CS 166 Phase 3 Report

Spyridon Catechis
Vivek Amatya

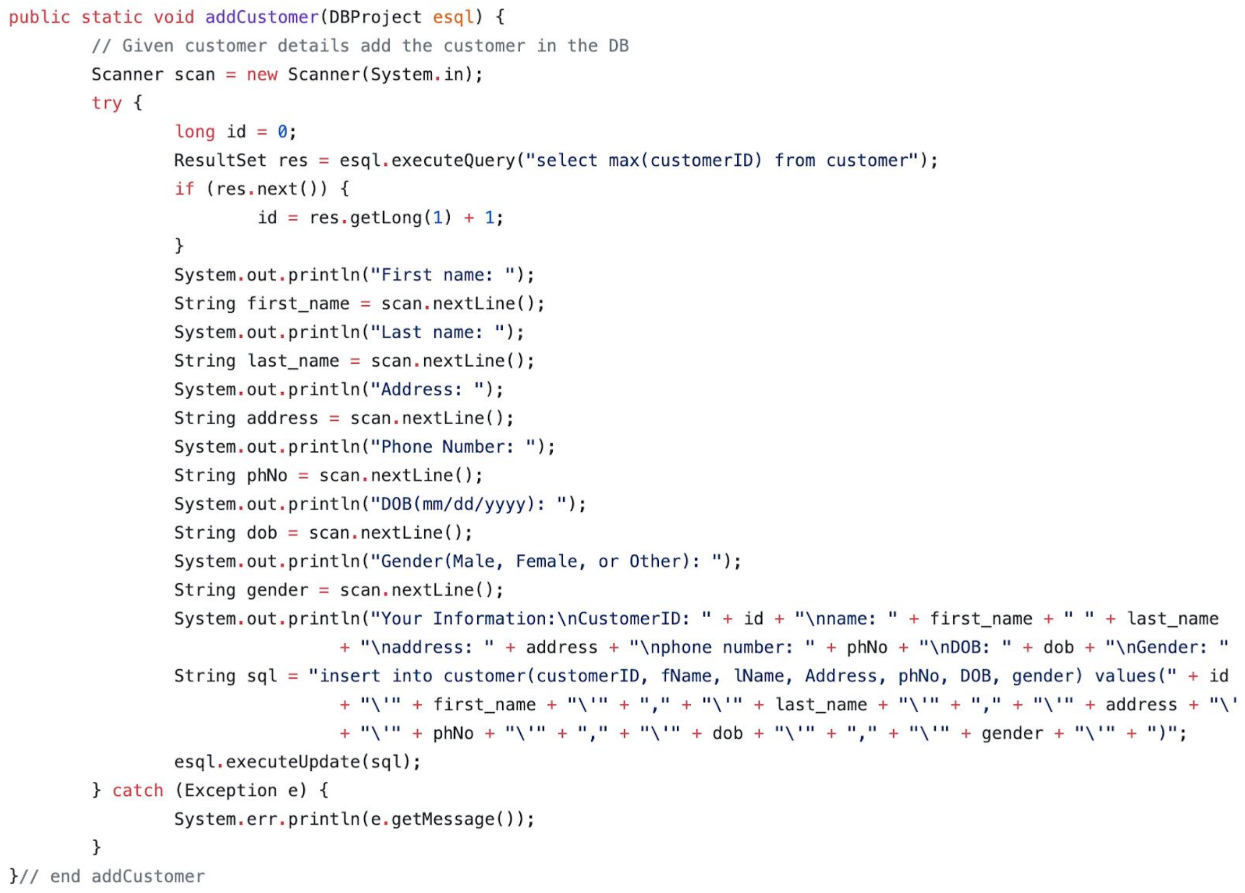
Assumptions:

1. Database is prepopulated (especially with ID values).
2. Access is restricted for certain functions.
3. Date formatting is checked by database.
4. Text, Numerical, and Char variable types are checked by database.
5. Multiple people do not write to the database at the same time.

Functions:

1: addCustomer(esql)

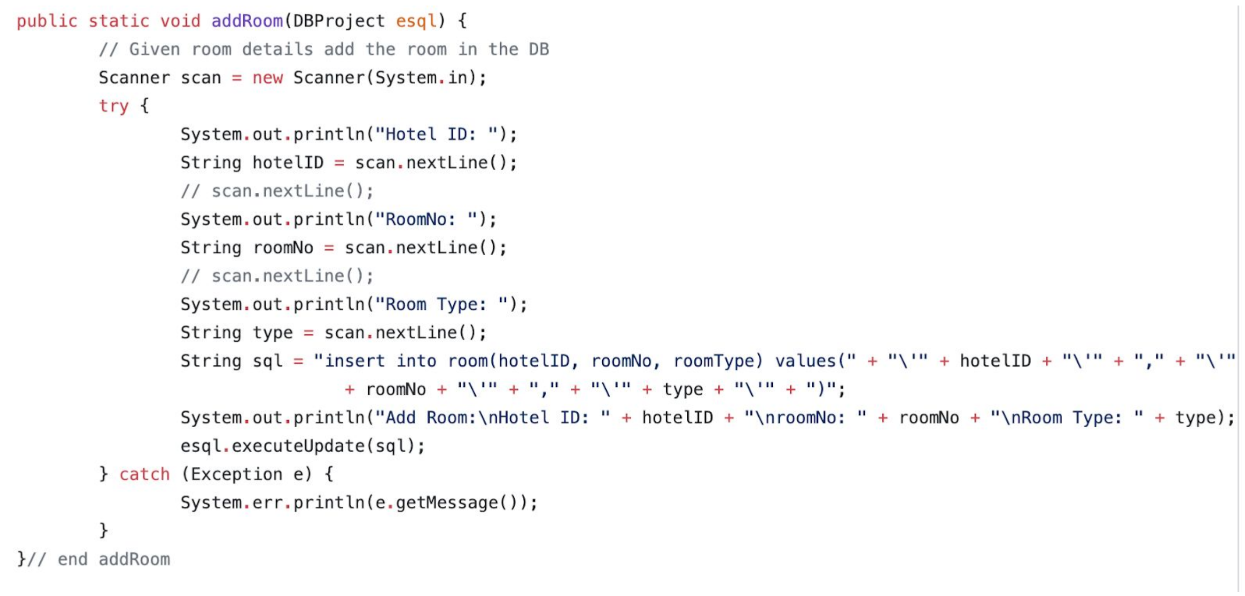
- Worked on by: Vivek



```java
public static void addCustomer(DBProject esql) {
        // Given customer details add the customer in the DB
        Scanner scan = new Scanner(System.in);
        try {
                long id = 0;
                ResultSet res = esql.executeQuery("select max(customerID) from customer");
                if (res.next()) {
                        id = res.getLong(1) + 1;
                }
                System.out.println("First name: ");
                String first_name = scan.nextLine();
                System.out.println("Last name: ");
                String last_name = scan.nextLine();
                System.out.println("Address: ");
                String address = scan.nextLine();
                System.out.println("Phone Number: ");
                String phNo = scan.nextLine();
                System.out.println("DOB(mm/dd/yyyy): ");
                String dob = scan.nextLine();
                System.out.println("Gender(Male, Female, or Other): ");
                String gender = scan.nextLine();
                System.out.println("Your Information:\nCustomerID: " + id + "\nname: " + first_name + " " + last_name
                                + "\naddress: " + address + "\nphone number: " + phNo + "\nDOB: " + dob + "\nGender: "
                String sql = "insert into customer(customerID, fName, lName, Address, phNo, DOB, gender) values(" + id
                                + "\'" + first_name + "\'" + "," + "\'" + last_name + "\'" + "," + "\'" + address + "\'
                                + "\'" + phNo + "\'" + "," + "\'" + dob + "\'" + "," + "\'" + gender + "\'" + ")";
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// end addCustomer
```

2: addRoom(esql)

