King Saud University
College of Computer and Information Sciences
Department of Information Systems

IS230: Introduction to Database Systems
1st Semester 1445 H

UniSchedular

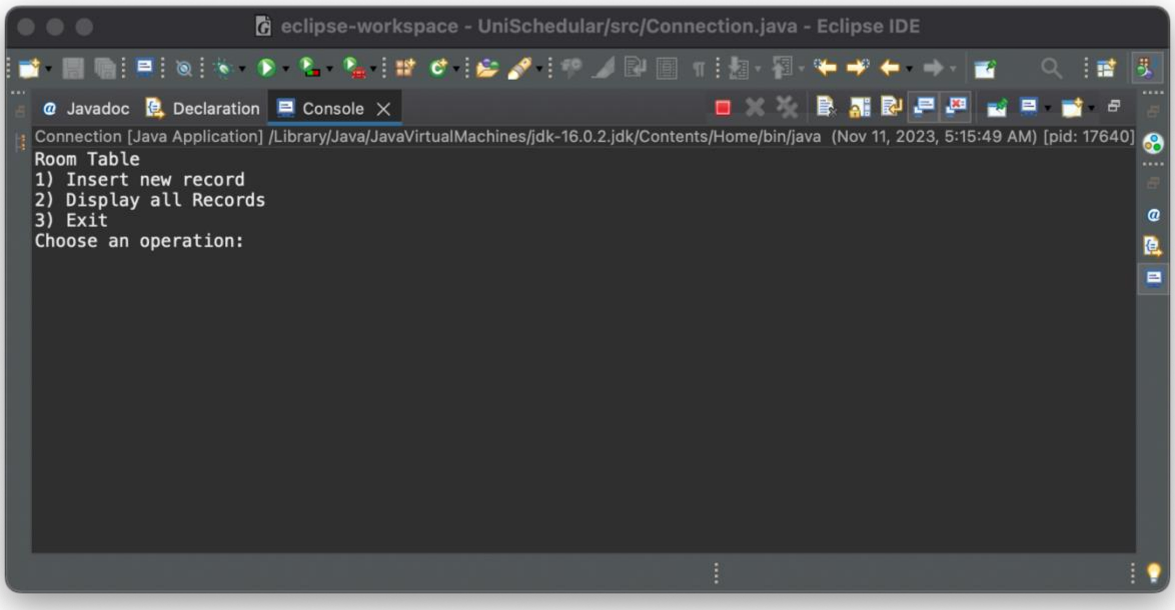
Phase # 2

Section #	NAME	ID
Group Number: 1		
60395	*Rana Ababtain*	*442200513*
60395	*Rahaf Alhammad*	*442200390*
60395	*Nouf Alkhashan*	*442201351*
60395	*Nouv B. Al-Qahtani*	*442201905*
60395	*Najah Al-Rowais*	*442201401*
60395	*Weam Alahmadi*	*442200412*

Supervised By: L. *Maram Alsuhaibani*
First Semester 1445

Part1: Screenshot of the execution showing how clear and specific messages will be displayed for each.

