

Case Study on Asset Management System

Team 7

Melvin Jones E

Navadharshini J

Project Overview :

The Digital Asset Management System is designed to streamline the tracking, maintenance, and allocation of assets within an organization. The application allows users to perform CRUD operations on assets, track their maintenance history, allocate them to employees, and manage reservations. The project was developed using Java with JDBC for database interaction, SQL for the schema, and JUnit for testing.

Schema Design :

Created tables for employees, assets, Maintenance Records, Reservations, Asset Allocations.

Employees Table:

```
mysql> desc employees;
```

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
department	varchar(50)	YES		NULL	
email	varchar(100)	YES		NULL	
password	varchar(255)	YES		NULL	

5 rows in set (0.02 sec)

Assets Table schema :

```
mysql> desc assets;
```

Field	Type	Null	Key	Default	Extra
asset_id	int	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
type	varchar(50)	YES		NULL	
serial_number	varchar(30)	YES	UNI	NULL	
purchase_date	date	YES		NULL	
location	varchar(50)	YES		NULL	
status	varchar(50)	YES		NULL	
owner_id	int	YES	MUL	NULL	

8 rows in set (0.01 sec)

Asset Allocations schema :

```
mysql> desc asset_allocations;
```

Field	Type	Null	Key	Default	Extra
allocation_id	int	NO	PRI	NULL	auto_increment
asset_id	int	YES	MUL	NULL	
employee_id	int	YES	MUL	NULL	
allocation_date	date	YES		NULL	
return_date	date	YES		NULL	

5 rows in set (0.01 sec)

Maintenance Records Schema :

```
mysql> desc maintenance_records;
```

Field	Type	Null	Key	Default	Extra
maintenance_id	int	NO	PRI	NULL	auto_increment
asset_id	int	YES	MUL	NULL	
maintenance_date	date	YES		NULL	
description	varchar(255)	YES		NULL	
cost	decimal(10,2)	YES		NULL	

5 rows in set (0.01 sec)

Reservations Schema :

```
mysql> desc reservations;
```

Field	Type	Null	Key	Default	Extra
reservation_id	int	NO	PRI	NULL	auto_increment
asset_id	int	YES	MUL	NULL	
employee_id	int	YES	MUL	NULL	
reservation_date	date	YES		NULL	
start_date	date	YES		NULL	
end_date	date	YES		NULL	
status	varchar(50)	YES		NULL	

7 rows in set (0.01 sec)

Project Structure :

1. **Entity** : Created Entity package to these classes and added getter, setter methods

- Employee Class
- Asset Class
- Asset Allocation Class
- Perform Maintenance Class
- Reservation Class

2. **Dao** : Created Interface called AssetManagementService and Implementation file called AssetMangementServiceImpl to implement the interface methods.

- AssetManagementService Interface
- AssetMangementServiceImpl class

3. **Exception** : Created exceptions for Asset, Employee not found and check if asset is not maintained for 2 years.

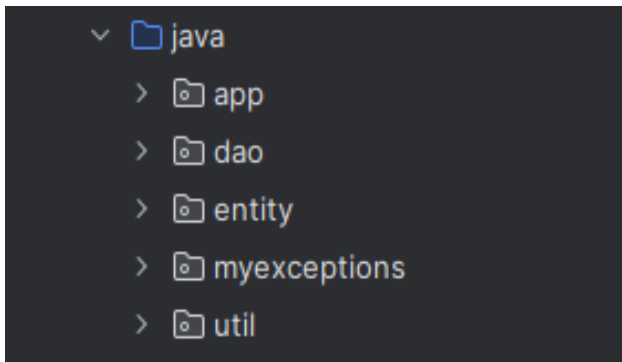
- AssetNotFound Exception
- AssetNotMaintain Exception
- EmployeeNotFound Exception

4. **Util** : Created Util package to Connect to the database.

- DBConnUtil class
- DBPropertyUtil class

5. **Main** : Main class to trigger all the methods and get input from user

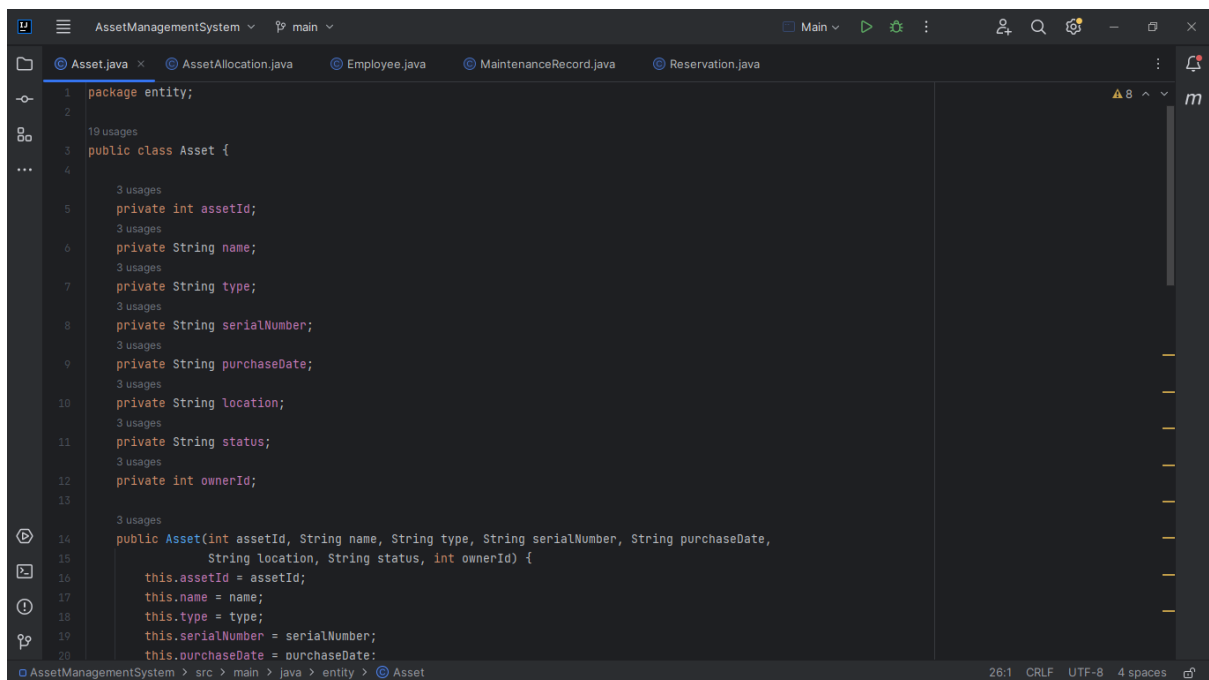
- Main class



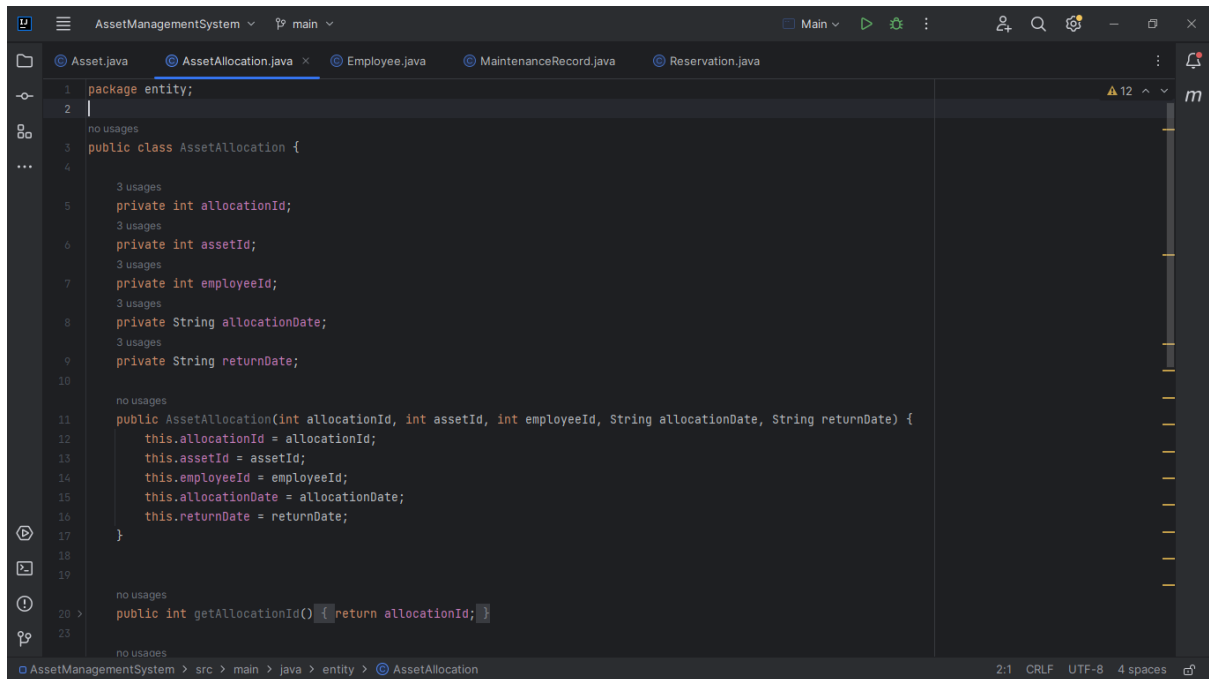
Entity Package :

Created entity package for Employee, Assets, Asset Allocations, Maintenance Records and Reservations.