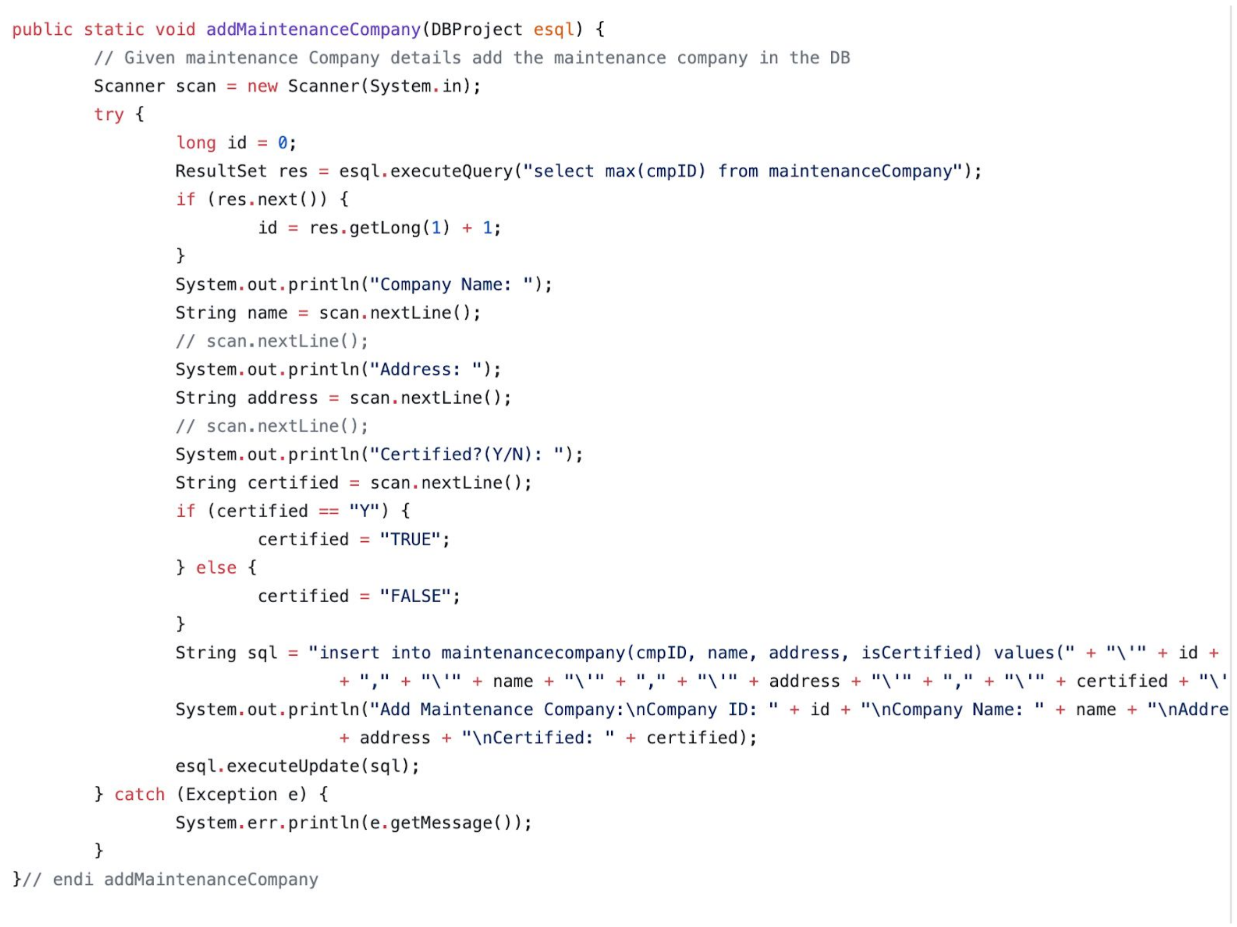
- Worked on by: Vivek



```
public static void addRoom(DBProject esql) {
        // Given room details add the room in the DB
        Scanner scan = new Scanner(System.in);
        try {
                System.out.println("Hotel ID: ");
                String hotelID = scan.nextLine();
                // scan.nextLine();
                System.out.println("RoomNo: ");
                String roomNo = scan.nextLine();
                // scan.nextLine();
                System.out.println("Room Type: ");
                String type = scan.nextLine();
                String sql = "insert into room(hotelID, roomNo, roomType) values(" + "\'" + hotelID + "\'" + "," + "\'"
                                + roomNo + "\'" + "," + "\'" + type + "\'" + ")";
                System.out.println("Add Room:\nHotel ID: " + hotelID + "\nroomNo: " + roomNo + "\nRoom Type: " + type);
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// end addRoom
```

3: addMaintenanceCompany(esql)

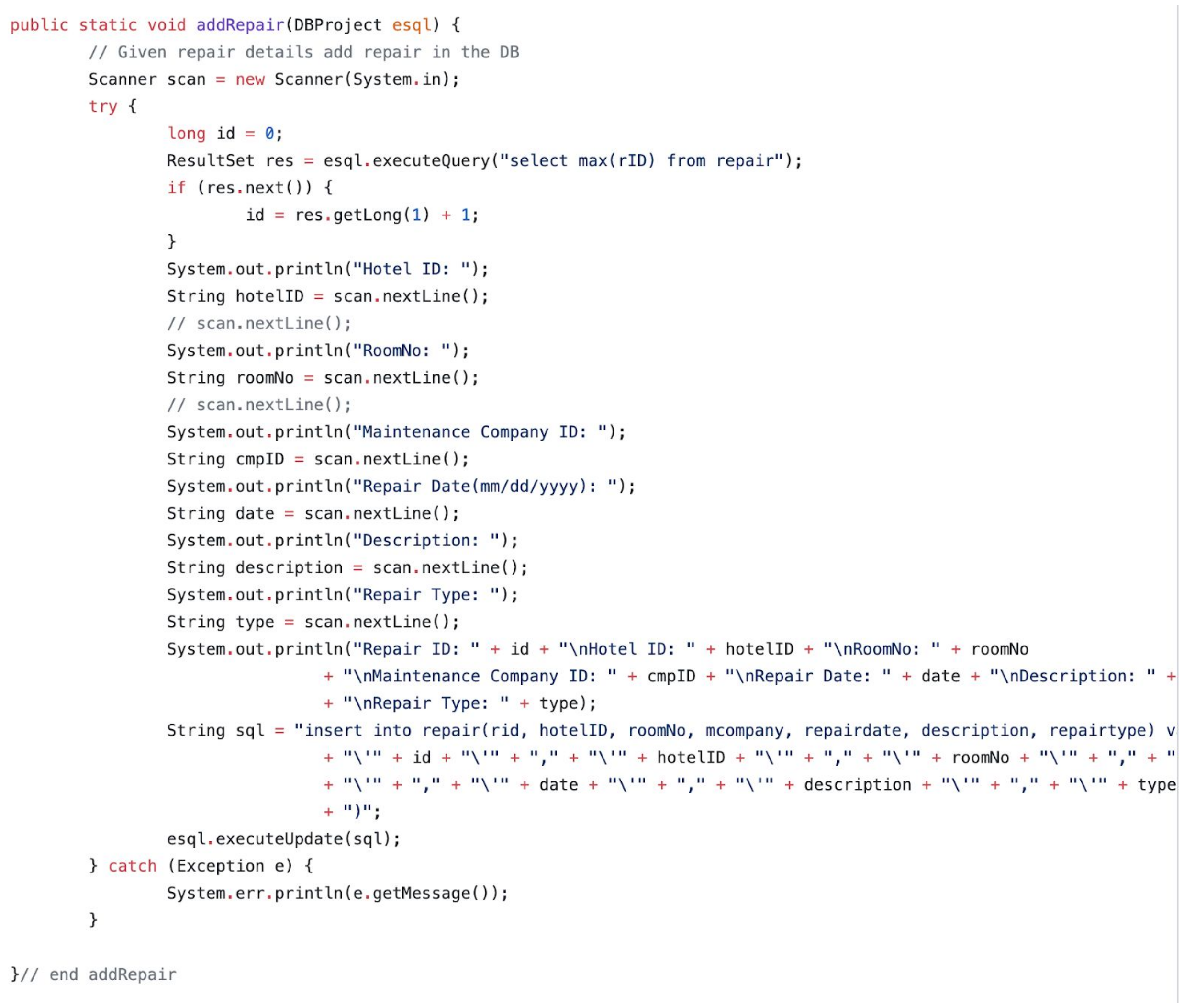
- Worked on by: Vivek



```
public static void addMaintenanceCompany(DBProject esql) {
        // Given maintenance Company details add the maintenance company in the DB
        Scanner scan = new Scanner(System.in);
        try {
                long id = 0;
                ResultSet res = esql.executeQuery("select max(cmpID) from maintenanceCompany");
                if (res.next()) {
                        id = res.getLong(1) + 1;
                }
                System.out.println("Company Name: ");
                String name = scan.nextLine();
                // scan.nextLine();
                System.out.println("Address: ");
                String address = scan.nextLine();
                // scan.nextLine();
                System.out.println("Certified?(Y/N): ");
                String certified = scan.nextLine();
                if (certified == "Y") {
                        certified = "TRUE";
                } else {
                        certified = "FALSE";
                }
                String sql = "insert into maintenancecompany(cmpID, name, address, isCertified) values(" + "\'" + id +
                                + "," + "\'" + name + "\'" + "," + "\'" + address + "\'" + "," + "\'" + certified + "\'
                System.out.println("Add Maintenance Company:\nCompany ID: " + id + "\nCompany Name: " + name + "\nAddre
                                + address + "\nCertified: " + certified);
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// endi addMaintenanceCompany
```