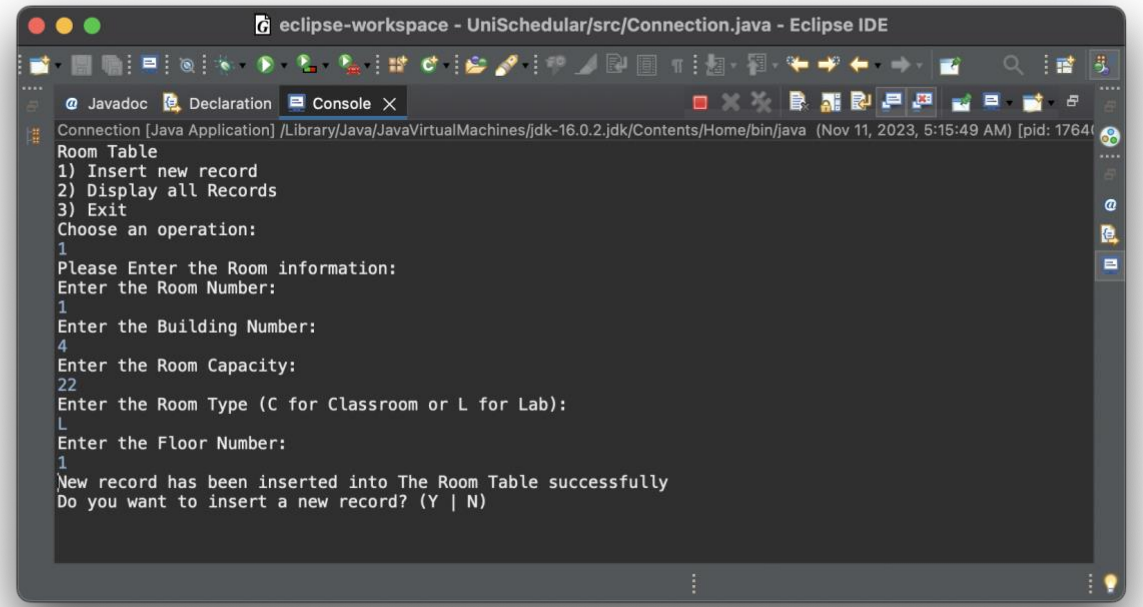
1)First Screen:


eclipse-workspace - UniSchedular/src/Connection.java - Eclipse IDE
Javadoc
Declaration
Console

```
Connection [Java Application] /Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java  (Nov 11, 2023, 5:15:49 AM) [pid: 17640]
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
```

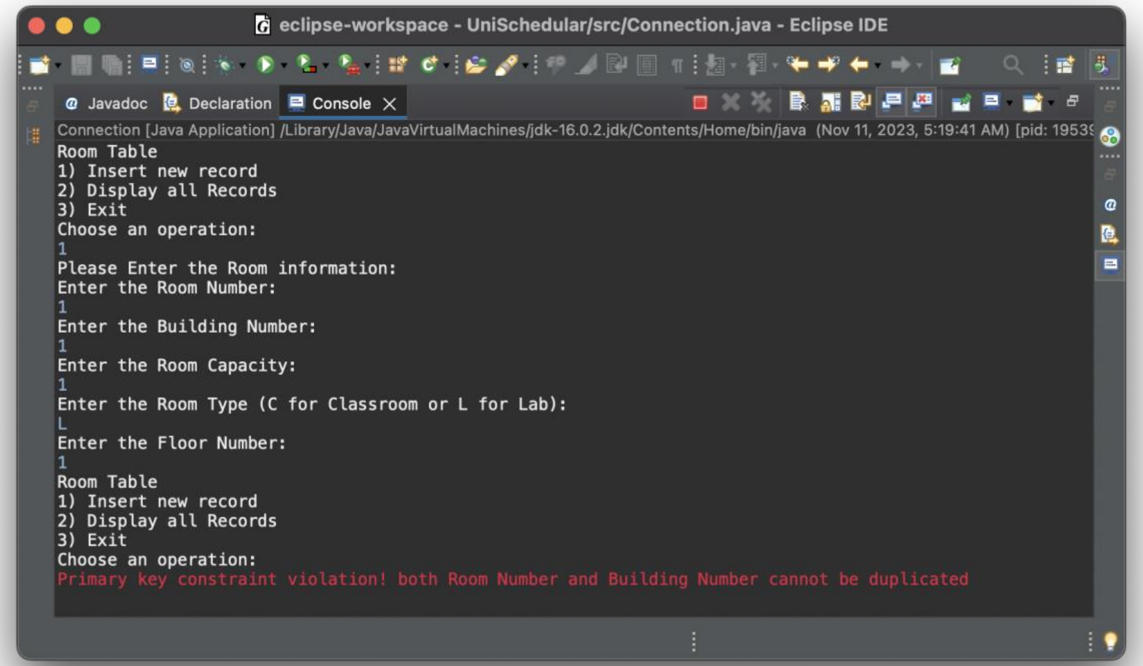
2) INSERT Operation (EXECUTION of multiple insertion + dealing with Exception):

Successful operation:


eclipse-workspace - UniSchedular/src/Connection.java - Eclipse IDE
Javadoc
Declaration
Console