Asset.java



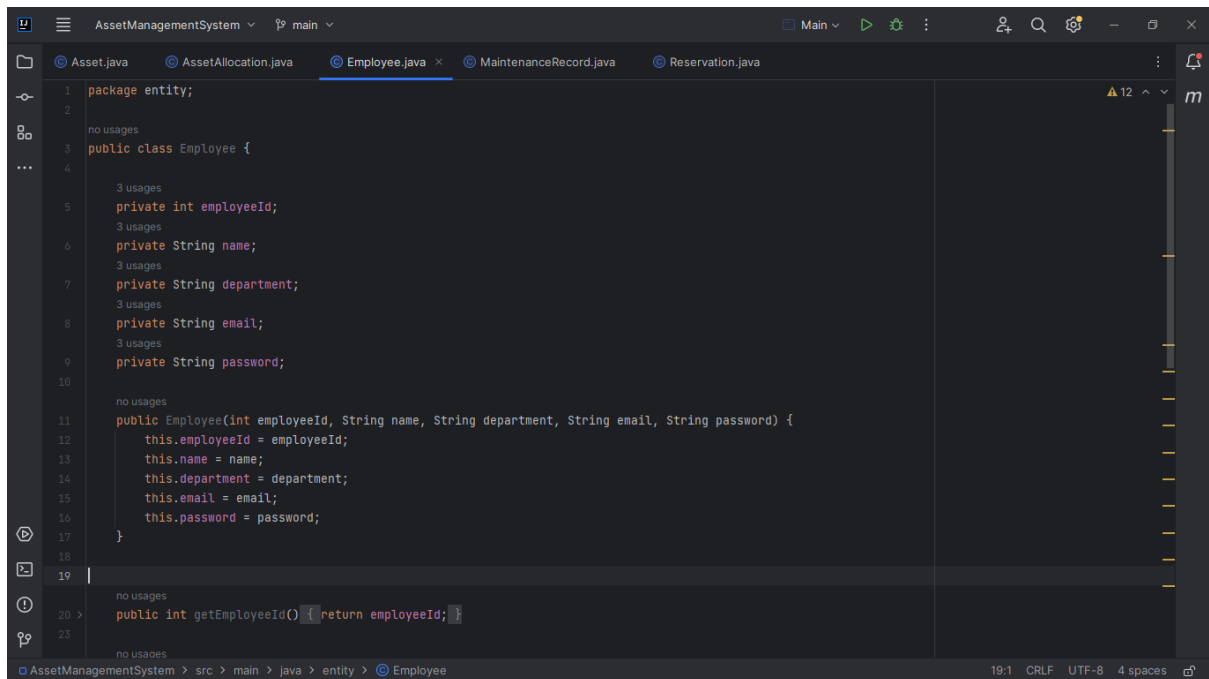
AssetAllocation.java



```
1 package entity;
2
3 no usages
4 public class AssetAllocation {
5     3 usages
6     private int allocationId;
7     3 usages
8     private int assetId;
9     3 usages
10    private int employeeId;
11    3 usages
12    private String allocationDate;
13    3 usages
14    private String returnDate;
15
16    no usages
17    public AssetAllocation(int allocationId, int assetId, int employeeId, String allocationDate, String returnDate) {
18        this.allocationId = allocationId;
19        this.assetId = assetId;
20        this.employeeId = employeeId;
21        this.allocationDate = allocationDate;
22        this.returnDate = returnDate;
23    }
24
25    no usages
26    public int getAllocationId() { return allocationId; }
27
28    no usages
29 }
```

AssetManagementSystem > src > main > java > entity > AssetAllocation 2:1 CRLF UTF-8 4 spaces

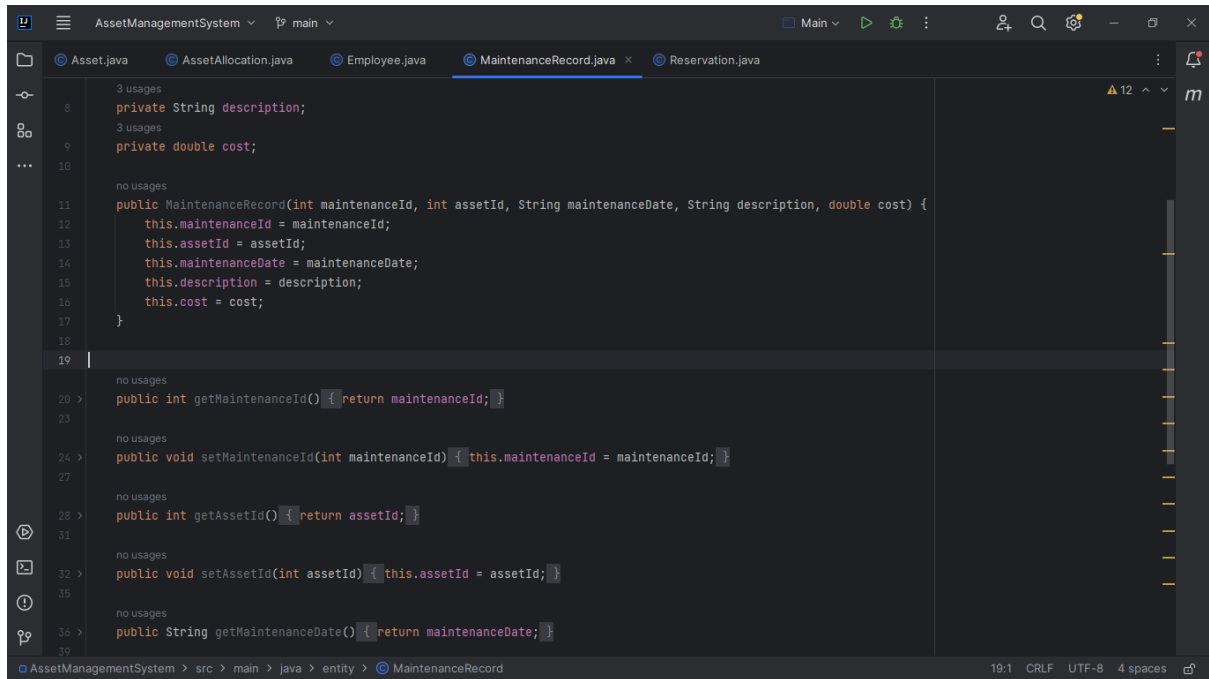
Employee.java



```
1 package entity;
2
3 no usages
4 public class Employee {
5     3 usages
6     private int employeeId;
7     3 usages
8     private String name;
9     3 usages
10    private String department;
11    3 usages
12    private String email;
13    3 usages
14    private String password;
15
16    no usages
17    public Employee(int employeeId, String name, String department, String email, String password) {
18        this.employeeId = employeeId;
19        this.name = name;
20        this.department = department;
21        this.email = email;
22        this.password = password;
23    }
24
25    no usages
26    public int getEmployeeId() { return employeeId; }
27
28    no usages
29 }
```

AssetManagementSystem > src > main > java > entity > Employee 19:1 CRLF UTF-8 4 spaces

MaintenanceRecords.java



```
AssetManagementSystem > src > main > java > entity > MaintenanceRecord

19:1 CRLF UTF-8 4 spaces
```

```
3 usages
private String description;
3 usages
private double cost;

no usages
public MaintenanceRecord(int maintenanceId, int assetId, String maintenanceDate, String description, double cost) {
    this.maintenanceId = maintenanceId;
    this.assetId = assetId;
    this.maintenanceDate = maintenanceDate;
    this.description = description;
    this.cost = cost;
}

no usages
public int getMaintenanceId() { return maintenanceId; }

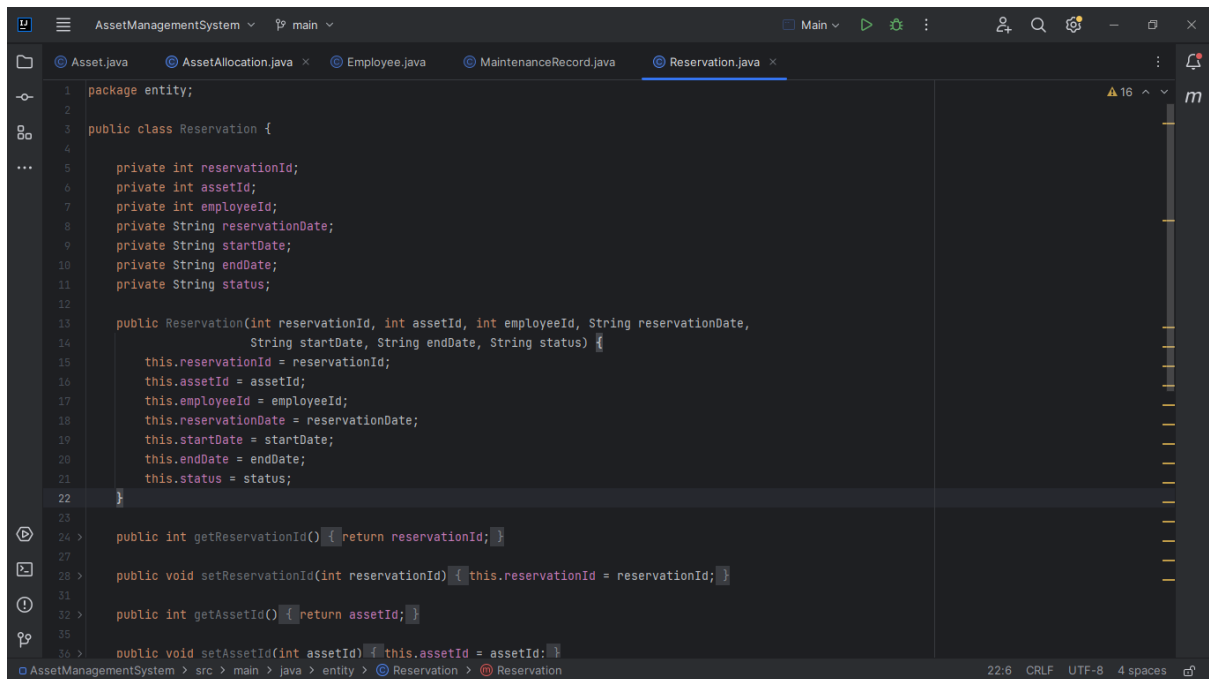
no usages
public void setMaintenanceId(int maintenanceId) { this.maintenanceId = maintenanceId; }

no usages
public int getAssetId() { return assetId; }

no usages
public void setAssetId(int assetId) { this.assetId = assetId; }

no usages
public String getMaintenanceDate() { return maintenanceDate; }
```

Reservation.java



```
AssetManagementSystem > src > main > java > entity > Reservation > Reservation

22:6 CRLF UTF-8 4 spaces
```

```
package entity;

public class Reservation {

    private int reservationId;
    private int assetId;
    private int employeeId;
    private String reservationDate;
    private String startDate;
    private String endDate;
    private String status;

    public Reservation(int reservationId, int assetId, int employeeId, String reservationDate,
        String startDate, String endDate, String status) {
        this.reservationId = reservationId;
        this.assetId = assetId;
        this.employeeId = employeeId;
        this.reservationDate = reservationDate;
        this.startDate = startDate;
        this.endDate = endDate;
        this.status = status;
    }

    public int getReservationId() { return reservationId; }

    public void setReservationId(int reservationId) { this.reservationId = reservationId; }

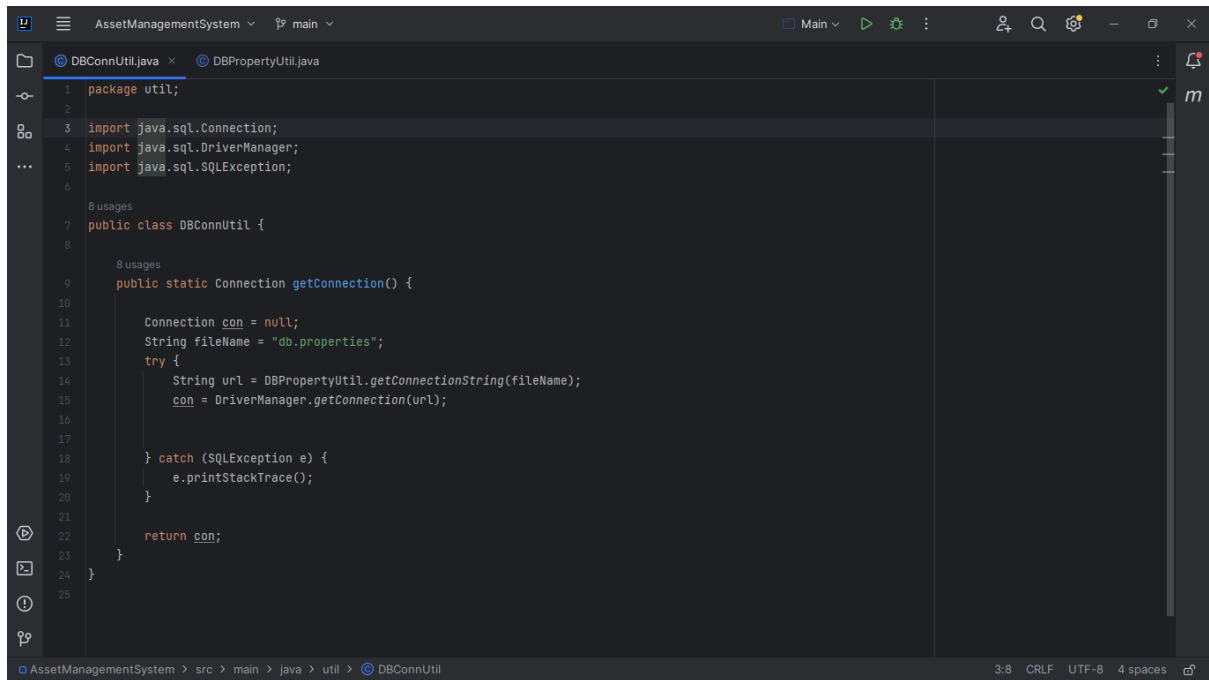
    public int getAssetId() { return assetId; }

    public void setAssetId(int assetId) { this.assetId = assetId; }
```