4: addRepair(esql)

- Worked on by: Vivek

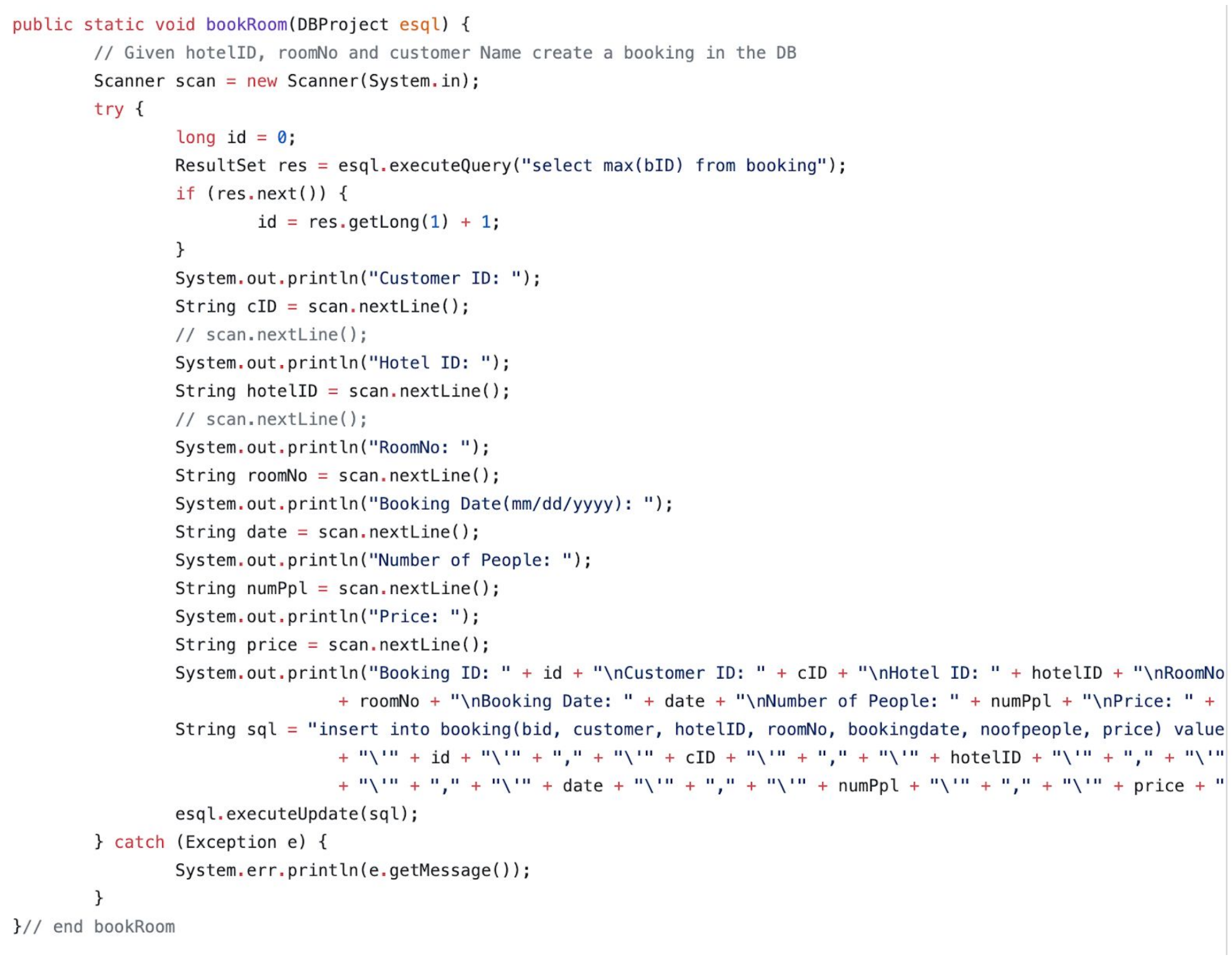


```
public static void addRepair(DBProject esql) {
        // Given repair details add repair in the DB
        Scanner scan = new Scanner(System.in);
        try {
                long id = 0;
                ResultSet res = esql.executeQuery("select max(rID) from repair");
                if (res.next()) {
                        id = res.getLong(1) + 1;
                }
                System.out.println("Hotel ID: ");
                String hotelID = scan.nextLine();
                // scan.nextLine();
                System.out.println("RoomNo: ");
                String roomNo = scan.nextLine();
                // scan.nextLine();
                System.out.println("Maintenance Company ID: ");
                String cmpID = scan.nextLine();
                System.out.println("Repair Date(mm/dd/yyyy): ");
                String date = scan.nextLine();
                System.out.println("Description: ");
                String description = scan.nextLine();
                System.out.println("Repair Type: ");
                String type = scan.nextLine();
                System.out.println("Repair ID: " + id + "\nHotel ID: " + hotelID + "\nRoomNo: " + roomNo
                                + "\nMaintenance Company ID: " + cmpID + "\nRepair Date: " + date + "\nDescription: " +
                                + "\nRepair Type: " + type);
                String sql = "insert into repair(rid, hotelID, roomNo, mcompany, repairdate, description, repairtype) v
                                + "\'" + id + "\'" + "," + "\'" + hotelID + "\'" + "," + "\'" + roomNo + "\'" + "," + "
                                + "\'" + "," + "\'" + date + "\'" + "," + "\'" + description + "\'" + "," + "\'" + type
                                + ")";
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }

}// end addRepair
```

5: bookRoom(esql); break;

