Inverted Exam
Lösung

Dokumentation der gefundenen Performance Probleme & deren Lösungsumsetzung:

1.)
Im CustomerDAO können in der Methode reserve() gleich die übergebenen IDs verwendet werden anstatt zuerst ein Seat- und Screening-Objekt zu erstellen.

Problem:

```
public int reserve(int seat, int screening){
    SeatDAO s = new SeatDAO();
    ScreeningDAO sc = new ScreeningDAO();
    BookingDAO b = new BookingDAO();
    Booking booking = new Booking();

    int len = b.readAllBookings().size() + 1; // +1 for a new id

    Seat se = s.read(seat);
    Screening scr = sc.read(screening);

    // before a booking can be made
    // check if free or not
    // if free
    if (s.isFree(se.id, scr.id)) {
        //make booking
        booking.id = len;
        booking.seat = seat;
        booking.screening = screening;
        b.create(booking);
        System.out.println("Seat is free");
        return 1;
    }

   System.out.println("Seat is taken");
   return 0;
}
```

Lösung:

```
    public int reserve(int seat, int screening){
        SeatDAO s = new SeatDAO();
        BookingDAO b = new BookingDAO();
        Booking booking = new Booking();

        int len = b.readAllBookings().size() + 1; // +1 for a new id

        // before a booking can be made
        // check if free or not
        // if free
        if (s.isFree(seat, screening)) {

            //make a booking
            booking.id = len;
            booking.seat = seat;
            booking.screening = screening;
            b.create(booking);

            System.out.println("Seat free");
            return 1;
        }

       System.out.println("Seat taken");
       return 0;
    }
}
```

2.)

Des Weiteren können im SeatDAO in der Methode isFree() ebenfalls sofort die übergebenen IDs verwendet werden anstatt zuerst ein Seat- und Screening-Objekt zu erstellen.

Problem:

```
public boolean isFree(int seat, int screening) {
    PreparedStatement stmt;
    Check c = new Check();

    //check if seat is in hall beforehand
    if (c.seatInHall(seat, screening)) {

        // if seat is in hall
        System.out.println("Seat is in hall");

        // add new screening and seat
        ScreeningDAO scdao = new ScreeningDAO();
        Seat s = new Seat();

        // read from existing seats
        List<Seat> seats = readAllSeats();
        for (Seat se : seats){
            if(se.id == seat) {
                s = read(seat);
            }
        }

        Screening sc = scdao.read(screening);

        try {
            // check if a booking for a certain seat id with a certain creening id exists
            stmt = conn.prepareStatement("SELECT * FROM BOOKING WHERE FK_SEAT = ? AND FK_SCREENING = ?");
            stmt.setInt(1, s.id);
            stmt.setInt(2, sc.id);
            ResultSet rs = stmt.executeQuery();
```

Lösung:

```
public boolean isFree(int seat, int screening) {
    PreparedStatement stmt;
    Check c = new Check();

    //check if seat is in hall beforehand
    if (c.seatInHall(seat, screening)) {

        // if seat is in hall
        System.out.println("Seat is in hall");

        try {
            // check if a booking for a certain seat id with a certain creening id exists
            stmt = conn.prepareStatement("SELECT * FROM BOOKING WHERE FK_SEAT = ? AND FK_SCREENING = ?");
            stmt.setInt(1, seat);
            stmt.setInt(2, screening);
            ResultSet rs = stmt.executeQuery();

            //if a booking exists return false
            if (rs.first()){
                System.out.println("Seat is not bookable");
                return false;
            }
            //if no booking for this seat at this screening exists, return true
            else {
                return true;
            }
        }
```

3.)

Zusätzlich werden unsinnigerweise im SeatDAO in der Methode isFree() alle Seats ausgelesen. Danach wird überprüft, ob die IDs mit der übergebenen übereinstimmen und wenn ja, wird der Seat in der Variable, die in Lösungspunkt 2 erstellt worden ist, gespeichert.

Problem:

```
public boolean isFree(int seat, int screening) {
    PreparedStatement stmt;
    Check c = new Check();

    //check if seat is in hall beforehand
    if (c.seatInHall(seat, screening)) {

        // if seat is in hall
        System.out.println("Seat is in hall");

        // add new screening and seat
        ScreeningDAO scdao = new ScreeningDAO();
        Seat s = new Seat();

        // read from existing seats
        List<Seat> seats = readAllSeats();
        for (Seat se : seats){
            if(se.id == seat) {
                s = read(seat);
            }
        }

        Screening sc = scdao.read(screening);
```

Lösung:

```
public boolean isFree(int seat, int screening) {
    PreparedStatement stmt;
    Check c = new Check();

    //check if seat is in hall beforehand
    if (c.seatInHall(seat, screening)) {

        // if seat is in hall
        System.out.println("seat in hall");

        try {
            // check if a booking for a certain seat id with a certain creening id exists
            stmt = conn.prepareStatement("SELECT * FROM BOOKING WHERE FK_SEAT = ? AND FK_SCREENING = ?");
            stmt.setInt(1, seat);
            stmt.setInt(2, screening);
            ResultSet rs = stmt.executeQuery();

            //if a booking exists return false
            if (rs.first()){
                System.out.println("seat not bookable");
                return false;
            }
            //if no booking for this seat at this screening exists, return true
            else {
                return true;
            }
        }
        catch (SQLException e) {
            e.printStackTrace();
            System.err.println("Booking failed.");
        }
    }
    return false;
```

4.)

In der Klasse Check gibt es die Methode seatInHall(), in welcher überprüft wird – wie der Name schon sagt – ob sich ein bestimmter Seat in einer bestimmten Hall befindet. Dafür wird mitunter eine SQL-Abfrage nötig, doch bei der gegebenen SELECT-Abfrage sind einige Unterabfragen irrelevant. Wichtig ist ja nur, ob ein Seat mit ID x und ein Screening mit ID x dieselbe hall ID x in der „gejointen“ View seatsPerScreening besitzen, also ob der geforderte Seat bei dem betreffenden Screening in dem entsprechenden Saal ist oder nicht.

Problem:

```
public boolean seatInHall(int seat, int screening) {

    PreparedStatement stmt;

    try {
        stmt = conn.prepareStatement("SELECT "
                + "sps.seat, "
                + "sps.screening, "
                + "(SELECT count(*) FROM seat), "
                + "(SELECT COUNT(*) FROM screening), "
                + "SUM(sps.seat + sps.screening) "
                + "FROM seatsPerScreening sps "
                + "WHERE seat = (SELECT id FROM seat WHERE id = ?) "
                + "AND screening = (SELECT id FROM screening WHERE id = ?)");
        stmt.setInt(1, seat);
        stmt.setInt(2, screening);
        ResultSet rs = stmt.executeQuery();

        System.out.println("Checking if your seat is in hall...");
        if (rs.first()){
            System.out.println("It is in hall");
            return true;
        }
        else {
            System.out.println("It is not in hall");
            return false;

        }
```

Lösung:

```
public boolean seatInHall(int seat, int screening) {

    PreparedStatement stmt;

    try {
        stmt = conn.prepareStatement("SELECT * FROM seatsPerScreening WHERE seat = ? AND screening = ?");
        stmt.setInt(1, seat);
        stmt.setInt(2, screening);
        ResultSet rs = stmt.executeQuery();

        System.out.println("Checking if your seat is in hall...");
        if (rs.first()){
            System.out.println("It is in hall");
            return true;
        }
        else {
            System.out.println("It is not in hall");
            return false;

        }

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

}
```