Util Package :

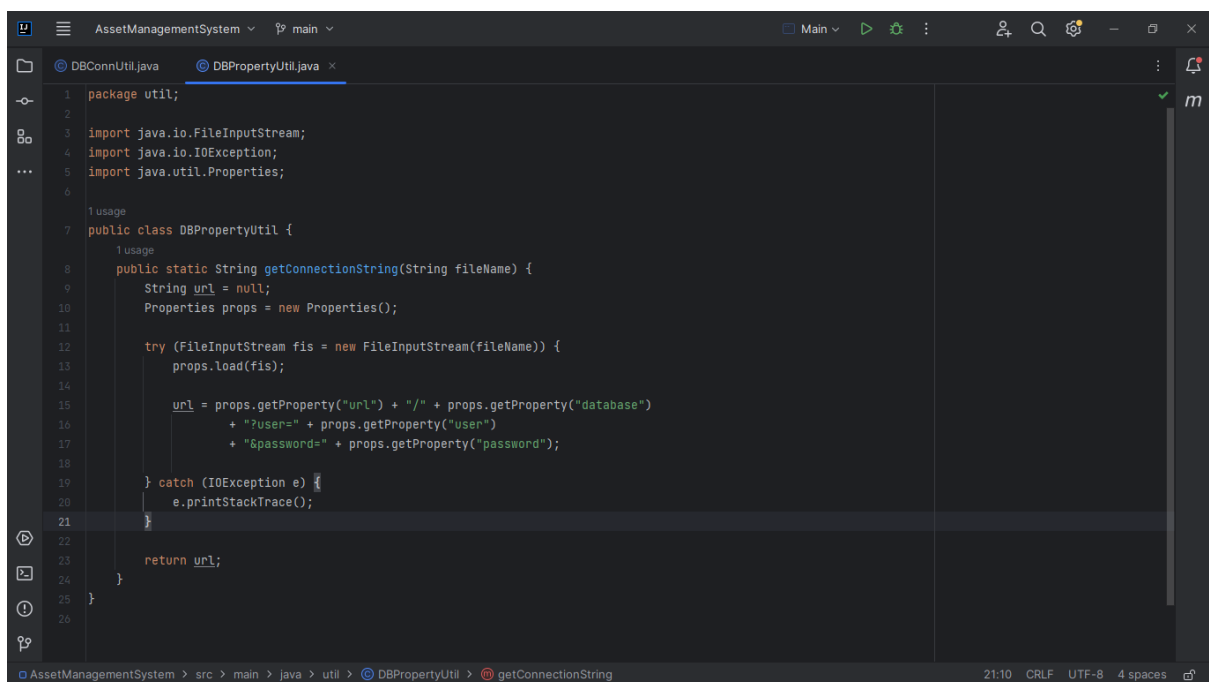
Created Util package to establish connection between database with the help of Maven dependency.

DBConnUtil.java



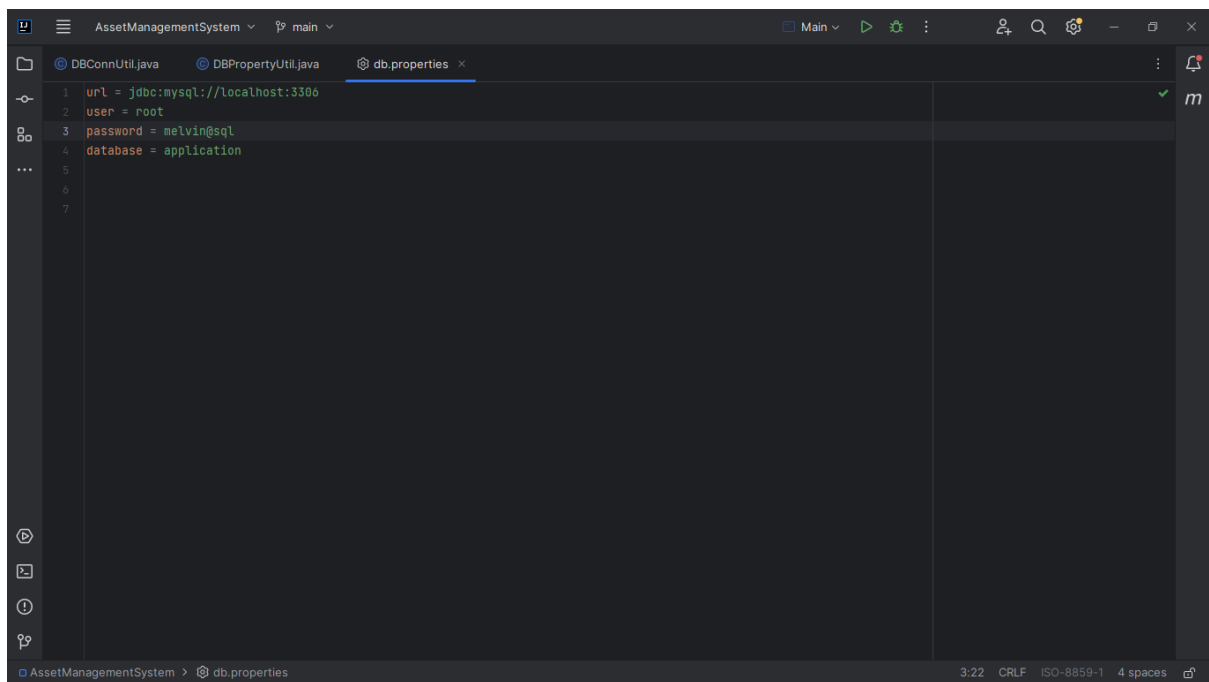
```
1 package util;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DBConnUtil {
8
9     public static Connection getConnection() {
10
11         Connection con = null;
12         String fileName = "db.properties";
13         try {
14             String url = DBPropertyUtil.getConnectionString(fileName);
15             con = DriverManager.getConnection(url);
16
17         } catch (SQLException e) {
18             e.printStackTrace();
19         }
20
21         return con;
22     }
23 }
24
25
```

DBPropertyUtil.java



```
1 package util;
2
3 import java.io.FileInputStream;
4 import java.io.IOException;
5 import java.util.Properties;
6
7 public class DBPropertyUtil {
8
9     public static String getConnectionString(String fileName) {
10
11         String url = null;
12         Properties props = new Properties();
13
14         try (FileInputStream fis = new FileInputStream(fileName)) {
15             props.load(fis);
16
17             url = props.getProperty("url") + "/" + props.getProperty("database")
18                 + "?user=" + props.getProperty("user")
19                 + "&password=" + props.getProperty("password");
20
21         } catch (IOException e) {
22             e.printStackTrace();
23         }
24
25         return url;
26     }
27 }
28
```


db.properties file to store data of database



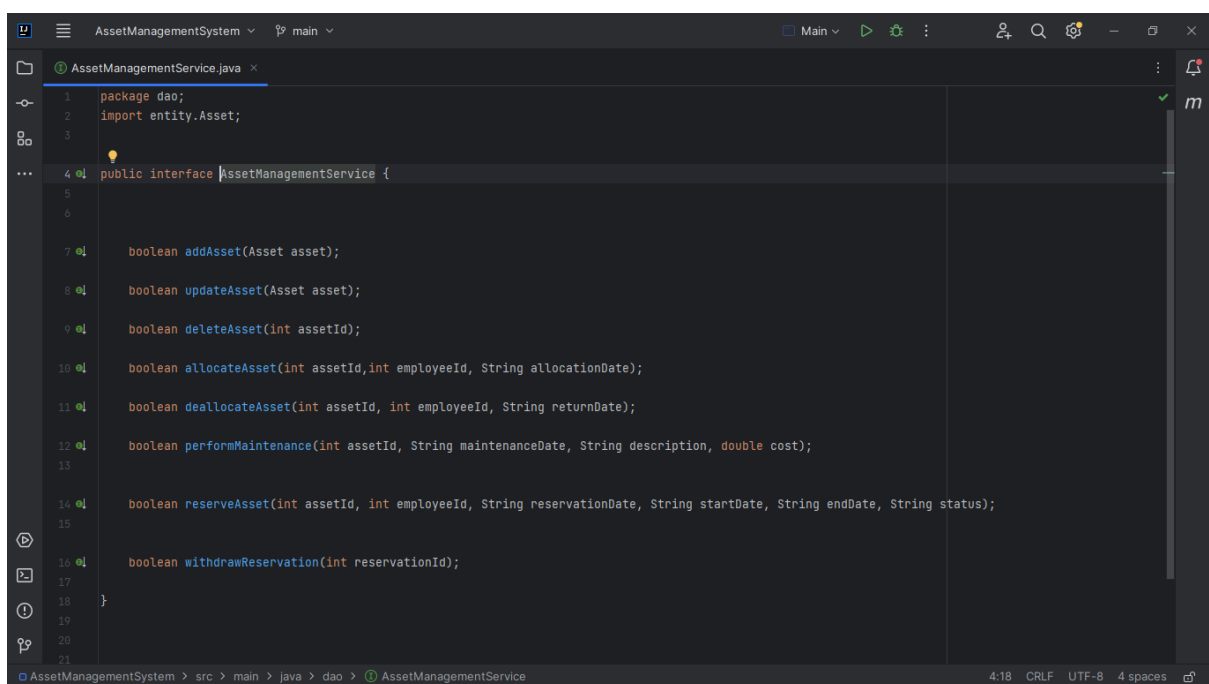
```
1 url = jdbc:mysql://localhost:3306
2 user = root
3 password = melvin@sql
4 database = application
5
6
7
```

The screenshot shows an IDE window titled 'AssetManagementSystem' with a tab for 'db.properties'. The file contains four lines of database connection properties: url, user, password, and database. The status bar at the bottom indicates the file is encoded in ISO-8859-1 with 4 spaces.

