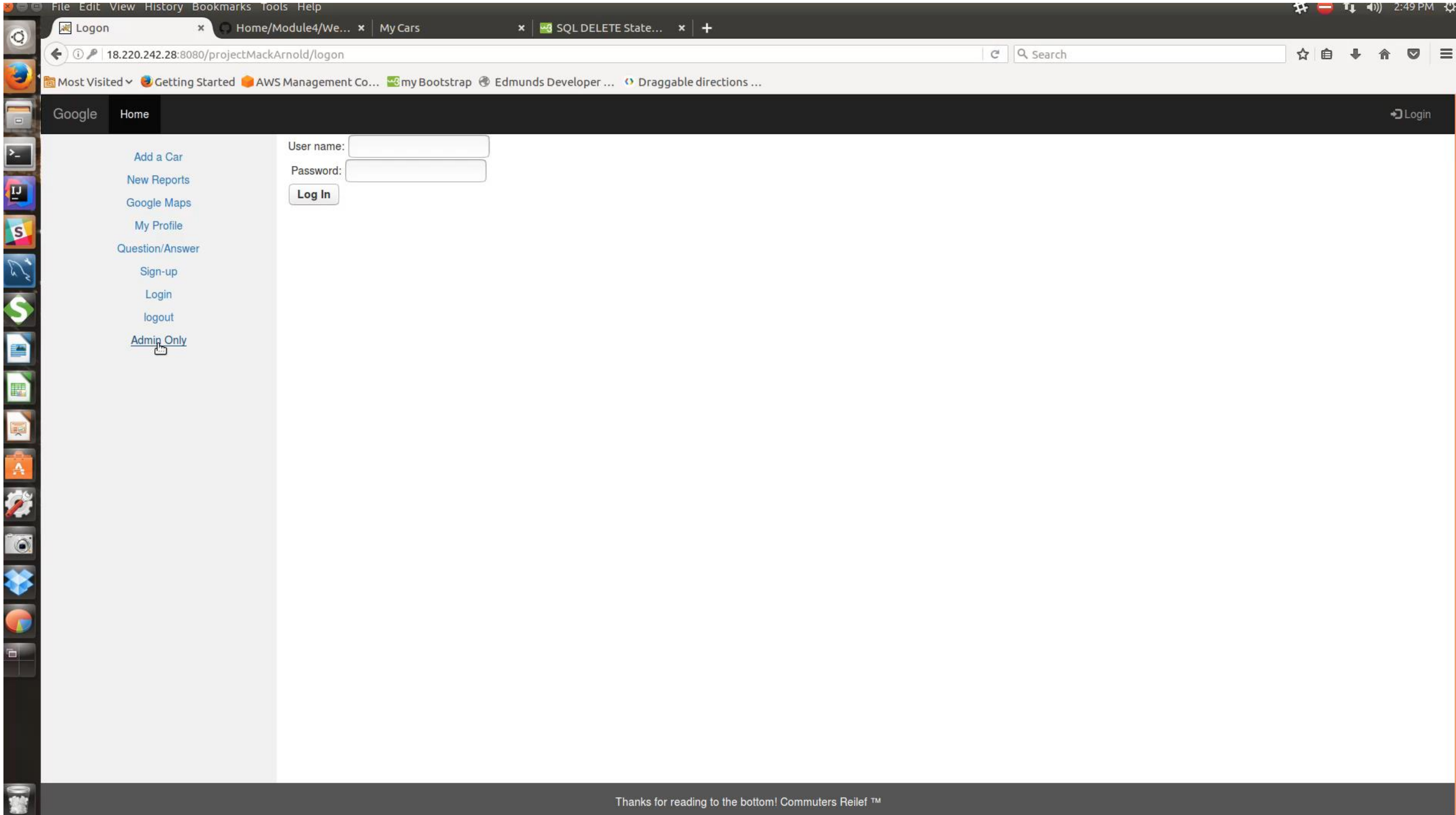
Final presentation

By: Mack Arnold

Commuter Releif

- This application will store your cars
- It will store your reports you make with the correct calculations
- Soon it will be able to email it to your employer
- You can delete cars and reports to your liking
- Only the Admin(me) can delete users.