- Worked on by: Vivek

- Insert into

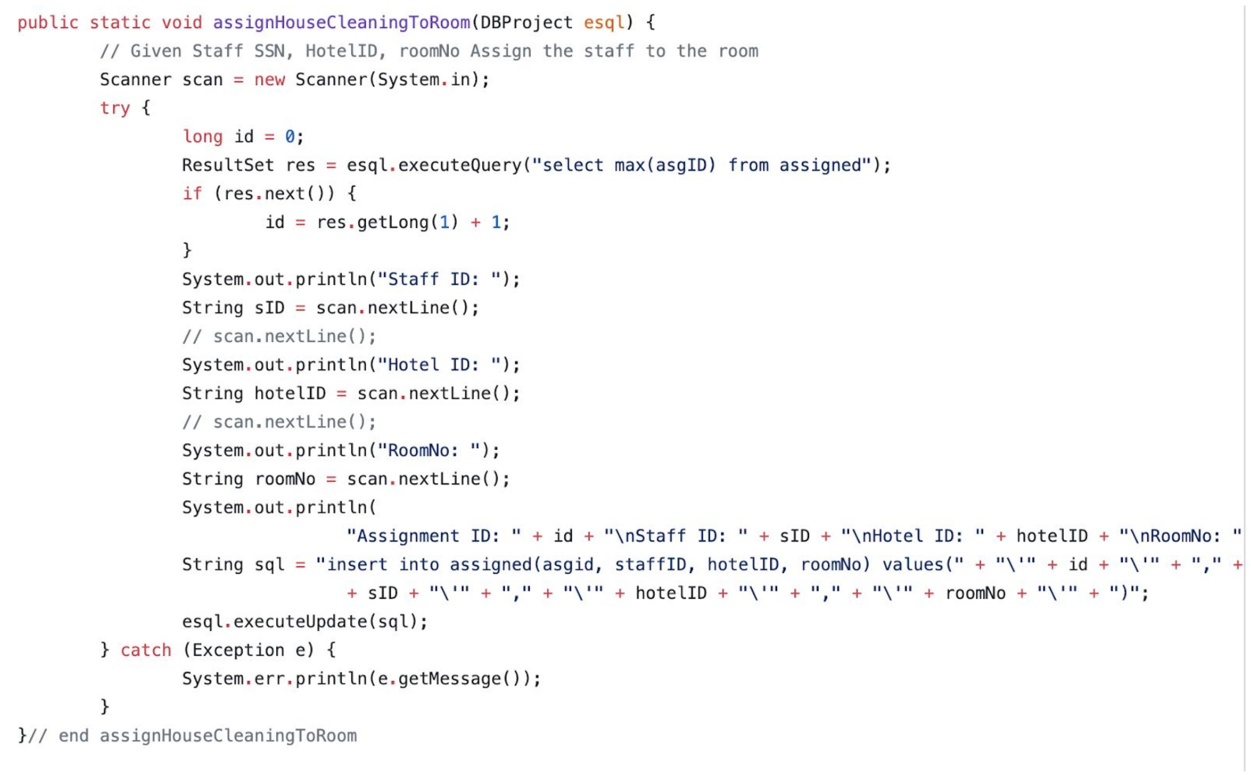


```java
public static void bookRoom(DBProject esql) {
        // Given hotelID, roomNo and customer Name create a booking in the DB
        Scanner scan = new Scanner(System.in);
        try {
                long id = 0;
                ResultSet res = esql.executeQuery("select max(bID) from booking");
                if (res.next()) {
                        id = res.getLong(1) + 1;
                }
                System.out.println("Customer ID: ");
                String cID = scan.nextLine();
                // scan.nextLine();
                System.out.println("Hotel ID: ");
                String hotelID = scan.nextLine();
                // scan.nextLine();
                System.out.println("RoomNo: ");
                String roomNo = scan.nextLine();
                System.out.println("Booking Date(mm/dd/yyyy): ");
                String date = scan.nextLine();
                System.out.println("Number of People: ");
                String numPpl = scan.nextLine();
                System.out.println("Price: ");
                String price = scan.nextLine();
                System.out.println("Booking ID: " + id + "\nCustomer ID: " + cID + "\nHotel ID: " + hotelID + "\nRoomNo
                                + roomNo + "\nBooking Date: " + date + "\nNumber of People: " + numPpl + "\nPrice: " +
                String sql = "insert into booking(bid, customer, hotelID, roomNo, bookingdate, noofpeople, price) value
                                + "\'" + id + "\'" + "," + "\'" + cID + "\'" + "," + "\'" + hotelID + "\'" + "," + "\'"
                                + "\'" + "," + "\'" + date + "\'" + "," + "\'" + numPpl + "\'" + "," + "\'" + price + "
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// end bookRoom
```

6: assignHouseCleaningToRoom(esql); break;

- Worked on by: Vivek

- Insert into assigned table after validating input.

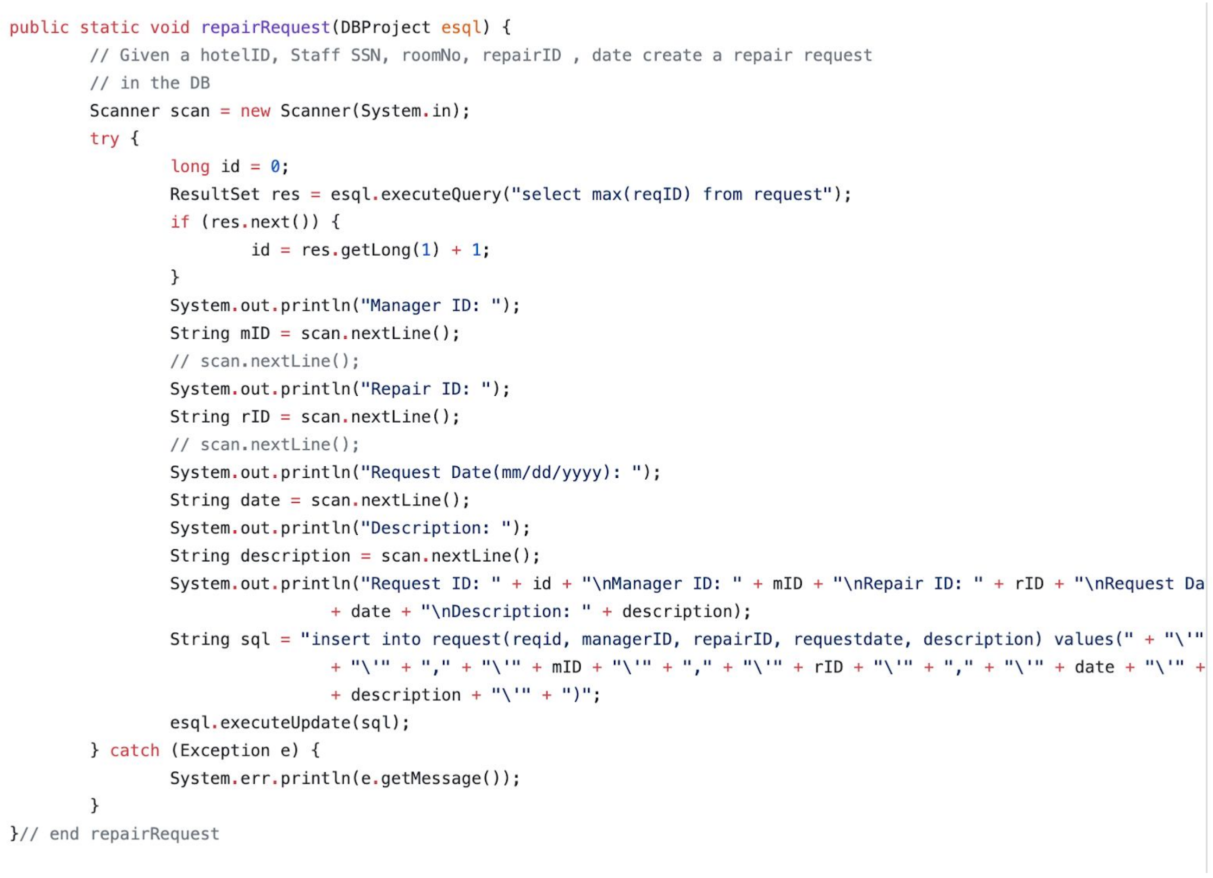


```
public static void assignHouseCleaningToRoom(DBProject esql) {
        // Given Staff SSN, HotelID, roomNo Assign the staff to the room
        Scanner scan = new Scanner(System.in);
        try {
                long id = 0;
                ResultSet res = esql.executeQuery("select max(asgID) from assigned");
                if (res.next()) {
                        id = res.getLong(1) + 1;
                }
                System.out.println("Staff ID: ");
                String sID = scan.nextLine();
                // scan.nextLine();
                System.out.println("Hotel ID: ");
                String hotelID = scan.nextLine();
                // scan.nextLine();
                System.out.println("RoomNo: ");
                String roomNo = scan.nextLine();
                System.out.println(
                                "Assignment ID: " + id + "\nStaff ID: " + sID + "\nHotel ID: " + hotelID + "\nRoomNo: "
                String sql = "insert into assigned(asgid, staffID, hotelID, roomNo) values(" + "\'" + id + "\'" + "," +
                                + sID + "\'" + "," + "\'" + hotelID + "\'" + "," + "\'" + roomNo + "\'" + ")";
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// end assignHouseCleaningToRoom
```

7: repairRequest(esql); break;