Dao Package :

Created an Interface **AssetManagementService** to show methods to be implemented

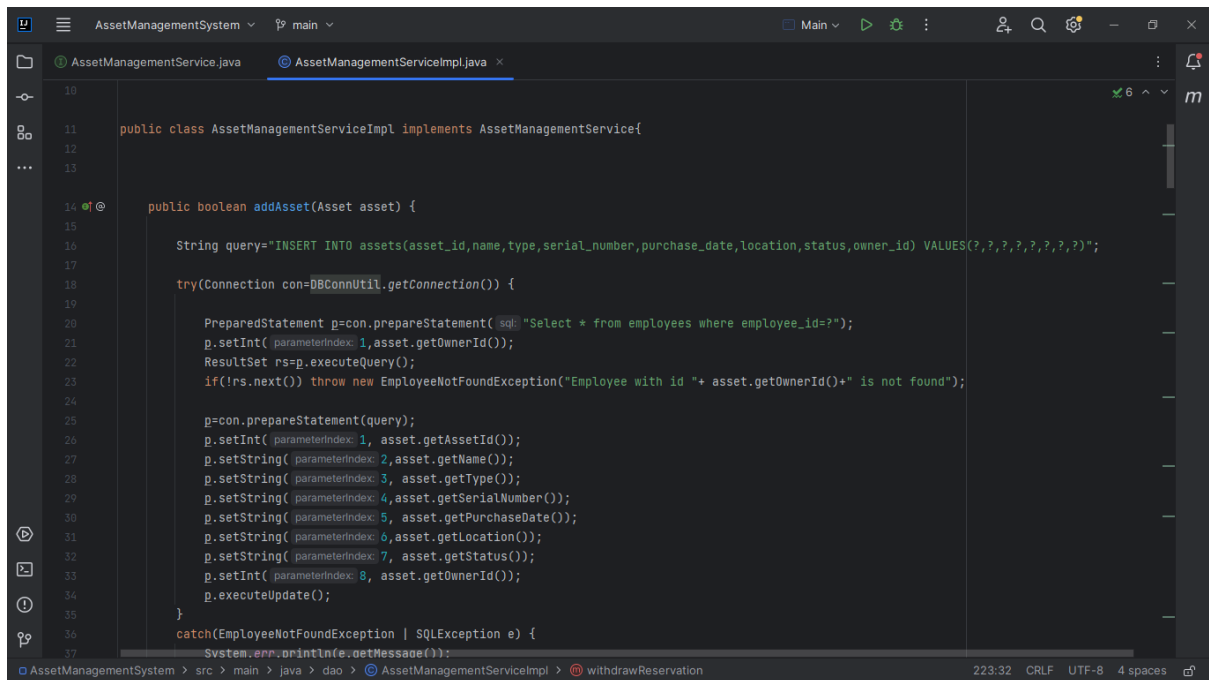
AssetManagementService.java



```
1 package dao;
2 import entity.Asset;
3
4 public interface AssetManagementService {
5
6
7     boolean addAsset(Asset asset);
8     boolean updateAsset(Asset asset);
9     boolean deleteAsset(int assetId);
10    boolean allocateAsset(int assetId,int employeeId, String allocationDate);
11    boolean deallocateAsset(int assetId, int employeeId, String returnDate);
12    boolean performMaintenance(int assetId, String maintenanceDate, String description, double cost);
13
14    boolean reserveAsset(int assetId, int employeeId, String reservationDate, String startDate, String endDate, String status);
15
16    boolean withdrawReservation(int reservationId);
17 }
18
19
20
21
```

The screenshot shows an IDE window titled 'AssetManagementSystem' with a tab for 'AssetManagementService.java'. The file defines a public interface with ten methods for managing assets. The status bar at the bottom indicates the file is encoded in UTF-8 with 4 spaces.

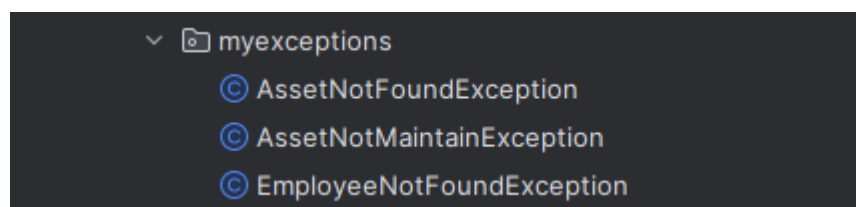
And implemented those methods in AssetManagementImpl class by inheriting the interface .



```
18
19
20 public class AssetManagementServiceImpl implements AssetManagementService{
21
22
23
24
25     public boolean addAsset(Asset asset) {
26
27         String query="INSERT INTO assets(asset_id,name,type,serial_number,purchase_date,location,status,owner_id) VALUES(?,?,?,?,?,?,?,?)";
28
29         try(Connection con=DBConnUtil.getConnection()) {
30
31             PreparedStatement p=con.prepareStatement("select * from employees where employee_id=?");
32             p.setInt(1,asset.getOwnerId());
33             ResultSet rs=p.executeQuery();
34             if(!rs.next()) throw new EmployeeNotFoundException("Employee with id "+ asset.getOwnerId()+" is not found");
35
36             p=con.prepareStatement(query);
37             p.setInt(1, asset.getAssetId());
38             p.setString(2,asset.getName());
39             p.setString(3, asset.getType());
40             p.setString(4,asset.getSerialNumber());
41             p.setString(5, asset.getPurchaseDate());
42             p.setString(6,asset.getLocation());
43             p.setString(7, asset.getStatus());
44             p.setInt(8, asset.getOwnerId());
45             p.executeUpdate();
46         }
47         catch(EmployeeNotFoundException | SQLException e) {
48             System.err.println(e.getMessage());
49         }
50     }
51 }
```

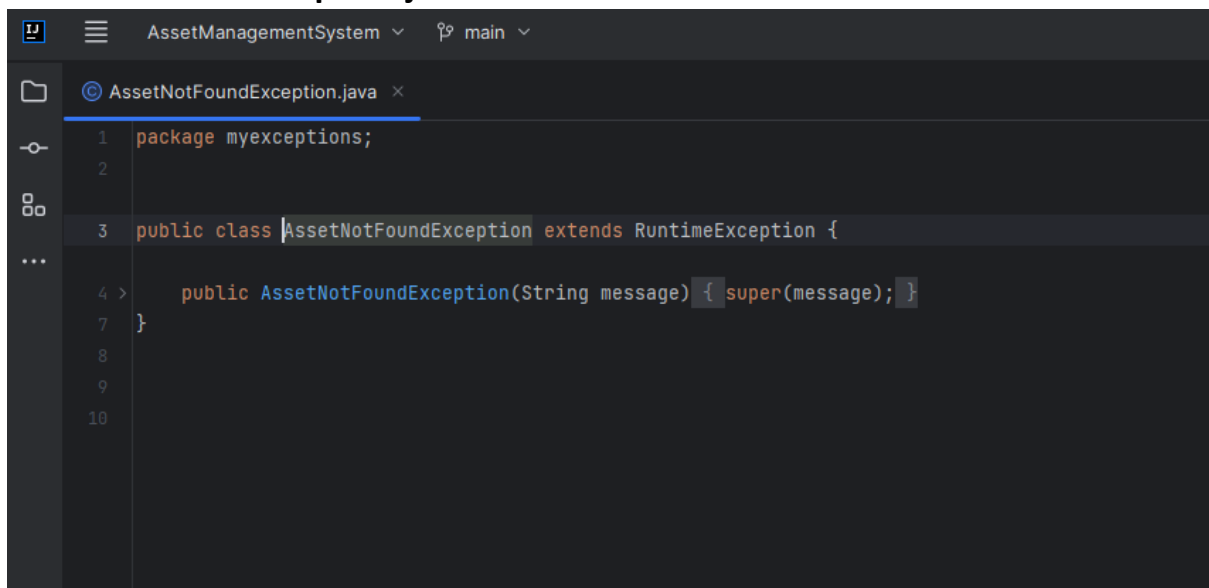
Myexceptions Package

Created User defined exceptions whenever an Asset Id or Employee Id is not found, By inheriting the Exception class.



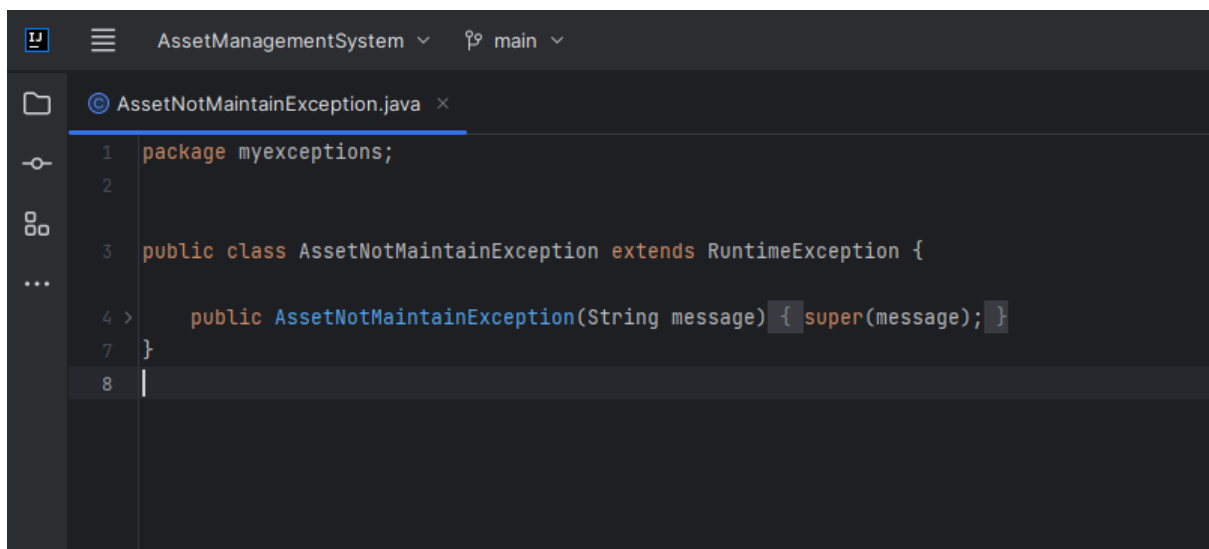
```
myexceptions
├── AssetNotFoundException
├── AssetNotMaintainException
└── EmployeeNotFoundException
```