Video Demo



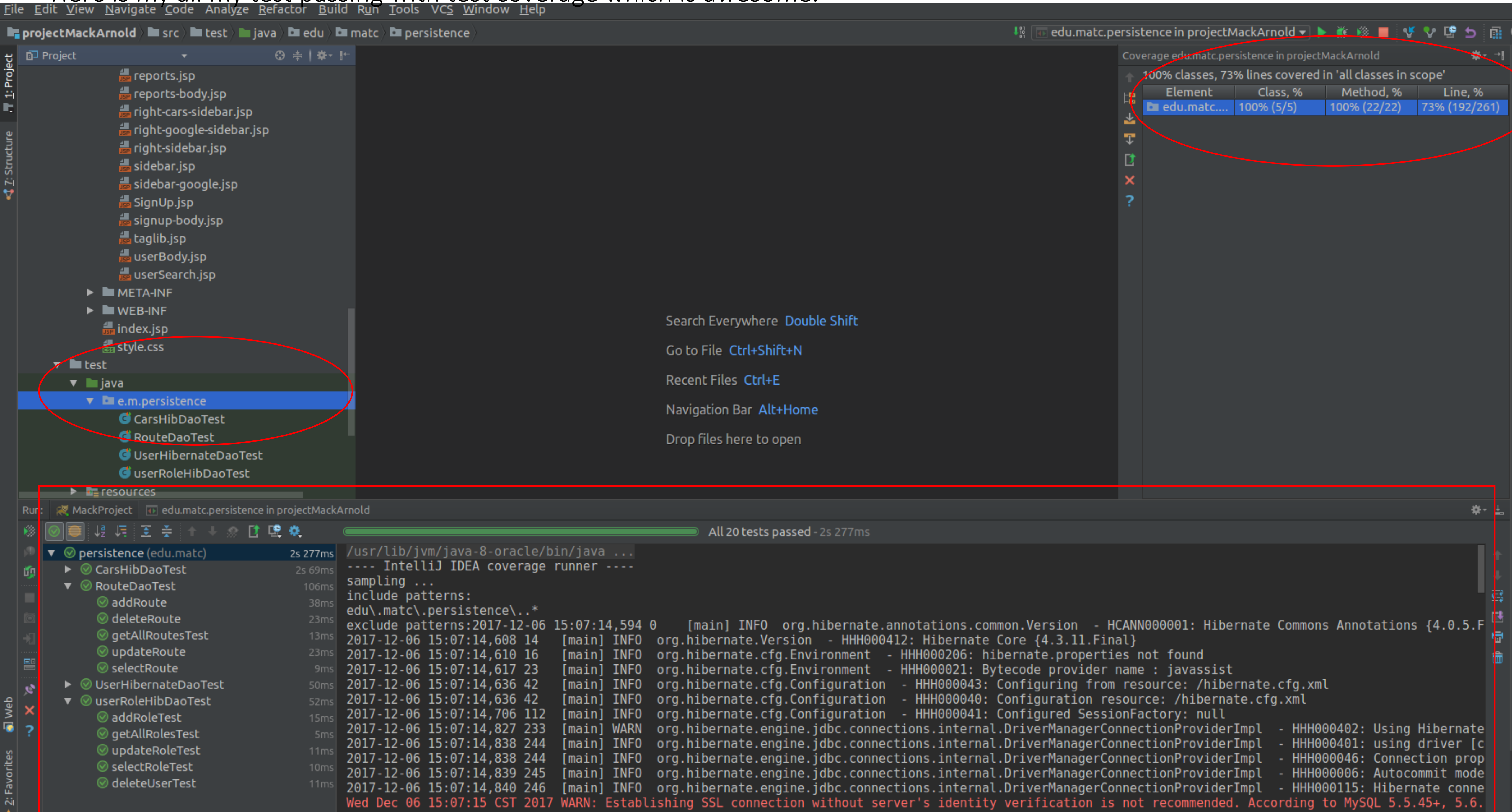
User name:

Password:

Log In

Thanks for reading to the bottom! Commuters Reilef ™

Here is my all my test passing with test coverage which is awesome.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
projectMackArnold › src › test › java › edu › matc › persistence
edu.matc.persistence in projectMackArnold
Project
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open
Run: MackProject edu.matc.persistence in projectMackArnold

```
reports.jsp
reports-body.jsp
right-cars-sidebar.jsp
right-google-sidebar.jsp
right-sidebar.jsp
sidebar.jsp
sidebar-google.jsp
SignUp.jsp
signup-body.jsp
taglib.jsp
userBody.jsp
userSearch.jsp
▶ META-INF
▶ WEB-INF
index.jsp
style.css
▼ test
  ▼ java
    ▼ e.m.persistence
        CarsHibDaoTest
        RouteDaoTest
        UserHibernateDaoTest
        userRoleHibDaoTest
  ▶ resources
```

Coverage edu.matc.persistence in projectMackArnold

100% classes, 73% lines covered in 'all classes in scope'