- Worked on by: Vivek

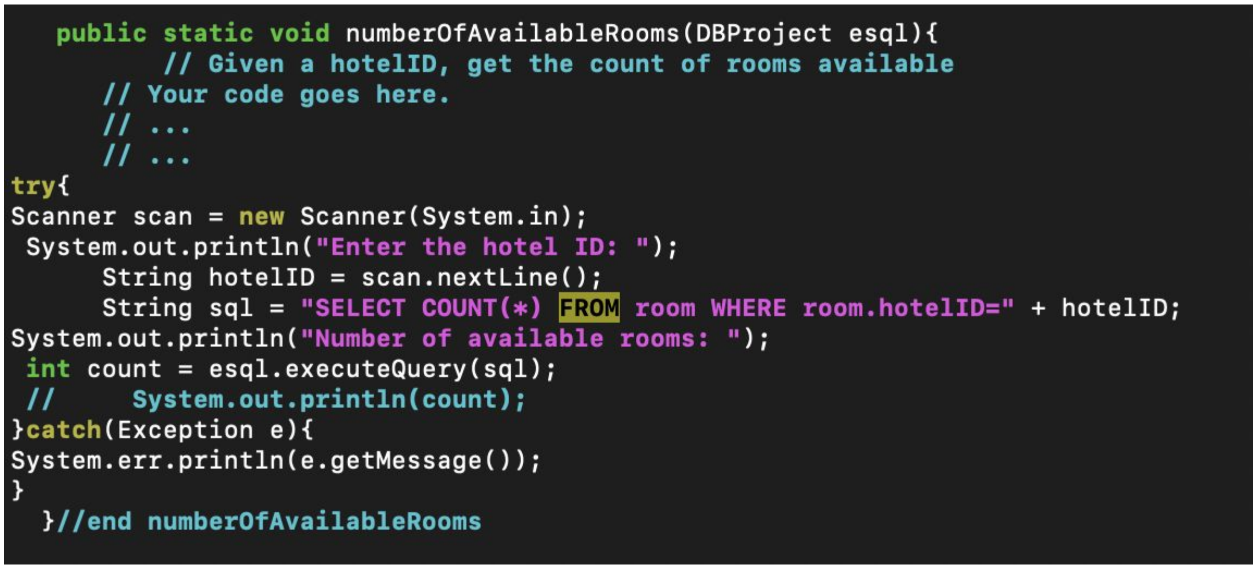
- Insert into request table after validating input.



```
public static void repairRequest(DBProject esql) {
        // Given a hotelID, Staff SSN, roomNo, repairID , date create a repair request
        // in the DB
        Scanner scan = new Scanner(System.in);
        try {
                long id = 0;
                ResultSet res = esql.executeQuery("select max(reqID) from request");
                if (res.next()) {
                        id = res.getLong(1) + 1;
                }
                System.out.println("Manager ID: ");
                String mID = scan.nextLine();
                // scan.nextLine();
                System.out.println("Repair ID: ");
                String rID = scan.nextLine();
                // scan.nextLine();
                System.out.println("Request Date(mm/dd/yyyy): ");
                String date = scan.nextLine();
                System.out.println("Description: ");
                String description = scan.nextLine();
                System.out.println("Request ID: " + id + "\nManager ID: " + mID + "\nRepair ID: " + rID + "\nRequest Da
                                + date + "\nDescription: " + description);
                String sql = "insert into request(reqid, managerID, repairID, requestdate, description) values(" + "\'"
                                + "\'" + "," + "\'" + mID + "\'" + "," + "\'" + rID + "\'" + "," + "\'" + date + "\'" +
                                + description + "\'" + ")";
                esql.executeUpdate(sql);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// end repairRequest
```

8: numberOfAvailableRooms(esql); break;

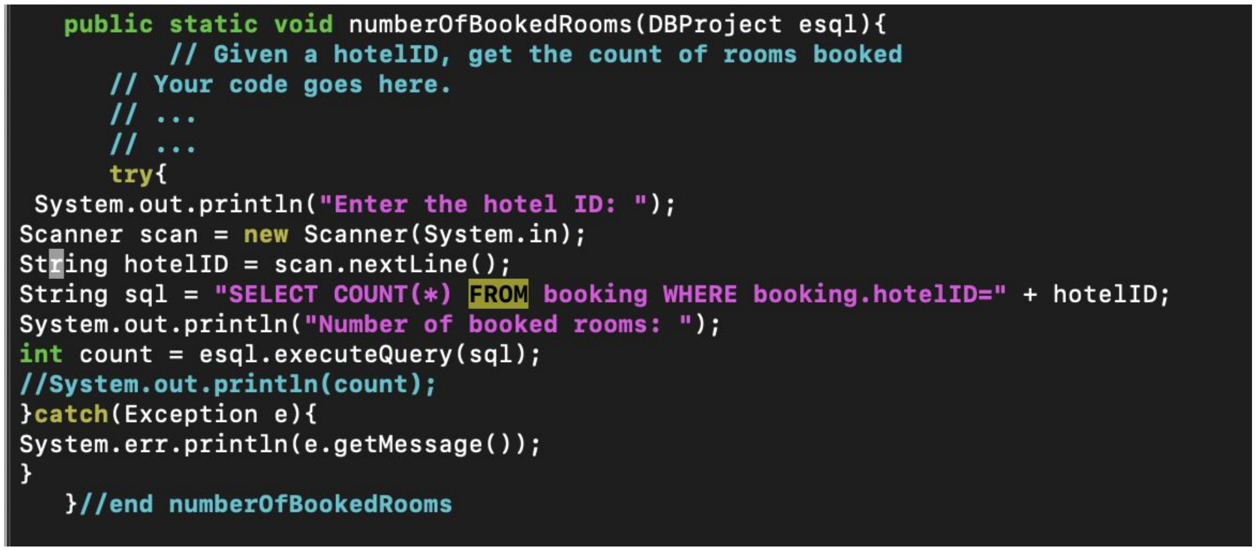
- Worked on by: Spyridon
- Standard SELECT statement returns the number of rooms at a given hotel, matched by the given hotelID to the the hotelID property of a room instance.



```
   public static void numberOfAvailableRooms(DBProject esql){
          // Given a hotelID, get the count of rooms available
      // Your code goes here.
      // ...
      // ...
try{
Scanner scan = new Scanner(System.in);
 System.out.println("Enter the hotel ID: ");
      String hotelID = scan.nextLine();
      String sql = "SELECT COUNT(*) FROM room WHERE room.hotelID=" + hotelID;
System.out.println("Number of available rooms: ");
 int count = esql.executeQuery(sql);
 //     System.out.println(count);
}catch(Exception e){
System.err.println(e.getMessage());
}
  }//end numberOfAvailableRooms
```

9: numberOfBookedRooms(esql); break;

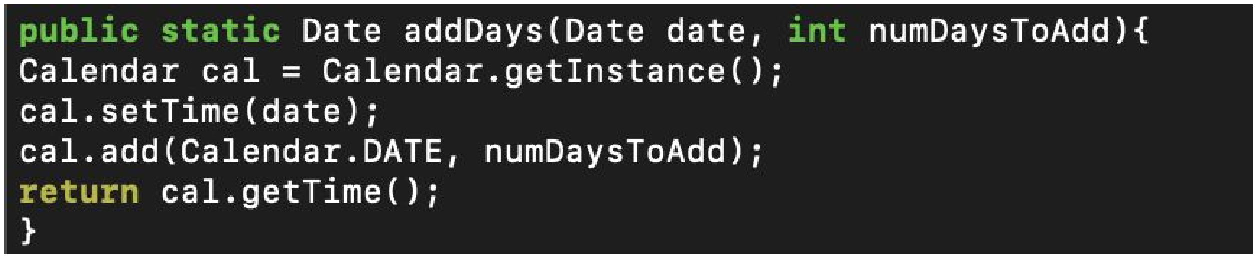
- Worked on by: Spyridon
- Standard SELECT statement returns the number of rooms booked for a given hotelID.