AssetNotFoundException.java



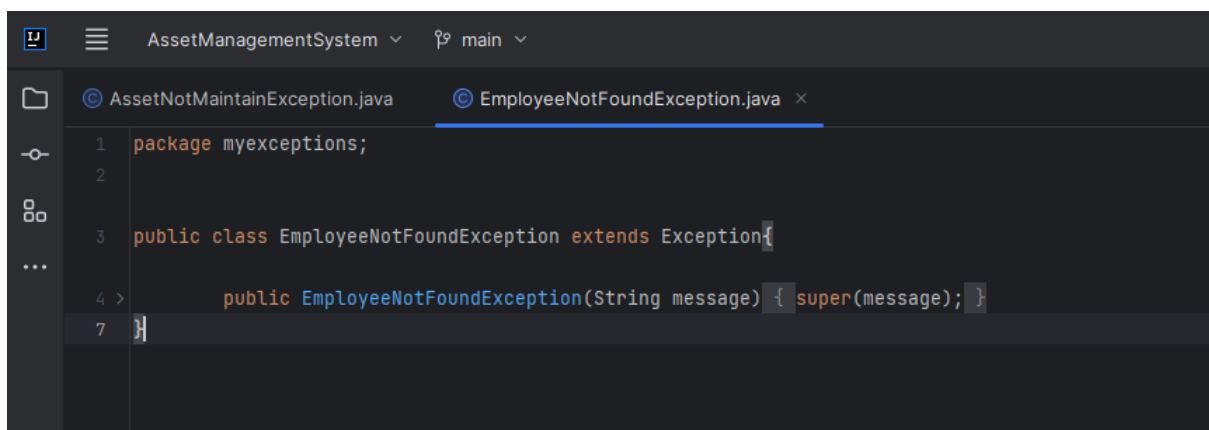
```
1 package myexceptions;
2
3 public class AssetNotFoundException extends RuntimeException {
4     public AssetNotFoundException(String message) { super(message); }
5 }
6
7
8
9
10
```

AssetNotMaintainException.java



```
1 package myexceptions;
2
3 public class AssetNotMaintainException extends RuntimeException {
4     public AssetNotMaintainException(String message) { super(message); }
5 }
6
7
8
```

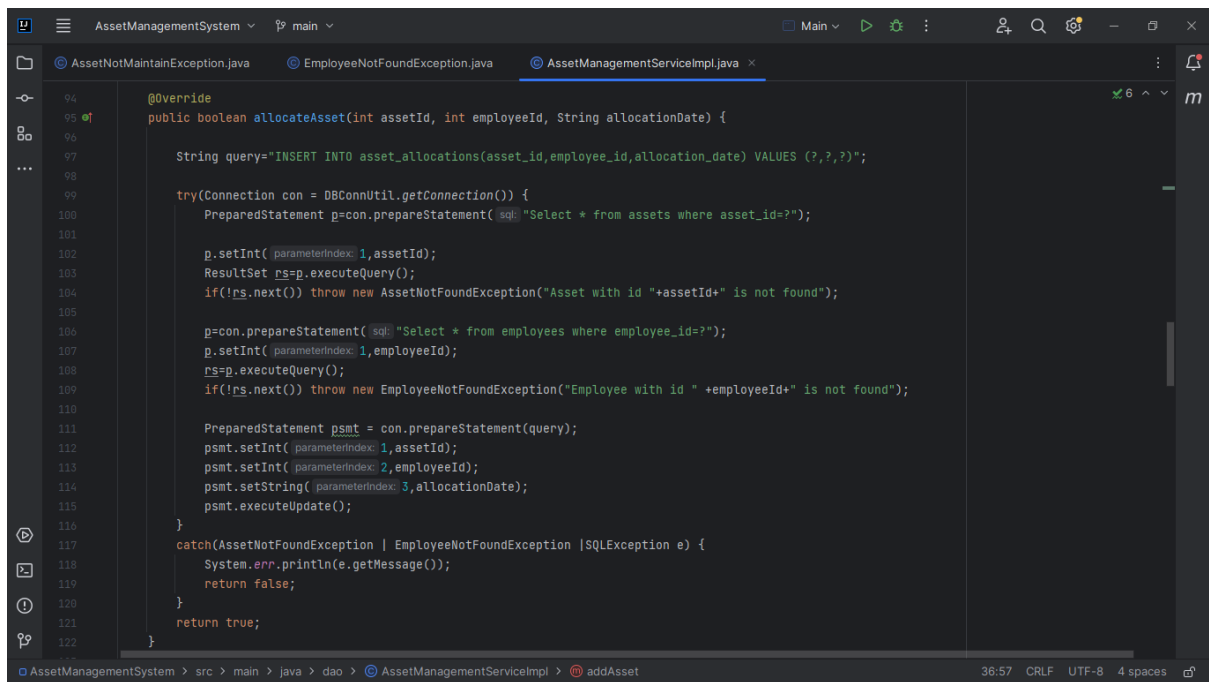
EmployeeNotFoundException.java



```
1 package myexceptions;
2
3 public class EmployeeNotFoundException extends Exception {
4     public EmployeeNotFoundException(String message) { super(message); }
5 }
6
7
```

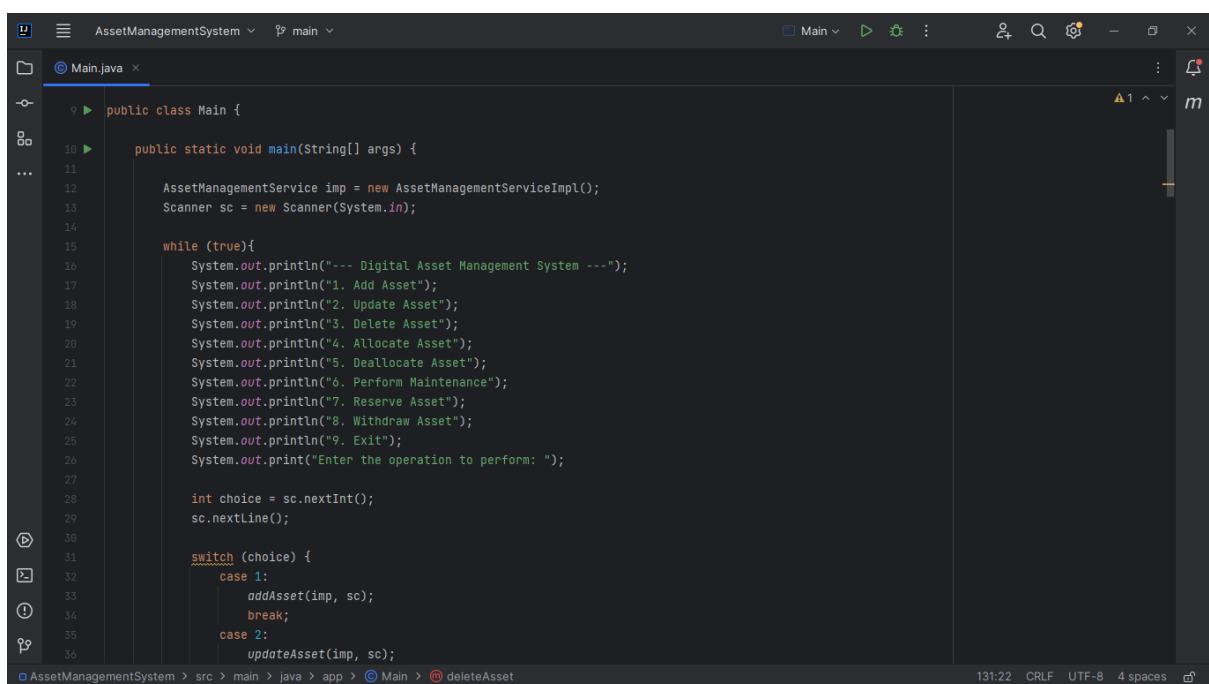
Added these Exceptions in required methods , which might throw errors and caught them .

Example :



```
94  @Override
95  public boolean allocateAsset(int assetId, int employeeId, String allocationDate) {
96
97      String query="INSERT INTO asset_allocations(asset_id,employee_id,allocation_date) VALUES (?,?,?)";
98
99      try(Connection con = DBConnUtil.getConnection()) {
100          PreparedStatement p=con.prepareStatement("Select * from assets where asset_id=?");
101
102          p.setInt( parameterIndex: 1,assetId);
103          ResultSet rs=p.executeQuery();
104          if(!rs.next()) throw new AssetNotFoundException("Asset with id "+assetId+" is not found");
105
106          p=con.prepareStatement("Select * from employees where employee_id=?");
107          p.setInt( parameterIndex: 1,employeeId);
108          rs=p.executeQuery();
109          if(!rs.next()) throw new EmployeeNotFoundException("Employee with id " +employeeId+" is not found");
110
111          PreparedStatement psmt = con.prepareStatement(query);
112          psmt.setInt( parameterIndex: 1,assetId);
113          psmt.setInt( parameterIndex: 2,employeeId);
114          psmt.setString( parameterIndex: 3,allocationDate);
115          psmt.executeUpdate();
116      }
117      catch(AssetNotFoundException | EmployeeNotFoundException |SQLException e) {
118          System.err.println(e.getMessage());
119          return false;
120      }
121      return true;
122  }
```

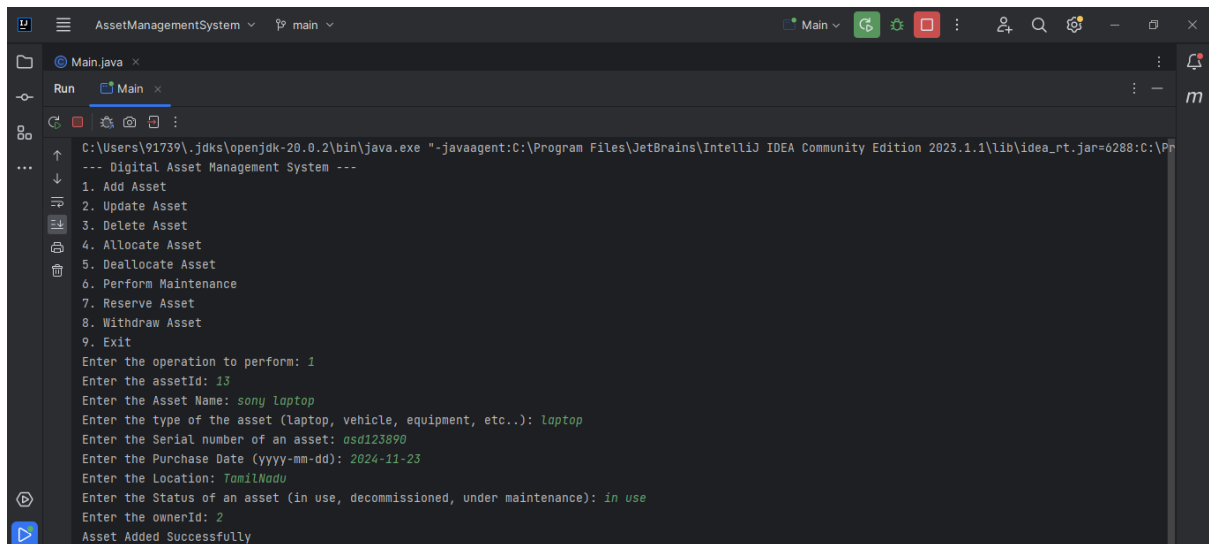
Created Main.java File to trigger the Implemented Methods.



```
10  public class Main {
11
12      public static void main(String[] args) {
13
14          AssetManagementService imp = new AssetManagementServiceImpl();
15          Scanner sc = new Scanner(System.in);
16
17          while (true){
18              System.out.println("--- Digital Asset Management System ---");
19              System.out.println("1. Add Asset");
20              System.out.println("2. Update Asset");
21              System.out.println("3. Delete Asset");
22              System.out.println("4. Allocate Asset");
23              System.out.println("5. Deallocate Asset");
24              System.out.println("6. Perform Maintenance");
25              System.out.println("7. Reserve Asset");
26              System.out.println("8. Withdraw Asset");
27              System.out.println("9. Exit");
28              System.out.print("Enter the operation to perform: ");
29
30              int choice = sc.nextInt();
31              sc.nextLine();
32
33              switch (choice) {
34                  case 1:
35                      addAsset(imp, sc);
36                      break;
37                  case 2:
38                      updateAsset(imp, sc);
39                      break;
34                  case 3:
35                      deleteAsset(imp, sc);
36                      break;
37                  case 4:
38                      allocateAsset(imp, sc);
39                      break;
40                  case 5:
41                      deallocateAsset(imp, sc);
42                      break;
43                  case 6:
44                      performMaintenance(imp, sc);
45                      break;
46                  case 7:
47                      reserveAsset(imp, sc);
48                      break;
49                  case 8:
50                      withdrawAsset(imp, sc);
51                      break;
52                  case 9:
53                      exit();
54                      break;
55              }
56          }
57      }
58  }
```