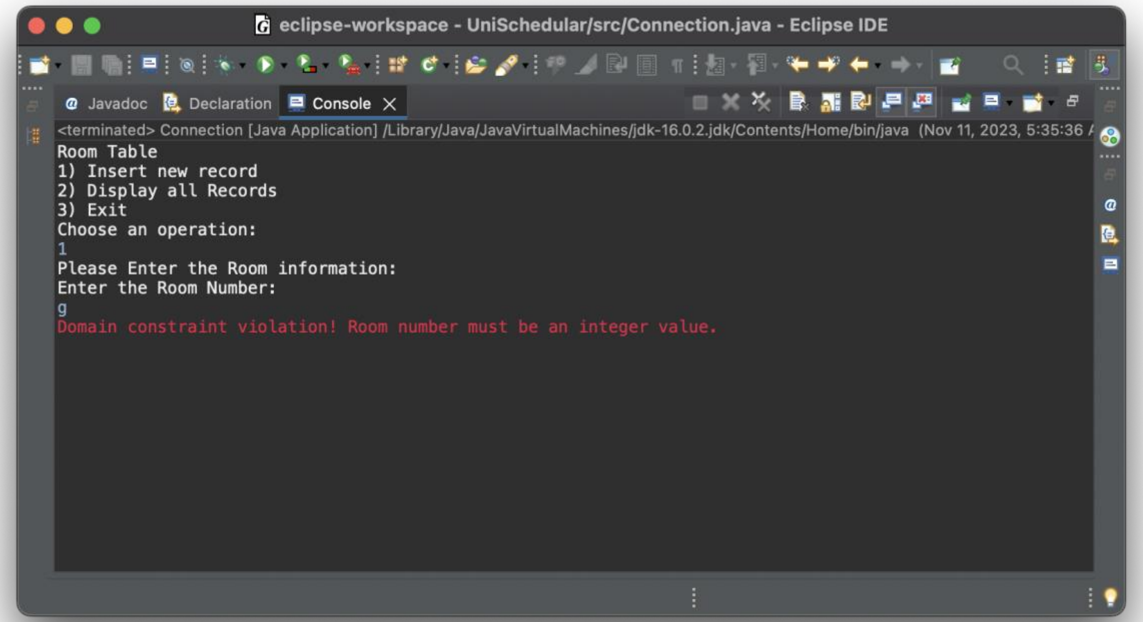
```
Connection [Java Application] /Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java  (Nov 11, 2023, 5:15:49 AM) [pid: 1764
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
1
Please Enter the Room information:
Enter the Room Number:
1
Enter the Building Number:
4
Enter the Room Capacity:
22
Enter the Room Type (C for Classroom or L for Lab):
L
Enter the Floor Number:
1
New record has been inserted into The Room Table successfully
Do you want to insert a new record? (Y | N)
```

Unsuccessful operation, duplicate primary key:


eclipse-workspace - UniSchedular/src/Connection.java - Eclipse IDE
Javadoc
Declaration
Console

```
Connection [Java Application] /Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java  (Nov 11, 2023, 5:19:41 AM) [pid: 1953
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
1
Please Enter the Room information:
Enter the Room Number:
1
Enter the Building Number:
1
Enter the Room Capacity:
1
Enter the Room Type (C for Classroom or L for Lab):
L
Enter the Floor Number:
1
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
Primary key constraint violation! both Room Number and Building Number cannot be duplicated
```

Unsuccessful operation, Domain constraint violation:

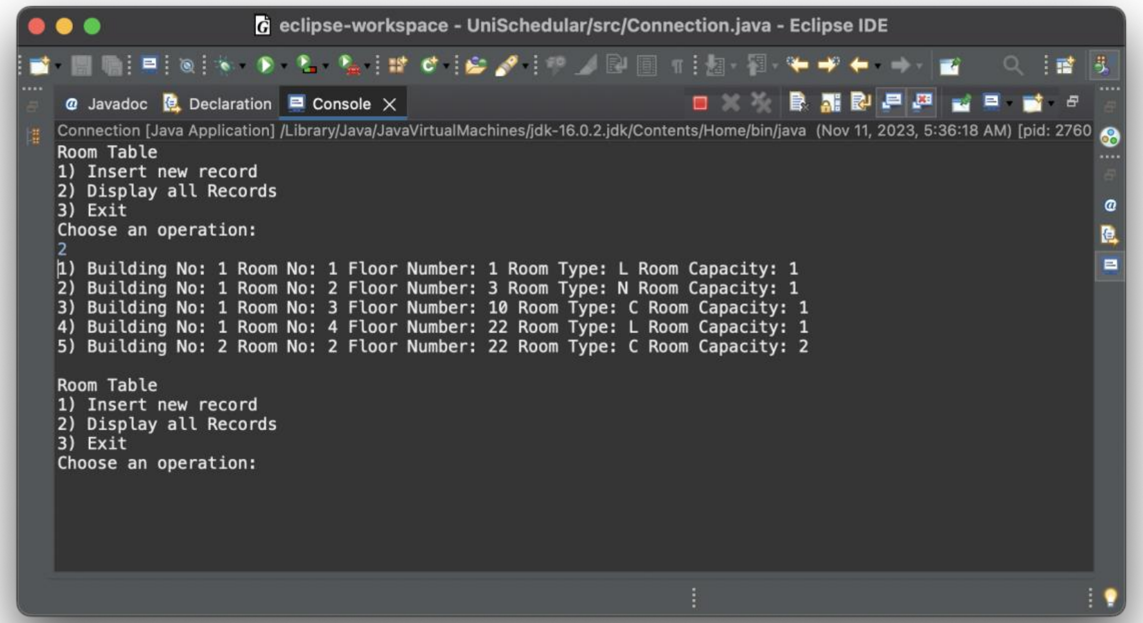

eclipse-workspace - UniSchedular/src/Connection.java - Eclipse IDE
Javadoc
Declaration
Console

```
<terminated> Connection [Java Application] /Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java  (Nov 11, 2023, 5:35:36
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
1
Please Enter the Room information:
Enter the Room Number:
g
Domain constraint violation! Room number must be an integer value.
```

Note:

The integrity constraint violation (entering null for Room number and Building number) couldn't be entered because we set the primary key in the sql queries as NOT NULL earlier (in phase 1).

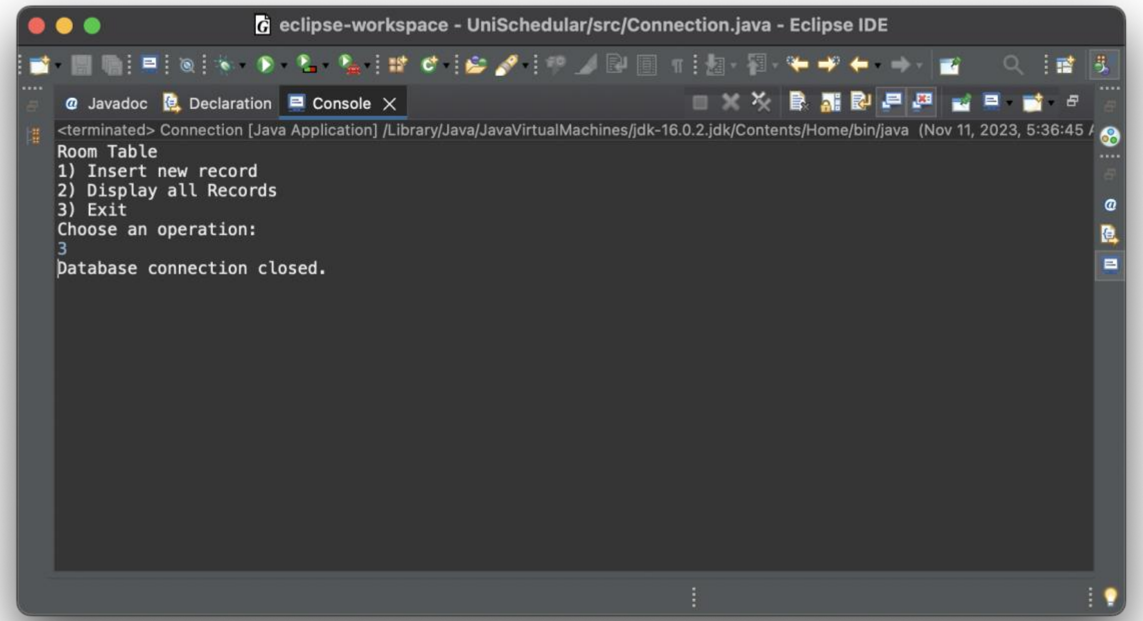
3) Display Operation:


eclipse-workspace - UniSchedular/src/Connection.java - Eclipse IDE
Javadoc
Declaration
Console

```
Connection [Java Application] /Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java  (Nov 11, 2023, 5:36:18 AM) [pid: 2760
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
2
1) Building No: 1 Room No: 1 Floor Number: 1 Room Type: L Room Capacity: 1
2) Building No: 1 Room No: 2 Floor Number: 3 Room Type: N Room Capacity: 1
3) Building No: 1 Room No: 3 Floor Number: 10 Room Type: C Room Capacity: 1
4) Building No: 1 Room No: 4 Floor Number: 22 Room Type: L Room Capacity: 1
5) Building No: 2 Room No: 2 Floor Number: 22 Room Type: C Room Capacity: 2

Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
```

4) Exit operation:


eclipse-workspace - UniSchedular/src/Connection.java - Eclipse IDE
Javadoc
Declaration
Console

```
<terminated> Connection [Java Application] /Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java (Nov 11, 2023, 5:36:45
Room Table
1) Insert new record
2) Display all Records
3) Exit
Choose an operation:
3
Database connection closed.
```

Part 2: Source code

1)INSERTION Code

```
case 1: // Insert new record
            try {
              boolean isTrue = true;
              while (isTrue) {

                System.out.println("Please Enter the Room information: ");

                     // 1
                System.out.println("Enter the Room Number: ");
                currentAttribute = "RoomNo";
                int roomNo = input.nextInt();

                if (roomNo == 0) {
                  System.err.println("Invalid Input, the  Room Number should be positive");
```

```
                    break; }

                        // 2
                  System.out.println("Enter the Building Number: ");
                  currentAttribute = "BuildingNo";
                  int buildingNo = input.nextInt();

                  if( buildingNo == 0 ) {
                    System.err.println("Invalid Input, the Building Number should be positive");
                    break; }

                        // 3
                  System.out.println("Enter the Room Capacity: ");
                  currentAttribute = "Capacity";
                  int capacity = input.nextInt();

                        // 4
                  System.out.println("Enter the Room Type (C for Classroom or L for Lab): ");
                  currentAttribute = "RoomType";
                  String roomType = input.next();

                  if (!roomType.equalsIgnoreCase("C") && !roomType.equalsIgnoreCase("L")) {
                      throw new IllegalArgumentException("Invalid room type! Please enter 'C' for Classroom or 'L'
for Lab.");
                  }

                        // 5
                  System.out.println("Enter the Floor Number: ");
                  currentAttribute = "FloorNo";
                  int floorNo = input.nextInt();




                        // Inserting the data to the database
                  stmt.executeUpdate("INSERT INTO Room VALUES(" + roomNo + "," + buildingNo + ","
                              + capacity + ",'" + roomType + "'," + floorNo + ")");
```

```
            System.out.println("New record has been inserted into The Room Table successfully");

                // Asking the user if they want to add more records
            System.out.println("Do you want to insert a new record? (Y | N)");
            char newRecord = input.next().charAt(0);
            newRecord = Character.toUpperCase(newRecord);

            boolean wrongChar = true;

                // Verify the input
            if (!(newRecord == 'N' || newRecord == 'Y')) {
              while (wrongChar) {
                System.out.println("Invalid character! Please use only (Y | N).");
                newRecord = input.next().charAt(0);
                newRecord = Character.toUpperCase(newRecord);
                if (newRecord == 'N' || newRecord == 'Y')
                  wrongChar = false;
              }
            }


            if (newRecord == 'N')
              isTrue = false;
          }
        } catch (SQLException sqle) {
          if (sqle.getErrorCode() == 1062)
            System.err.println("Primary key constraint violation! both Room Number and Building Number
cannot be duplicated");
          else if (sqle.getErrorCode() == 1452)
            System.err.println("Foreign key constraint violation! The entered building number does not
exist");
          else if (sqle.getErrorCode() == 1048)
            System.err.println("Not NULL constraint violation! All values cannot be NULL");
          else if (sqle.getErrorCode() == 1265 || sqle.getErrorCode() == 1366)
            System.err.println("Domain constraint violation!");
          else {
            System.err.println(sqle.getErrorCode());
            System.err.println(sqle.getMessage());
        }
    }

    break;
```