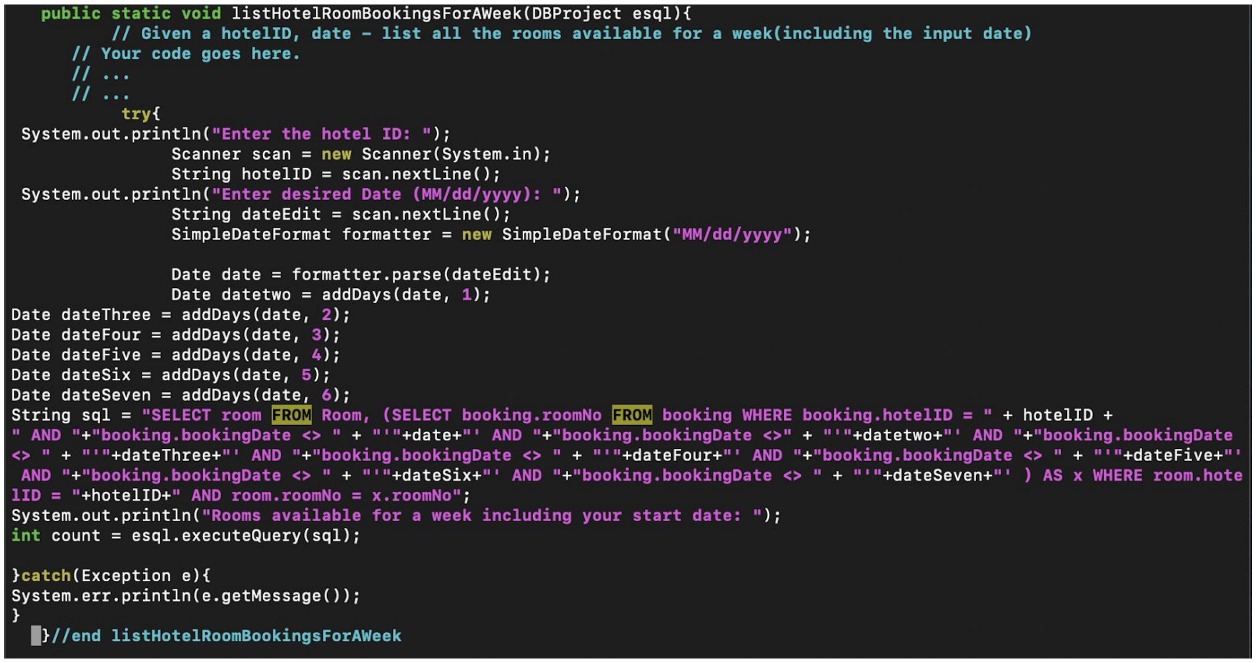
```
   public static void numberOfBookedRooms(DBProject esql){
          // Given a hotelID, get the count of rooms booked
      // Your code goes here.
      // ...
      // ...
      try{
 System.out.println("Enter the hotel ID: ");
Scanner scan = new Scanner(System.in);
String hotelID = scan.nextLine();
String sql = "SELECT COUNT(*) FROM booking WHERE booking.hotelID=" + hotelID;
System.out.println("Number of booked rooms: ");
int count = esql.executeQuery(sql);
//System.out.println(count);
}catch(Exception e){
System.err.println(e.getMessage());
}
   }//end numberOfBookedRooms
```

10: listHotelRoomBookingsForAWeek(esql); break;

- Worked on by: Spyridon
- Helper function: addDays is used to increment the user supplied date by a number of days. This helps the listHotelRoomBookingsForAWeek function calculate date variables that make up the entire week including the user provided start date.



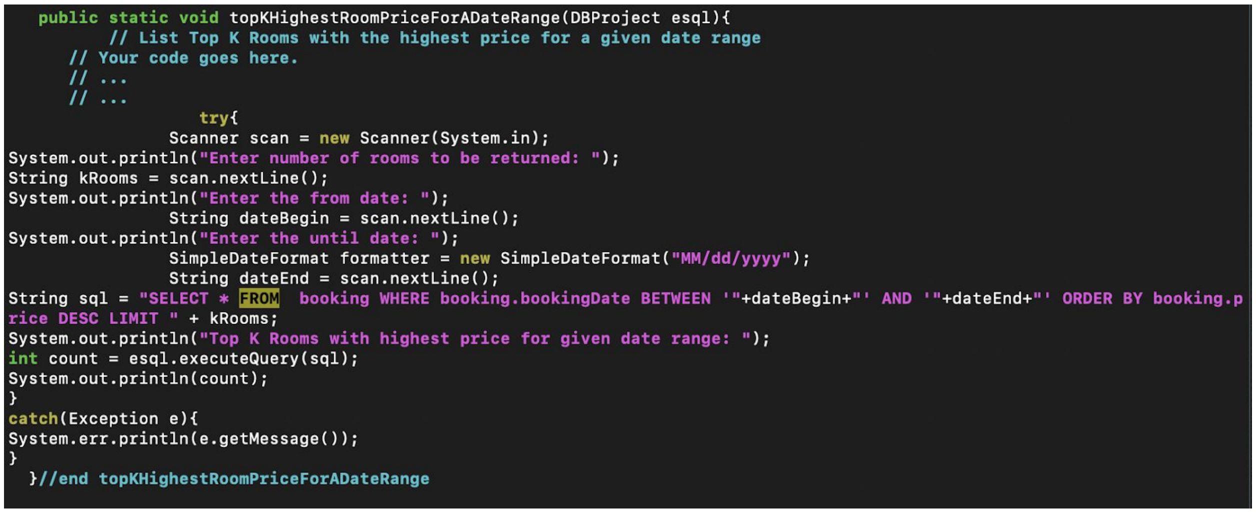
```java
public static Date addDays(Date date, int numDaysToAdd){
Calendar cal = Calendar.getInstance();
cal.setTime(date);
cal.add(Calendar.DATE, numDaysToAdd);
return cal.getTime();
}
```



```java
   public static void listHotelRoomBookingsForAWeek(DBProject esql){
          // Given a hotelID, date - list all the rooms available for a week(including the input date)
      // Your code goes here.
      // ...
      // ...
           try{
 System.out.println("Enter the hotel ID: ");
                Scanner scan = new Scanner(System.in);
                String hotelID = scan.nextLine();
 System.out.println("Enter desired Date (MM/dd/yyyy): ");
                String dateEdit = scan.nextLine();
                SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

                Date date = formatter.parse(dateEdit);
                Date datetwo = addDays(date, 1);
Date dateThree = addDays(date, 2);
Date dateFour = addDays(date, 3);
Date dateFive = addDays(date, 4);
Date dateSix = addDays(date, 5);
Date dateSeven = addDays(date, 6);
String sql = "SELECT room FROM Room, (SELECT booking.roomNo FROM booking WHERE booking.hotelID = " + hotelID +
" AND "+"booking.bookingDate <> " + "'"+date+"' AND "+"booking.bookingDate <>" + "'"+datetwo+"' AND "+"booking.bookingDate
<> " + "'"+dateThree+"' AND "+"booking.bookingDate <> " + "'"+dateFour+"' AND "+"booking.bookingDate <> " + "'"+dateFive+"'
 AND "+"booking.bookingDate <> " + "'"+dateSix+"' AND "+"booking.bookingDate <> " + "'"+dateSeven+"' ) AS x WHERE room.hote
lID = "+hotelID+" AND room.roomNo = x.roomNo";
System.out.println("Rooms available for a week including your start date: ");
int count = esql.executeQuery(sql);

}catch(Exception e){
System.err.println(e.getMessage());
}
  }//end listHotelRoomBookingsForAWeek
```

11: topKHighestRoomPriceForADateRange(esql); break;