Results :

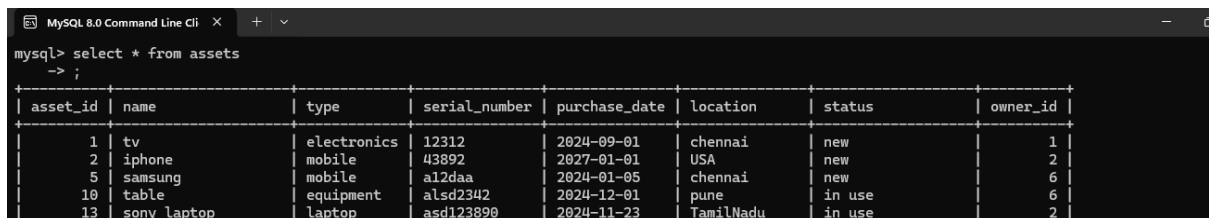
Case 1 : Add



The screenshot shows the IntelliJ IDEA interface with a Java application running. The console output displays a menu for the Digital Asset Management System with options 1 through 9. Option 1, 'Add Asset', is selected. The user is prompted to enter the assetId (13), Asset Name (sony laptop), type (laptop), serial number (asd123890), purchase date (2024-11-23), location (TamilNadu), status (in use), and ownerId (2). The final output is 'Asset Added Successfully'.

```
C:\Users\91739\jdk\openjdk-20.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.1\lib\idea_rt.jar=6288:C:\Pr
--- Digital Asset Management System ---
1. Add Asset
2. Update Asset
3. Delete Asset
4. Allocate Asset
5. Deallocate Asset
6. Perform Maintenance
7. Reserve Asset
8. Withdraw Asset
9. Exit
Enter the operation to perform: 1
Enter the assetId: 13
Enter the Asset Name: sony laptop
Enter the type of the asset (Laptop, vehicle, equipment, etc.): laptop
Enter the Serial number of an asset: asd123890
Enter the Purchase Date (yyyy-mm-dd): 2024-11-23
Enter the Location: TamilNadu
Enter the Status of an asset (in use, decommissioned, under maintenance): in use
Enter the ownerId: 2
Asset Added Successfully
```

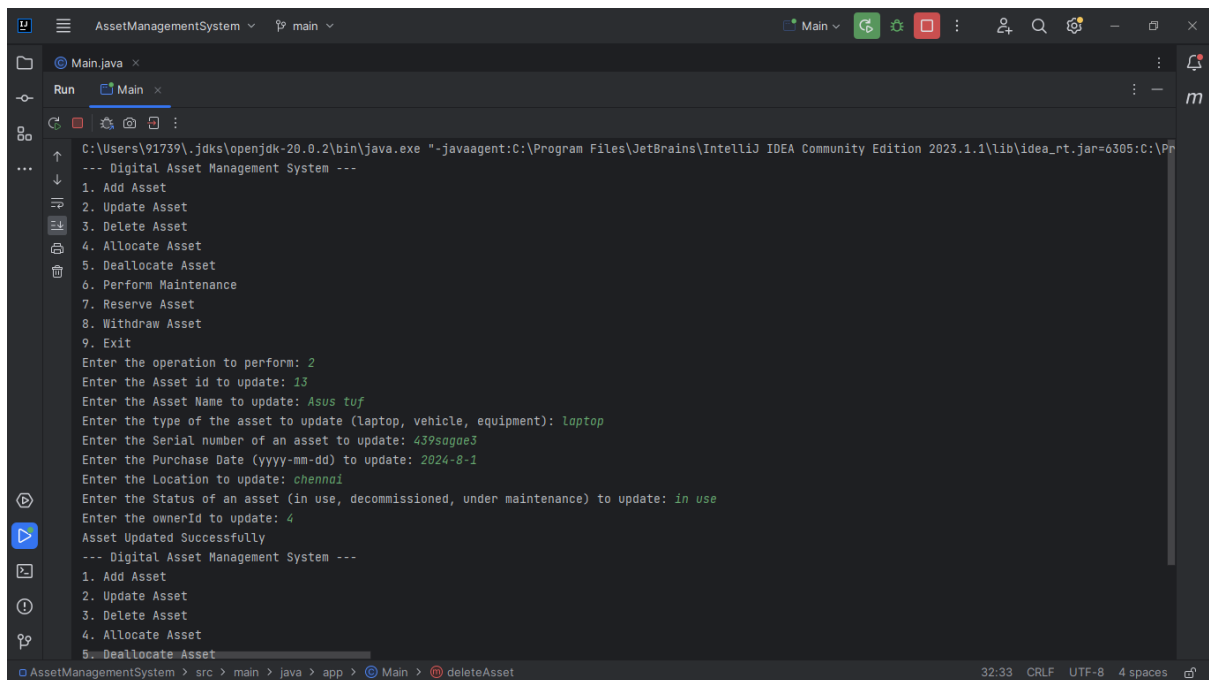
Database check whether the asset added or not



The screenshot shows the MySQL 8.0 Command Line Client with the command 'select * from assets' executed. The output is a table with 8 columns: asset_id, name, type, serial_number, purchase_date, location, status, and owner_id. The table contains 5 rows of data, including the newly added asset with id 13.

asset_id	name	type	serial_number	purchase_date	location	status	owner_id
1	tv	electronics	12312	2024-09-01	chennai	new	1
2	iphone	mobile	43892	2027-01-01	USA	new	2
5	samsung	mobile	a12daa	2024-01-05	chennai	new	6
10	table	equipment	alsd2342	2024-12-01	pune	in use	6
13	sony laptop	laptop	asd123890	2024-11-23	TamilNadu	in use	2

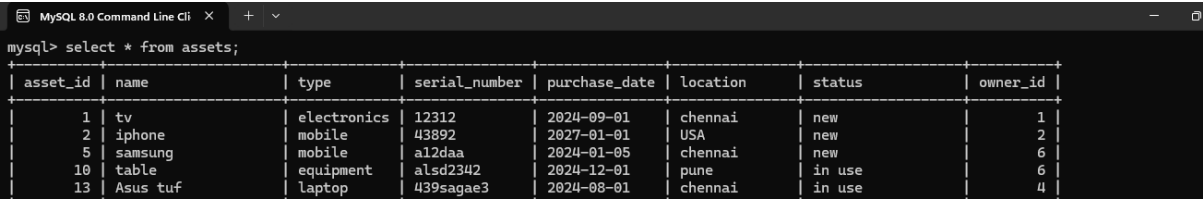
Case 2 : Update



The screenshot shows the IntelliJ IDEA interface with a Java application running. The console output displays the same menu as Case 1. Option 2, 'Update Asset', is selected. The user is prompted to enter the asset id to update (13), the Asset Name to update (Asus tuf), the type of the asset to update (laptop), the serial number of an asset to update (439sagae3), the purchase date (2024-8-1), the location (chennai), the status of an asset (in use), and the ownerId (4). The final output is 'Asset Updated Successfully'.

```
C:\Users\91739\jdk\openjdk-20.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.1\lib\idea_rt.jar=6305:C:\Pr
--- Digital Asset Management System ---
1. Add Asset
2. Update Asset
3. Delete Asset
4. Allocate Asset
5. Deallocate Asset
6. Perform Maintenance
7. Reserve Asset
8. Withdraw Asset
9. Exit
Enter the operation to perform: 2
Enter the Asset id to update: 13
Enter the Asset Name to update: Asus tuf
Enter the type of the asset to update (Laptop, vehicle, equipment): laptop
Enter the Serial number of an asset to update: 439sagae3
Enter the Purchase Date (yyyy-mm-dd) to update: 2024-8-1
Enter the Location to update: chennai
Enter the Status of an asset (in use, decommissioned, under maintenance) to update: in use
Enter the ownerId to update: 4
Asset Updated Successfully
--- Digital Asset Management System ---
1. Add Asset
2. Update Asset
3. Delete Asset
4. Allocate Asset
5. Deallocate Asset
```