2)Display Code

```
case 2: // Display all records

        java.sql.Connection connection = DriverManager.getConnection(jdbcUrl, username, password);

        Statement statement = connection.createStatement();

        String sql = "SELECT * FROM room";

        ResultSet resultSet = statement.executeQuery(sql);

        int rowNumber = 1;

        while (resultSet.next()) {
          int buildingNo = resultSet.getInt("BuildingNO");
          int roomNo = resultSet.getInt("RoomNo");
          int floorNumber = resultSet.getInt("FloorNo");
          String roomType = resultSet.getString("Roomtype");
          int roomCapacity = resultSet.getInt("Capacity");
          System.out.print(rowNumber + ") ");
          System.out.print("Building No: " + buildingNo + " ");
          System.out.print("Room No: " + roomNo + " ");
          System.out.print("Floor Number: " + floorNumber + " ");
          System.out.print("Room Type: " + roomType + " ");
          System.out.print("Room Capacity: " + roomCapacity + " ");

          System.out.println();
```

```
               rowNumber =  rowNumber + 1;


             }

             resultSet.close();
             statement.close();
             connection.close();
             System.out.println();
             break;
```

3)Exit Code

```
case 3: // Exit
             alwaysTrue = false;
             break;

           default:
             System.out.println("Invalid choice! Please select a valid option.");
             break;
         }

       } while (alwaysTrue);

     try {
                     // Closing the database connection
       if (rs != null)
```

```
            rs.close();
          if (stmt != null)
            stmt.close();
          if (con != null)
            con.close();
        } catch (SQLException e) {
          e.printStackTrace();
        } finally {
                        // Close the database connection
          if (con != null) {
            try {
              con.close();
              System.out.println("Database connection closed.");
            } catch (SQLException e) {
              e.printStackTrace();
            }
          }
          input.close(); }
```

4) Code dealing with Exceptions

The catch within the INSERT case:

```
catch (SQLException sqle) {
            if (sqle.getErrorCode() == 1062)
              System.err.println("Primary key constraint violation! both Room Number and Building Number cannot be duplicated");
            else if (sqle.getErrorCode() == 1452)
              System.err.println("Foreign key constraint violation! The entered building number does not exist");
            else if (sqle.getErrorCode() == 1048)
              System.err.println("Not NULL constraint violation! All values cannot be NULL");
            else if (sqle.getErrorCode() == 1265 || sqle.getErrorCode() == 1366)
              System.err.println("Domain constraint violation!");
            else {
              System.err.println(sqle.getErrorCode());
              System.err.println(sqle.getMessage());
            }
          }
```

The catch within EXIT case:

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

All handlers for all possible exceptions

```
   catch (SQLException sqle) {
       if (sqle.getErrorCode() == 1062) {
          System.err.println("Primary key constraint violation! A record with the same BuildingNo and RoomNo already
exists.");
       } else if (sqle.getErrorCode() == 1452) {
          System.err.println("Foreign key constraint violation! Either TechnicianEmpID or MediaProvidedCode references a
non-existing record.");
       } else if (sqle.getErrorCode() == 1048) {
          System.err.println("Not NULL constraint violation! One or more required attributes are missing.");
       } else if (sqle.getErrorCode() == 1265 || sqle.getErrorCode() == 1366) {
          System.err.println("Domain constraint violation! Invalid data type or value provided for an attribute.");
       } else {
          System.err.println(sqle.getMessage());
       }
     }
     catch (InputMismatchException ex) {
       switch (currentAttribute) {
          case "BuildingNo":
            System.err.println("Domain constraint violation! Building number must be an integer value.");
            break;
          case "RoomNo":
            System.err.println("Domain constraint violation! Room number must be an integer value.");
            break;
          case "FloorNo":
            System.err.println("Domain constraint violation! Floor number must be an integer value.");
            break;
          case "RoomType":
            System.err.println("Domain constraint violation! Room type must be a string.");
            break;
          case "Capacity":
            System.err.println("Domain constraint violation! Capacity must be an integer value.");
            break;
          default:
            System.err.println("Invalid input for attribute: " + currentAttribute);
            break;
        }
        input.next();
    } catch (IllegalArgumentException ex) {
         System.err.println(ex.getMessage());
     } catch (Exception e) {
        System.err.print(e.getMessage());
    }
```