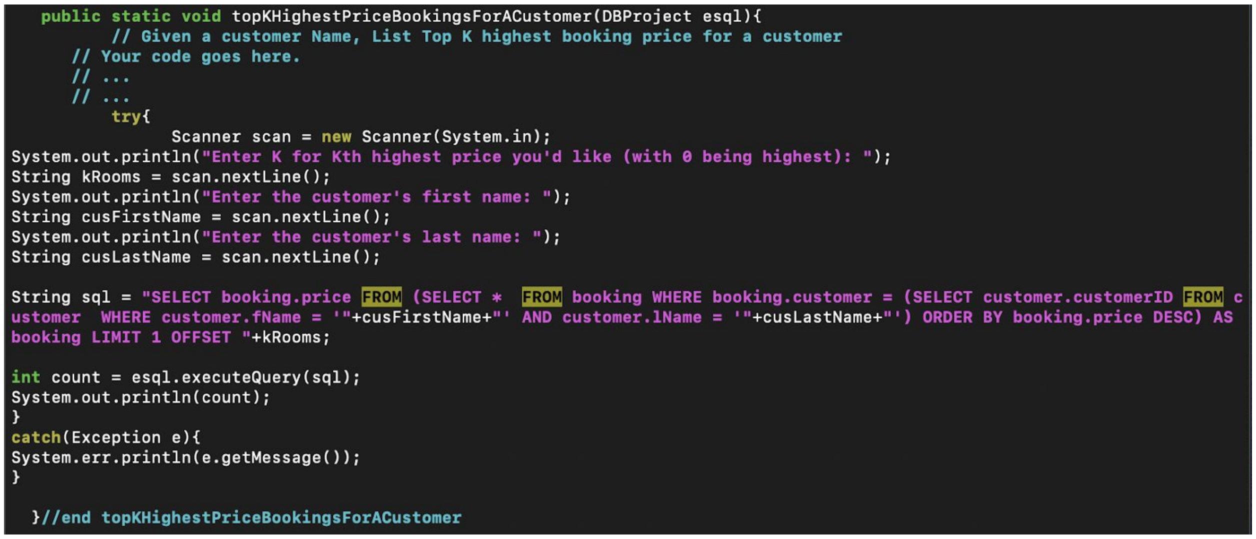
- Worked on by: Spyridon
- Here we find room information in bookings by searching within the given date range and limiting results by the K provided number.



```java
   public static void topKHighestRoomPriceForADateRange(DBProject esql){
          // List Top K Rooms with the highest price for a given date range
      // Your code goes here.
      // ...
      // ...
                  try{
                Scanner scan = new Scanner(System.in);
System.out.println("Enter number of rooms to be returned: ");
String kRooms = scan.nextLine();
System.out.println("Enter the from date: ");
                String dateBegin = scan.nextLine();
System.out.println("Enter the until date: ");
                SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");
                String dateEnd = scan.nextLine();
String sql = "SELECT * FROM  booking WHERE booking.bookingDate BETWEEN '"+dateBegin+"' AND '"+dateEnd+"' ORDER BY booking.p
rice DESC LIMIT " + kRooms;
System.out.println("Top K Rooms with highest price for given date range: ");
int count = esql.executeQuery(sql);
System.out.println(count);
}
catch(Exception e){
System.err.println(e.getMessage());
}
  }//end topKHighestRoomPriceForADateRange
```

12: topKHighestPriceBookingsForACustomer(esql); break;

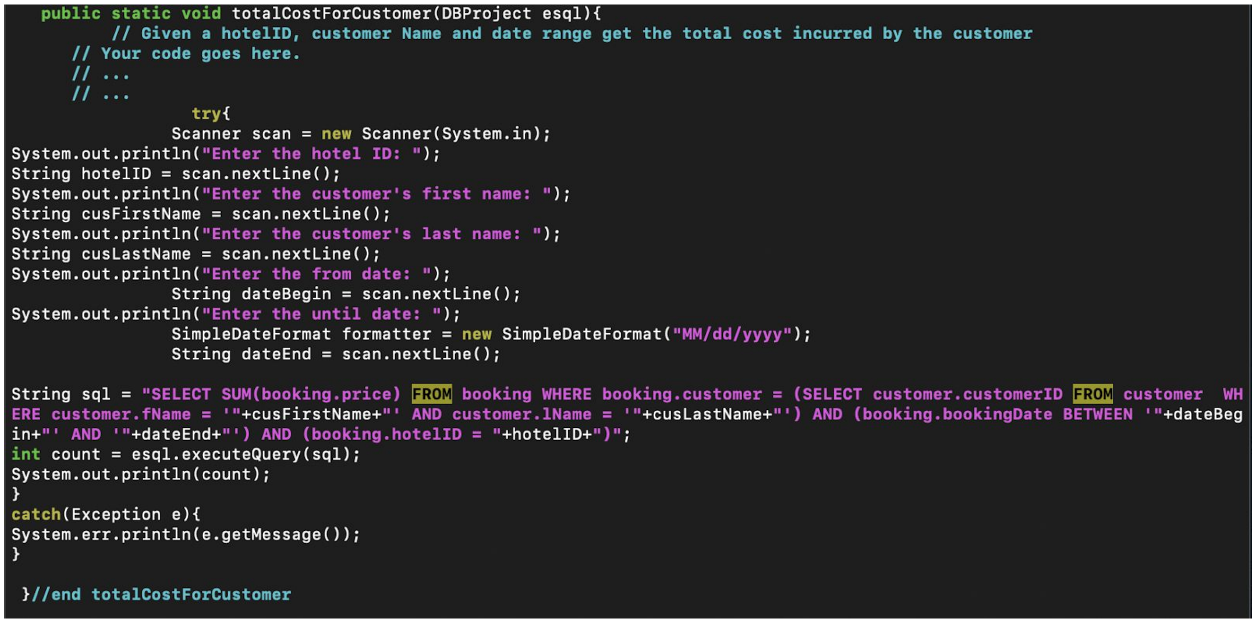
- Worked on by: Spyridon
- Here we use an inner inner query to match the customer name provided with a customerID then we use that in an inner query to find bookings from that customer ordered by price in descending order. The outermost query takes the first booking in that list.



```
   public static void topKHighestPriceBookingsForACustomer(DBProject esql){
          // Given a customer Name, List Top K highest booking price for a customer
      // Your code goes here.
      // ...
      // ...
          try{
                Scanner scan = new Scanner(System.in);
System.out.println("Enter K for Kth highest price you'd like (with 0 being highest): ");
String kRooms = scan.nextLine();
System.out.println("Enter the customer's first name: ");
String cusFirstName = scan.nextLine();
System.out.println("Enter the customer's last name: ");
String cusLastName = scan.nextLine();

String sql = "SELECT booking.price FROM (SELECT *  FROM booking WHERE booking.customer = (SELECT customer.customerID FROM c
ustomer  WHERE customer.fName = '"+cusFirstName+"' AND customer.lName = '"+cusLastName+"') ORDER BY booking.price DESC) AS
booking LIMIT 1 OFFSET "+kRooms;

int count = esql.executeQuery(sql);
System.out.println(count);
}
catch(Exception e){
System.err.println(e.getMessage());
}

  }//end topKHighestPriceBookingsForACustomer
```

13: totalCostForCustomer(esql); break;