Database Check

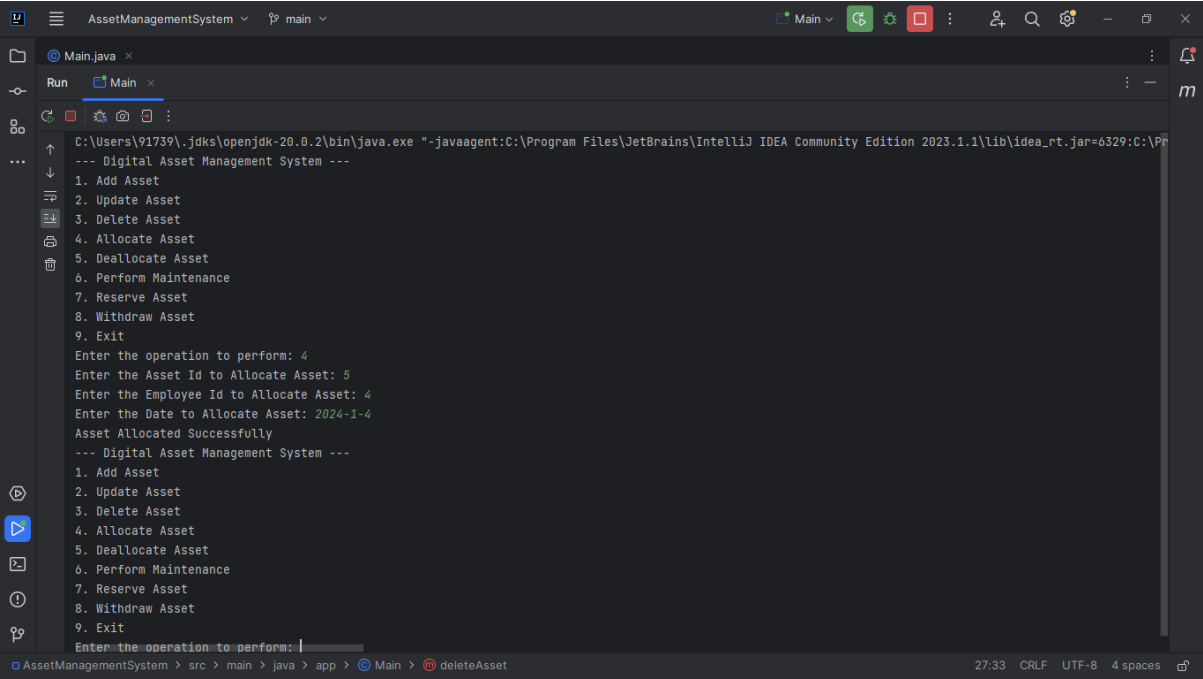
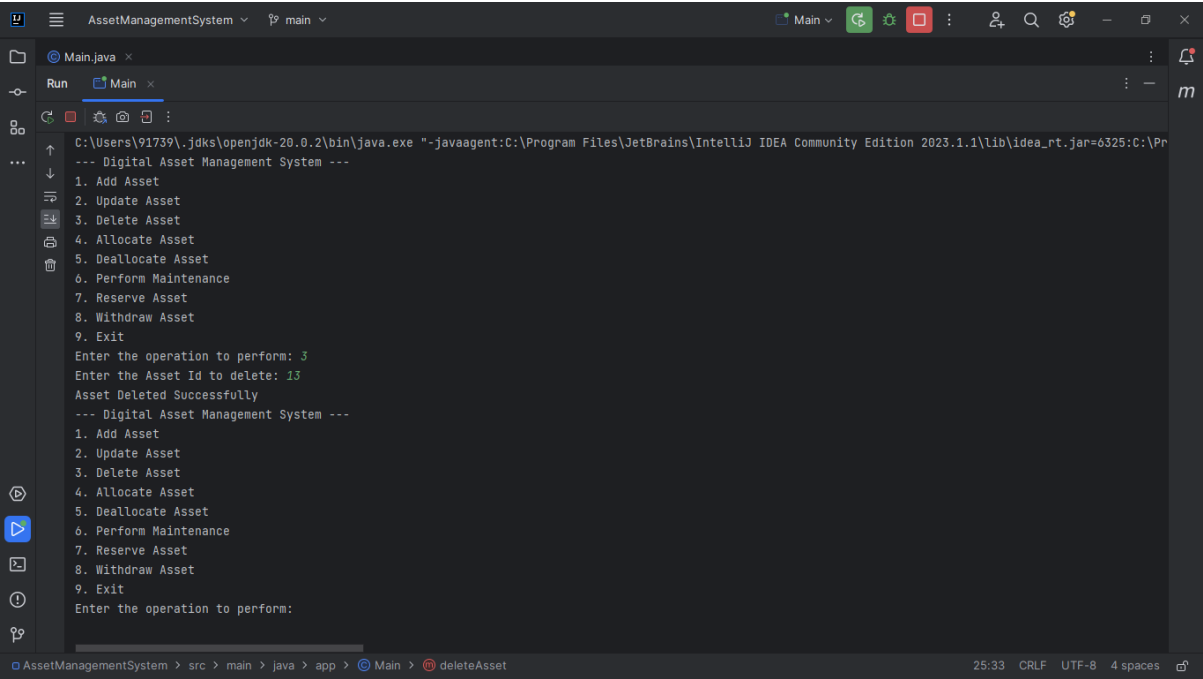


MySQL 8.0 Command Line Cli

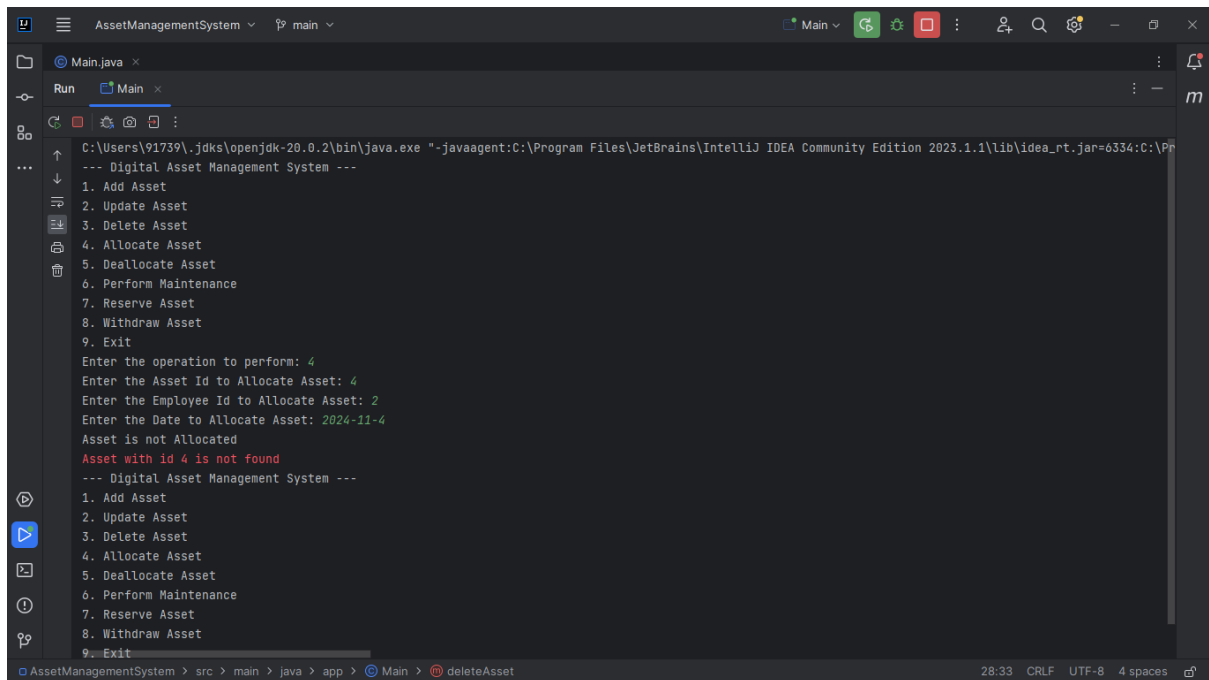
```
mysql> select * from assets;
```

asset_id	name	type	serial_number	purchase_date	location	status	owner_id
1	tv	electronics	12312	2024-09-01	chennai	new	1
2	iphone	mobile	43892	2027-01-01	USA	new	2
5	samsung	mobile	a12daa	2024-01-05	chennai	new	6
10	table	equipment	alsd2342	2024-12-01	pune	in use	6
13	Asus tuf	laptop	439sagae3	2024-08-01	chennai	in use	4

Similarly for all the cases

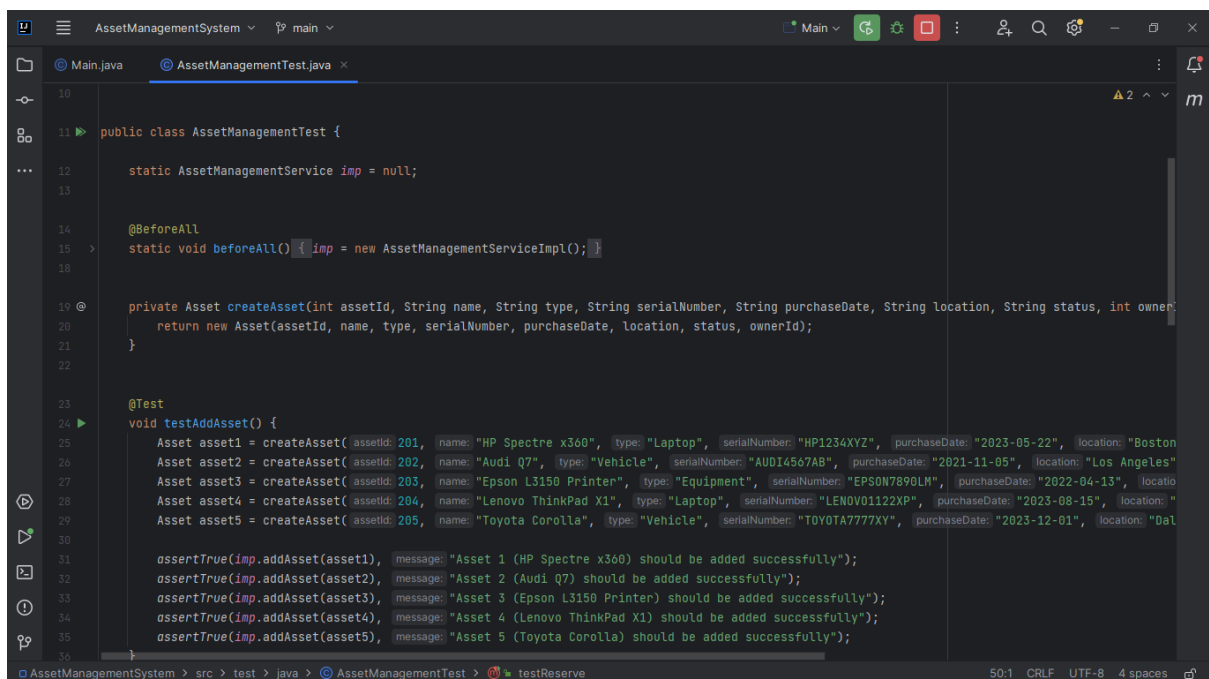


Exception is thrown If Asset Id or Employee Id is not Found



```
AssetManagementSystem > main > Main > deleteAsset
C:\Users\91739\jdk\openjdk-20.0.2\bin\java.exe --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.1\lib\idea_rt.jar=6334:C:\P
--- Digital Asset Management System ---
1. Add Asset
2. Update Asset
3. Delete Asset
4. Allocate Asset
5. Deallocate Asset
6. Perform Maintenance
7. Reserve Asset
8. Withdraw Asset
9. Exit
Enter the operation to perform: 4
Enter the Asset Id to Allocate Asset: 4
Enter the Employee Id to Allocate Asset: 2
Enter the Date to Allocate Asset: 2024-11-4
Asset is not Allocated
Asset with id 4 is not found
--- Digital Asset Management System ---
1. Add Asset
2. Update Asset
3. Delete Asset
4. Allocate Asset
5. Deallocate Asset
6. Perform Maintenance
7. Reserve Asset
8. Withdraw Asset
9. Exit
```

