for various components of the system:

Incident Creation:

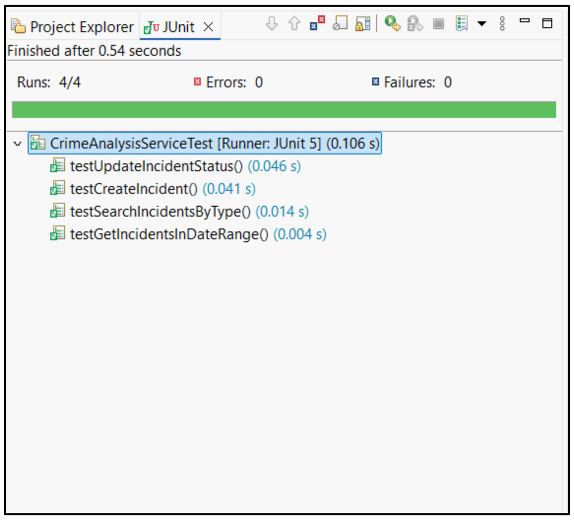
Does the createIncident method correctly create an incident with the provided attributes?

Are the attributes of the created incident accurate?

Incident Status Update:

Does the updateIncidentStatus method effectively update the status of an incident?

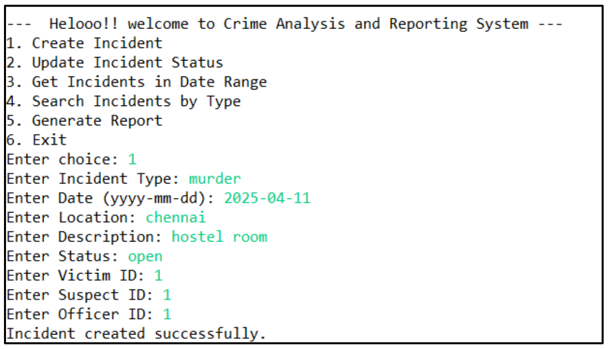
Does it handle invalid status updates appropriately?



```
Finished after 0.54 seconds
Runs: 4/4        Errors: 0        Failures: 0
CrimeAnalysisServiceTest [Runner: JUnit 5] (0.106 s)
    testUpdateIncidentStatus() (0.046 s)
    testCreateIncident() (0.041 s)
    testSearchIncidentsByType() (0.014 s)
    testGetIncidentsInDateRange() (0.004 s)
```

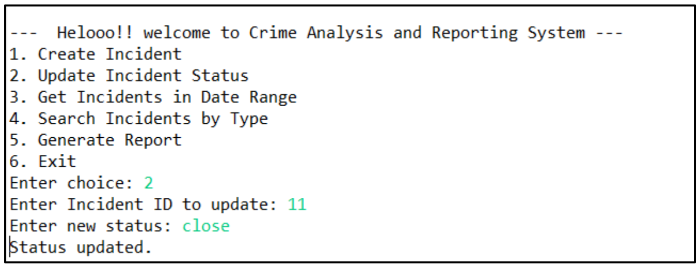
9. Final Output

- Create Incident



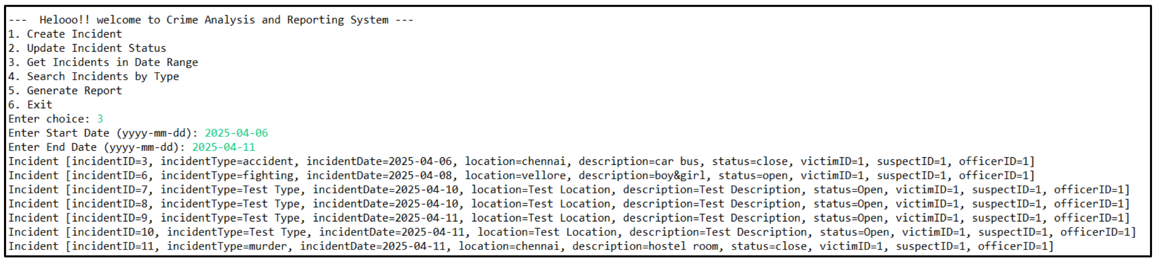
```
---  Helooo!! welcome to Crime Analysis and Reporting System ---
1. Create Incident
2. Update Incident Status
3. Get Incidents in Date Range
4. Search Incidents by Type
5. Generate Report
6. Exit
Enter choice: 1
Enter Incident Type: murder
Enter Date (yyyy-mm-dd): 2025-04-11
Enter Location: chennai
Enter Description: hostel room
Enter Status: open
Enter Victim ID: 1
Enter Suspect ID: 1
Enter Officer ID: 1
Incident created successfully.
```