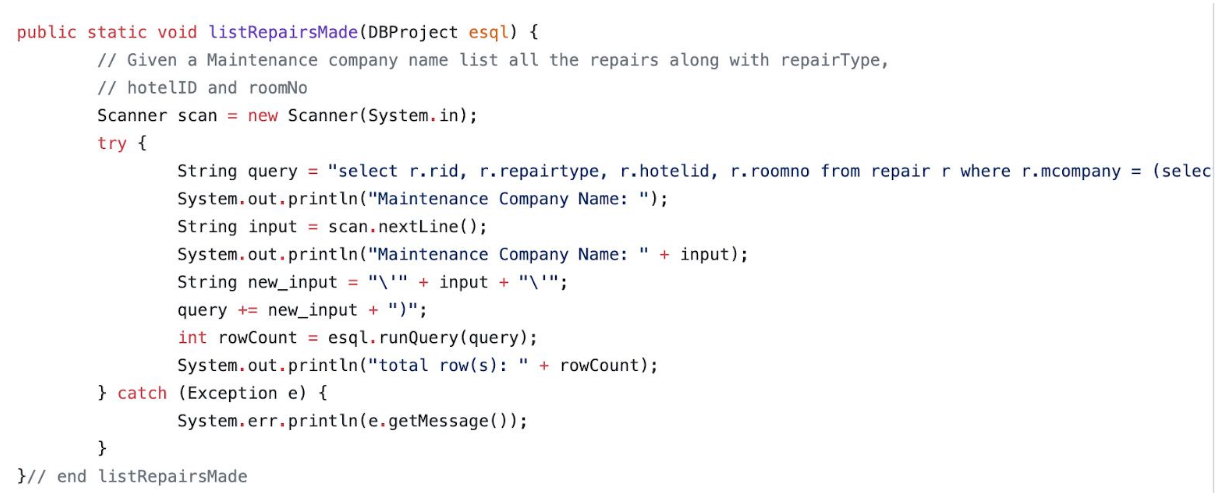
- Worked on by: Spyridon
- Here we use SUM() to sum all the costs for a customer's bookings from a given date range.



```
   public static void totalCostForCustomer(DBProject esql){
          // Given a hotelID, customer Name and date range get the total cost incurred by the customer
      // Your code goes here.
      // ...
      // ...
                try{
                Scanner scan = new Scanner(System.in);
System.out.println("Enter the hotel ID: ");
String hotelID = scan.nextLine();
System.out.println("Enter the customer's first name: ");
String cusFirstName = scan.nextLine();
System.out.println("Enter the customer's last name: ");
String cusLastName = scan.nextLine();
System.out.println("Enter the from date: ");
                String dateBegin = scan.nextLine();
System.out.println("Enter the until date: ");
                SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");
                String dateEnd = scan.nextLine();

String sql = "SELECT SUM(booking.price) FROM booking WHERE booking.customer = (SELECT customer.customerID FROM customer  WH
ERE customer.fName = '"+cusFirstName+"' AND customer.lName = '"+cusLastName+"') AND (booking.bookingDate BETWEEN '"+dateBeg
in+"' AND '"+dateEnd+"') AND (booking.hotelID = "+hotelID+")";
int count = esql.executeQuery(sql);
System.out.println(count);
}
catch(Exception e){
System.err.println(e.getMessage());
}

 }//end totalCostForCustomer
```

14: listRepairsMade(esql); break;

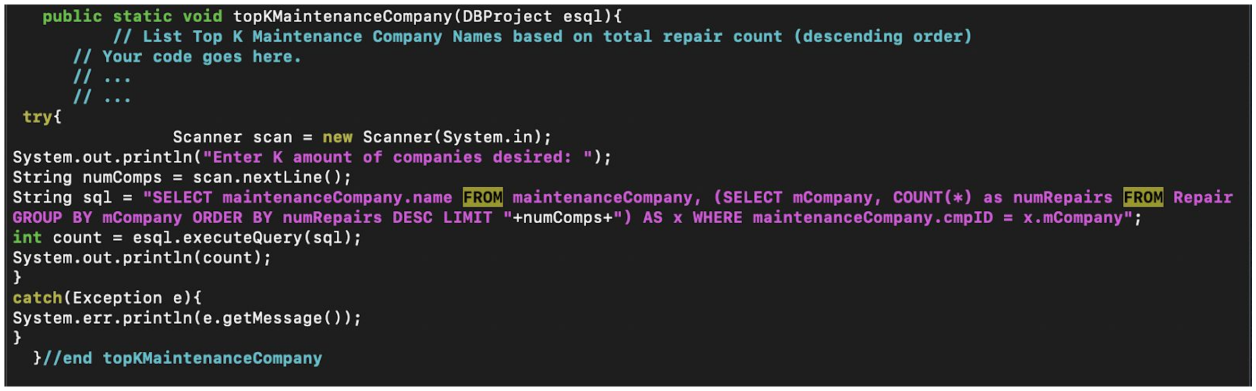
- Worked on by: Vivek
- Use sub-query to get maintenance company ID from user input name, then query repairs table using that ID.



```
public static void listRepairsMade(DBProject esql) {
        // Given a Maintenance company name list all the repairs along with repairType,
        // hotelID and roomNo
        Scanner scan = new Scanner(System.in);
        try {
                String query = "select r.rid, r.repairtype, r.hotelid, r.roomno from repair r where r.mcompany = (selec
                System.out.println("Maintenance Company Name: ");
                String input = scan.nextLine();
                System.out.println("Maintenance Company Name: " + input);
                String new_input = "\'" + input + "\'";
                query += new_input + ")";
                int rowCount = esql.runQuery(query);
                System.out.println("total row(s): " + rowCount);
        } catch (Exception e) {
                System.err.println(e.getMessage());
        }
}// end listRepairsMade
```

15: topKMaintenanceCompany(esql); break;

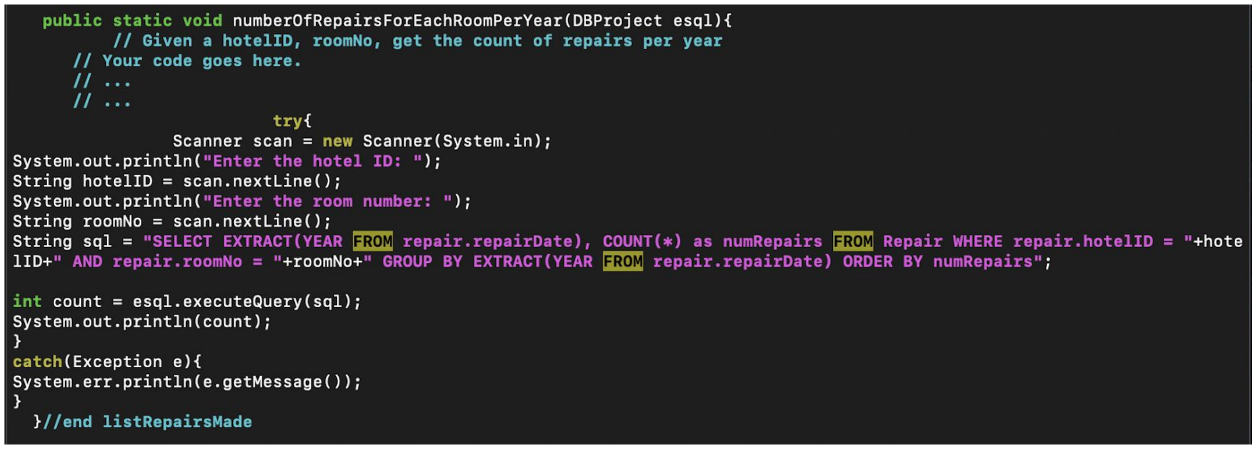
- Worked on by: Spyridon
- We use an inner query to get K companies ordered by number of repairs. The outermost query gets the names of these companies.



```
   public static void topKMaintenanceCompany(DBProject esql){
          // List Top K Maintenance Company Names based on total repair count (descending order)
      // Your code goes here.
      // ...
      // ...
 try{
                Scanner scan = new Scanner(System.in);
System.out.println("Enter K amount of companies desired: ");
String numComps = scan.nextLine();
String sql = "SELECT maintenanceCompany.name FROM maintenanceCompany, (SELECT mCompany, COUNT(*) as numRepairs FROM Repair
GROUP BY mCompany ORDER BY numRepairs DESC LIMIT "+numComps+") AS x WHERE maintenanceCompany.cmpID = x.mCompany";
int count = esql.executeQuery(sql);
System.out.println(count);
}
catch(Exception e){
System.err.println(e.getMessage());
}
  }//end topKMaintenanceCompany
```

16: numberOfRepairsForEachRoomPerYear(esql); break;

- Worked on by: Spyridon
- Here we use Extract(Year(x)) to get the number of repairs for a year as well as order the results by year.



```
   public static void numberOfRepairsForEachRoomPerYear(DBProject esql){
          // Given a hotelID, roomNo, get the count of repairs per year
      // Your code goes here.
      // ...
      // ...
                          try{
                Scanner scan = new Scanner(System.in);
System.out.println("Enter the hotel ID: ");
String hotelID = scan.nextLine();
System.out.println("Enter the room number: ");
String roomNo = scan.nextLine();
String sql = "SELECT EXTRACT(YEAR FROM repair.repairDate), COUNT(*) as numRepairs FROM Repair WHERE repair.hotelID = "+hote
lID+" AND repair.roomNo = "+roomNo+" GROUP BY EXTRACT(YEAR FROM repair.repairDate) ORDER BY numRepairs";

int count = esql.executeQuery(sql);
System.out.println(count);
}
catch(Exception e){
System.err.println(e.getMessage());
}
  }//end listRepairsMade
```