Implemented Unit testing with the help of Junit library, Used Maven to add the library.



```
AssetManagementSystem > src > test > java > AssetManagementTest > testReserve
10
11 public class AssetManagementTest {
12
13     static AssetManagementService imp = null;
14
15     @BeforeAll
16     static void beforeAll() { imp = new AssetManagementServiceImpl(); }
17
18
19     private Asset createAsset(int assetId, String name, String type, String serialNumber, String purchaseDate, String location, String status, int owner)
20     return new Asset(assetId, name, type, serialNumber, purchaseDate, location, status, ownerId);
21 }
22
23 @Test
24 void testAddAsset() {
25     Asset asset1 = createAsset(assetId: 201, name: "HP Spectre x360", type: "Laptop", serialNumber: "HP1234XYZ", purchaseDate: "2023-05-22", location: "Boston
26     Asset asset2 = createAsset(assetId: 202, name: "Audi Q7", type: "Vehicle", serialNumber: "AUDI4567AB", purchaseDate: "2021-11-05", location: "Los Angeles"
27     Asset asset3 = createAsset(assetId: 203, name: "Epson L3150 Printer", type: "Equipment", serialNumber: "EPSON7890LM", purchaseDate: "2022-04-13", locatio
28     Asset asset4 = createAsset(assetId: 204, name: "Lenovo ThinkPad X1", type: "Laptop", serialNumber: "LEN0V01122XP", purchaseDate: "2023-08-15", location: "
29     Asset asset5 = createAsset(assetId: 205, name: "Toyota Corolla", type: "Vehicle", serialNumber: "TOYOTA7777XY", purchaseDate: "2023-12-01", location: "Dal
30
31     assertTrue(imp.addAsset(asset1), message: "Asset 1 (HP Spectre x360) should be added successfully");
32     assertTrue(imp.addAsset(asset2), message: "Asset 2 (Audi Q7) should be added successfully");
33     assertTrue(imp.addAsset(asset3), message: "Asset 3 (Epson L3150 Printer) should be added successfully");
34     assertTrue(imp.addAsset(asset4), message: "Asset 4 (Lenovo ThinkPad X1) should be added successfully");
35     assertTrue(imp.addAsset(asset5), message: "Asset 5 (Toyota Corolla) should be added successfully");
36 }
```

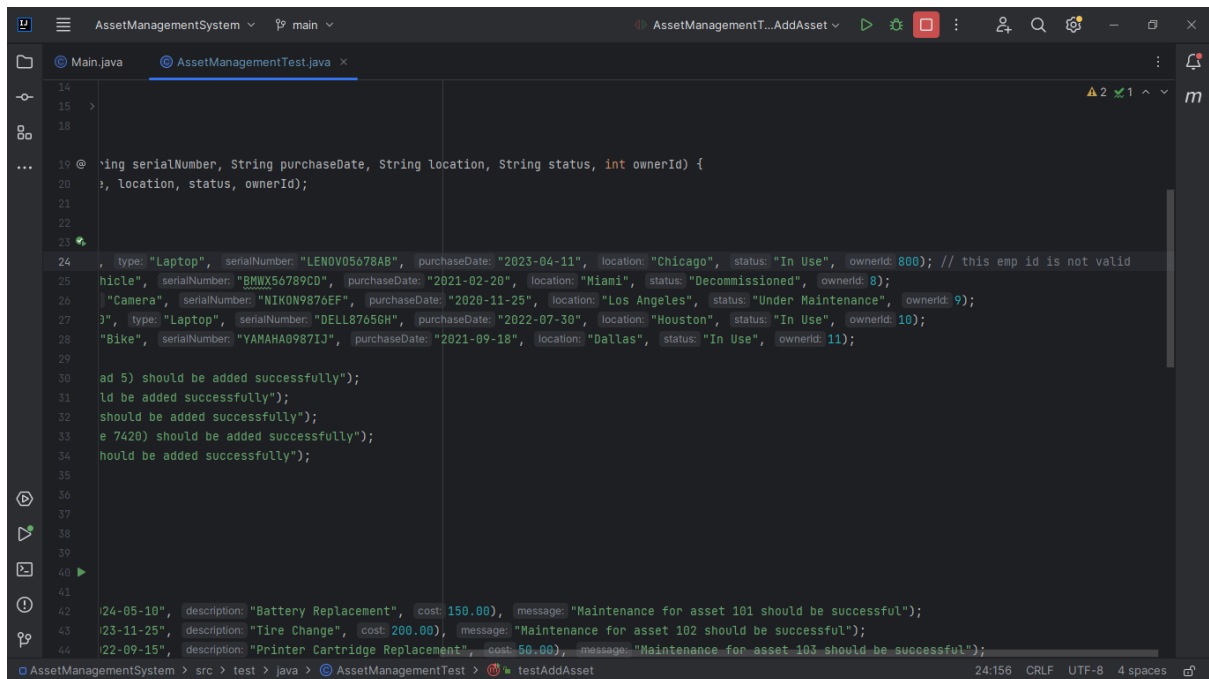
```
31     assertTrue(inp.addAsset(asset1), message: "Asset 1 (HP Spectre x360) should be added successfully");
32     assertTrue(inp.addAsset(asset2), message: "Asset 2 (Audi Q7) should be added successfully");
33     assertTrue(inp.addAsset(asset3), message: "Asset 3 (Epson L3150 Printer) should be added successfully");
34     assertTrue(inp.addAsset(asset4), message: "Asset 4 (Lenovo ThinkPad X1) should be added successfully");
35     assertTrue(inp.addAsset(asset5), message: "Asset 5 (Toyota Corolla) should be added successfully");
36 }
37
38 @Test
39 void performMaintenance() {
40
41     assertTrue(inp.performMaintenance(assetId: 101, maintenanceDate: "2024-05-10", description: "Battery Replacement", cost: 150.00), message: "Maintenance for asset 101 should be successful");
42     assertTrue(inp.performMaintenance(assetId: 102, maintenanceDate: "2023-11-25", description: "Tire Change", cost: 200.00), message: "Maintenance for asset 102 should be successful");
43     assertTrue(inp.performMaintenance(assetId: 103, maintenanceDate: "2022-09-15", description: "Printer Cartridge Replacement", cost: 50.00), message: "Maintenance for asset 103 should be successful");
44     assertTrue(inp.performMaintenance(assetId: 104, maintenanceDate: "2024-03-30", description: "Screen Repair", cost: 300.00), message: "Maintenance for asset 104 should be successful");
45 }
46
47 @Test
48 void testReserve() {
49
50     assertTrue(inp.reserveAsset(assetId: 1, employeeId: 1, reservationDate: "2024-11-25", startDate: "2024-11-27", endDate: "2024-11-29", status: "approved"), message: "Reservation for asset 1 should be successful");
51     assertTrue(inp.reserveAsset(assetId: 2, employeeId: 2, reservationDate: "2024-11-26", startDate: "2024-11-28", endDate: "2024-11-30", status: "canceled"), message: "Reservation for asset 2 should be successful");
52     assertTrue(inp.reserveAsset(assetId: 5, employeeId: 3, reservationDate: "2024-11-23", startDate: "2024-11-26", endDate: "2024-11-28", status: "approved"), message: "Reservation for asset 5 should be successful");
53     assertTrue(inp.reserveAsset(assetId: 1, employeeId: 4, reservationDate: "2024-11-25", startDate: "2024-11-27", endDate: "2024-11-29", status: "approved"), message: "Reservation for asset 1 should be successful");
54     assertTrue(inp.reserveAsset(assetId: 1, employeeId: 5, reservationDate: "2024-11-25", startDate: "2024-11-27", endDate: "2024-11-29", status: "pending"), message: "Reservation for asset 1 should be successful");
55 }
56
57 @Test
```

Test case check :

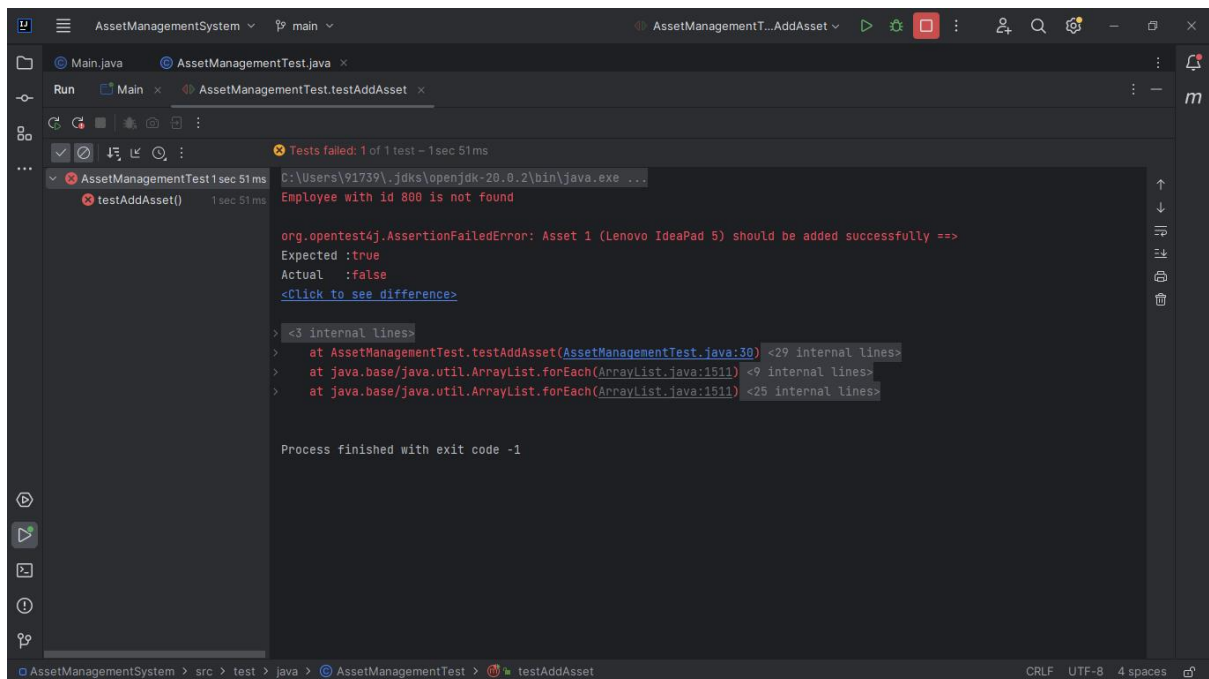
Passed

```
Run AssetManagementTest.testAddAsset
Tests passed: 1 of 1 test - 1sec 136 ms
C:\Users\91739\jdk\openjdk-20.0.2\bin\java.exe ...
Process finished with exit code 0
```


Let us see if it fails if the employee id is invalid and check



```
14  
15  
18  
...  
19 @Test  
20 void testAddAsset(String serialNumber, String purchaseDate, String location, String status, int ownerId) {  
21     // Add assets  
22     // Add assets  
23  
24     // Add Laptop  
25     // Add Laptop  
26     // Add Camera  
27     // Add Laptop  
28     // Add Bike  
29  
30     // Add Laptop  
31     // Add Laptop  
32     // Add Laptop  
33     // Add Laptop  
34     // Add Laptop  
35  
36  
37  
38  
39  
40  
41  
42     // Add Laptop  
43     // Add Laptop  
44     // Add Laptop  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```



```
Run  
Main  
AssetManagementTest.testAddAsset  
Tests failed: 1 of 1 test - 1 sec 51 ms  
AssetManagementTest 1 sec 51 ms  
testAddAsset() 1 sec 51 ms  
Employee with id 800 is not found  
org.opentest4j.AssertionFailedError: Asset 1 (Lenovo IdeaPad 5) should be added successfully ==>  
Expected :true  
Actual :false  
<Click to see difference>  
> <3 internal lines>  
> at AssetManagementTest.testAddAsset(AssetManagementTest.java:30) <29 internal lines>  
> at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>  
> at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <25 internal lines>  
Process finished with exit code -1
```