- Update Incident Status



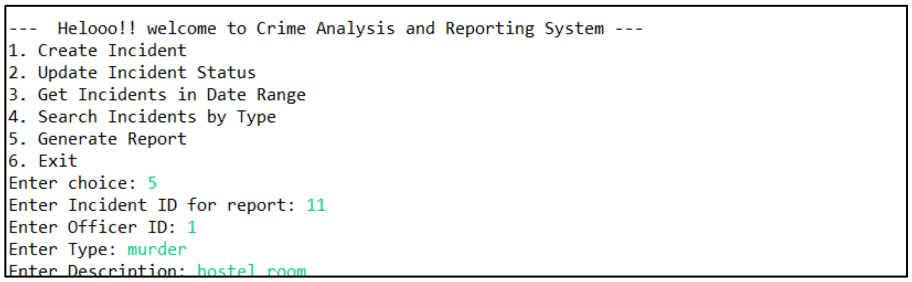
```
---  Helooo!! welcome to Crime Analysis and Reporting System ---
1. Create Incident
2. Update Incident Status
3. Get Incidents in Date Range
4. Search Incidents by Type
5. Generate Report
6. Exit
Enter choice: 2
Enter Incident ID to update: 11
Enter new status: close
Status updated.
```

- Get Incident in Date Range



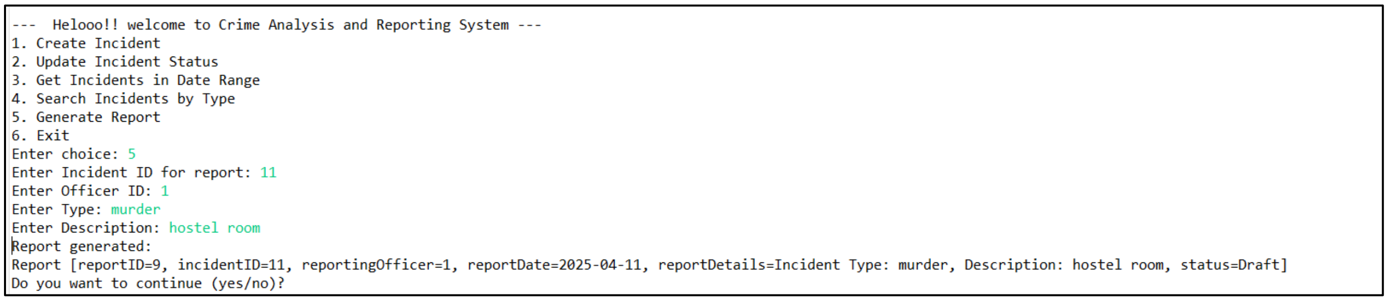
```
---  Helooo!! welcome to Crime Analysis and Reporting System ---
1. Create Incident
2. Update Incident Status
3. Get Incidents in Date Range
4. Search Incidents by Type
5. Generate Report
6. Exit
Enter choice: 3
Enter Start Date (yyyy-mm-dd): 2025-04-06
Enter End Date (yyyy-mm-dd): 2025-04-11
Incident [incidentID=3, incidentType=accident, incidentDate=2025-04-06, location=chennai, description=car bus, status=close, victimID=1, suspectID=1, officerID=1]
Incident [incidentID=6, incidentType=fighting, incidentDate=2025-04-08, location=vellore, description=boy&girl, status=open, victimID=1, suspectID=1, officerID=1]
Incident [incidentID=7, incidentType=Test Type, incidentDate=2025-04-10, location=Test Location, description=Test Description, status=Open, victimID=1, suspectID=1, officerID=1]
Incident [incidentID=8, incidentType=Test Type, incidentDate=2025-04-10, location=Test Location, description=Test Description, status=Open, victimID=1, suspectID=1, officerID=1]
Incident [incidentID=9, incidentType=Test Type, incidentDate=2025-04-11, location=Test Location, description=Test Description, status=Open, victimID=1, suspectID=1, officerID=1]
Incident [incidentID=10, incidentType=Test Type, incidentDate=2025-04-11, location=Test Location, description=Test Description, status=Open, victimID=1, suspectID=1, officerID=1]
Incident [incidentID=11, incidentType=murder, incidentDate=2025-04-11, location=chennai, description=hostel room, status=close, victimID=1, suspectID=1, officerID=1]
```

- Search Incident by Type



```
---  Helooo!! welcome to Crime Analysis and Reporting System ---
1. Create Incident
2. Update Incident Status
3. Get Incidents in Date Range
4. Search Incidents by Type
5. Generate Report
6. Exit
Enter choice: 5
Enter Incident ID for report: 11
Enter Officer ID: 1
Enter Type: murder
Enter Description: hostel room
```

➢ Generate Report



```
---  Helooo!! welcome to Crime Analysis and Reporting System ---
1. Create Incident
2. Update Incident Status
3. Get Incidents in Date Range
4. Search Incidents by Type
5. Generate Report
6. Exit
Enter choice: 5
Enter Incident ID for report: 11
Enter Officer ID: 1
Enter Type: murder
Enter Description: hostel room
Report generated:
Report [reportID=9, incidentID=11, reportingOfficer=1, reportDate=2025-04-11, reportDetails=Incident Type: murder, Description: hostel room, status=Draft]
Do you want to continue (yes/no)?
```