The Whole Source code:

```
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.InputMismatchException;
import java.util.Scanner;
import java.sql.*;

public class Connection {
  static java.sql.Connection con = null;
  static Statement stmt = null;
  static PreparedStatement ps = null;
  static ResultSet rs = null;
  static Scanner input = new Scanner(System.in);
  static String currentAttribute = "";

  public static void main(String[] args) {

    try {
      String jdbcUrl = "jdbc:mysql://localhost:3306/UniSchedular";
      String username = "root";
      String password = "";

        // Establish the database connection
```

```
con = DriverManager.getConnection(jdbcUrl, username, password);
stmt = con.createStatement();

boolean alwaysTrue = true;


do {
  System.out.println("Room Table");
  System.out.println("1) Insert new record");
  System.out.println("2) Display all Records");
  System.out.println("3) Exit");
  System.out.println("Choose an operation:");
  int queryChoice = input.nextInt();

  switch (queryChoice) {
    case 1: // Insert new record
      try {
        boolean isTrue = true;
        while (isTrue) {
               // Insert new record
          System.out.println("Please Enter the Room information: ");

               // 1
          System.out.println("Enter the Room Number: ");
          currentAttribute = "RoomNo";
          int roomNo = input.nextInt();

          if (roomNo == 0 ) {
             System.err.println("Invalid Input, the  Room Number should be positive");
             break; }

               // 2
          System.out.println("Enter the Building Number: ");
          currentAttribute = "BuildingNo";
          int buildingNo = input.nextInt();

          if( buildingNo == 0 ) {
             System.err.println("Invalid Input, the Building Number should be positive");
             break; }
               // 3
          System.out.println("Enter the Room Capacity: ");
          currentAttribute = "Capacity";
          int capacity = input.nextInt();


               // 4
          System.out.println("Enter the Room Type (C for Classroom or L for Lab): ");
          currentAttribute = "RoomType";
          String roomType = input.next();

          if (!roomType.equalsIgnoreCase("C") && !roomType.equalsIgnoreCase("L")) {
              throw new IllegalArgumentException("Invalid room type! Please enter 'C' for Classroom or 'L' for Lab.");
          }


               // 5
          System.out.println("Enter the Floor Number: ");
          currentAttribute = "FloorNo";
          int floorNo = input.nextInt();




               // Inserting the data to the database
          stmt.executeUpdate("INSERT INTO Room VALUES(" + roomNo + "," + buildingNo + ","
                          + capacity + ",'" + roomType + "'," + floorNo + ")");

          System.out.println("New record has been inserted into The Room Table successfully");

               // Asking the user if they want to add more records
          System.out.println("Do you want to insert a new record? (Y | N)");
          char newRecord = input.next().charAt(0);
          newRecord = Character.toUpperCase(newRecord);

          boolean wrongChar = true;

               // Verify the input
          if (!(newRecord == 'N' || newRecord == 'Y')) {
             while (wrongChar) {
                System.out.println("Invalid character! Please use only (Y | N).");
                newRecord = input.next().charAt(0);
                newRecord = Character.toUpperCase(newRecord);
```

```
                    if (newRecord == 'N' || newRecord == 'Y')
                      wrongChar = false;
                    }
                  }

                if (newRecord == 'N')
                  isTrue = false;
                }
            } catch (SQLException sqle) {
              if (sqle.getErrorCode() == 1062)
                System.err.println("Primary key constraint violation! both Room Number and Building Number cannot be
duplicated");
              else if (sqle.getErrorCode() == 1452)
                System.err.println("Foreign key constraint violation! The entered building number does not exist");
              else if (sqle.getErrorCode() == 1048)
                System.err.println("Not NULL constraint violation! All values cannot be NULL");
              else if (sqle.getErrorCode() == 1265 || sqle.getErrorCode() == 1366)
                System.err.println("Domain constraint violation!");
              else {
                System.err.println(sqle.getErrorCode());
                System.err.println(sqle.getMessage());
              }
            }

            break;

          case 2: // Display all records

            java.sql.Connection connection = DriverManager.getConnection(jdbcUrl, username, password);

            Statement statement = connection.createStatement();

            String sql = "SELECT * FROM room";

            ResultSet resultSet = statement.executeQuery(sql);

            int rowNumber = 1;

            while (resultSet.next()) {
              int buildingNo = resultSet.getInt("BuildingNO");
            int roomNo = resultSet.getInt("RoomNo");
            int floorNumber = resultSet.getInt("FloorNo");
            String roomType = resultSet.getString("Roomtype");
            int roomCapacity = resultSet.getInt("Capacity");
            System.out.print(  rowNumber + ") ");
            System.out.print("Building No: " + buildingNo + " ");
            System.out.print("Room No: " + roomNo + " ");
            System.out.print("Floor Number: " + floorNumber + " ");
            System.out.print("Room Type: " + roomType + " ");
            System.out.print("Room Capacity: " + roomCapacity + " ");

            System.out.println();

            rowNumber =  rowNumber + 1;


          }

          resultSet.close();
          statement.close();
          connection.close();
          System.out.println();
          break;

        case 3: // Exit
          alwaysTrue = false;
          break;

        default:
          System.out.println("Invalid choice! Please select a valid option.");
          break;
      }

    } while (alwaysTrue);

    try {
                    // Closing the database connection
      if (rs != null)
        rs.close();
      if (stmt != null)
```

```
          stmt.close();
        if (con != null)
          con.close();
      } catch (SQLException e) {
        e.printStackTrace();
      } finally {
                     // Close the database connection
        if (con != null) {
          try {
            con.close();
            System.out.println("Database connection closed.");
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        input.close(); }
      }

    catch (SQLException sqle) {
      if (sqle.getErrorCode() == 1062) {
        System.err.println("Primary key constraint violation! A record with the same BuildingNo and RoomNo already exists.");
      } else if (sqle.getErrorCode() == 1452) {
        System.err.println("Foreign key constraint violation! Either TechnicianEmpID or MediaProvidedCode references a non-
existing record.");
      } else if (sqle.getErrorCode() == 1048) {
        System.err.println("Not NULL constraint violation! One or more required attributes are missing.");
      } else if (sqle.getErrorCode() == 1265 || sqle.getErrorCode() == 1366) {
        System.err.println("Domain constraint violation! Invalid data type or value provided for an attribute.");
      } else {
        System.err.println(sqle.getMessage());
      }
    }
    catch (InputMismatchException ex) {
      switch (currentAttribute) {
        case "BuildingNo":
          System.err.println("Domain constraint violation! Building number must be an integer value.");
          break;
        case "RoomNo":
          System.err.println("Domain constraint violation! Room number must be an integer value.");
          break;
        case "FloorNo":
          System.err.println("Domain constraint violation! Floor number must be an integer value.");
          break;
        case "RoomType":
          System.err.println("Domain constraint violation! Room type must be a string.");
          break;
        case "Capacity":
          System.err.println("Domain constraint violation! Capacity must be an integer value.");
          break;
        default:
          System.err.println("Invalid input for attribute: " + currentAttribute);
          break;
      }
      input.next();
    } catch (IllegalArgumentException ex) {
       System.err.println(ex.getMessage());
     } catch (Exception e) {
      System.err.print(e.getMessage());
    }

  }}
```