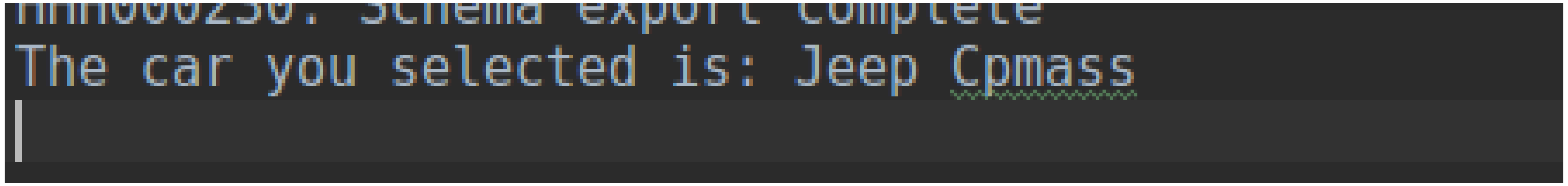
Element	Class, %	Method, %	Line, %
edu.matc....	100% (5/5)	100% (22/22)	73% (192/261)

All 20 tests passed - 2s 277ms

```
▼ persistence (edu.matc)        2s 277ms
  ▶ CarsHibDaoTest              2s 69ms
  ▼ RouteDaoTest                106ms
      addRoute                  38ms
      deleteRoute               23ms
      getAllRoutesTest          13ms
      updateRoute               23ms
      selectRoute               9ms
  ▶ UserHibernateDaoTest        50ms
  ▼ userRoleHibDaoTest          52ms
      addRoleTest               15ms
      getAllRolesTest           5ms
      updateRoleTest            11ms
      selectRoleTest            10ms
      deleteUserTest            11ms
```

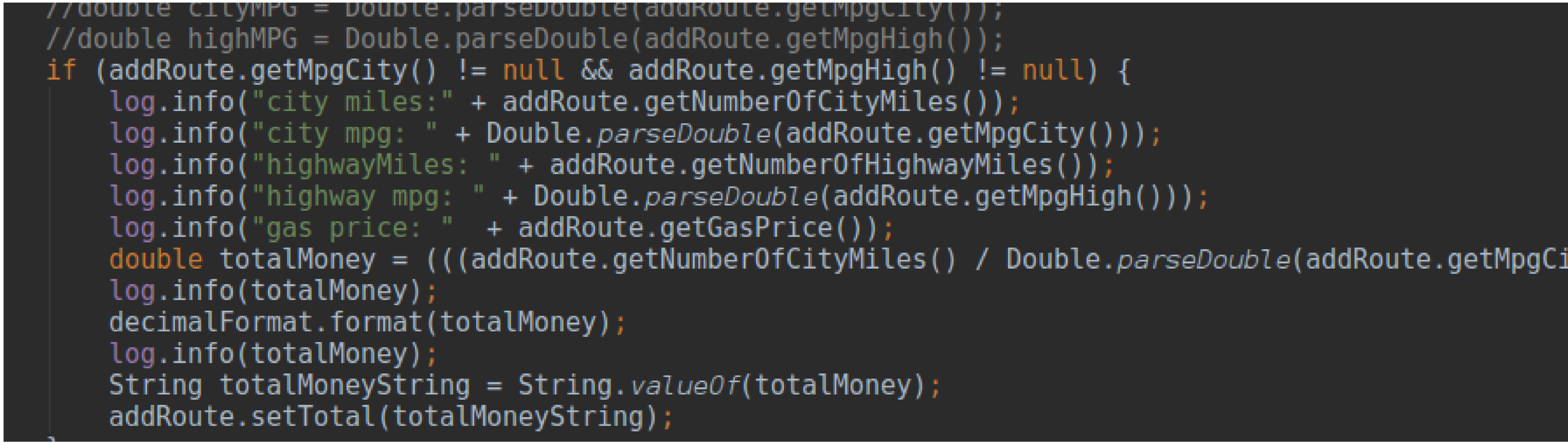
```
/usr/lib/jvm/java-8-oracle/bin/java ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
edu\.matc\.persistence\..*
exclude patterns:2017-12-06 15:07:14,594 0    [main] INFO  org.hibernate.annotations.common.Version  - HCANN000001: Hibernate Commons Annotations {4.0.5.F
2017-12-06 15:07:14,608 14   [main] INFO  org.hibernate.Version  - HHH000412: Hibernate Core {4.3.11.Final}
2017-12-06 15:07:14,610 16   [main] INFO  org.hibernate.cfg.Environment  - HHH000206: hibernate.properties not found
2017-12-06 15:07:14,617 23   [main] INFO  org.hibernate.cfg.Environment  - HHH000021: Bytecode provider name : javassist
2017-12-06 15:07:14,636 42   [main] INFO  org.hibernate.cfg.Configuration  - HHH000043: Configuring from resource: /hibernate.cfg.xml
2017-12-06 15:07:14,636 42   [main] INFO  org.hibernate.cfg.Configuration  - HHH000040: Configuration resource: /hibernate.cfg.xml
2017-12-06 15:07:14,706 112  [main] INFO  org.hibernate.cfg.Configuration  - HHH000041: Configured SessionFactory: null
2017-12-06 15:07:14,827 233  [main] WARN  org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl  - HHH000402: Using Hibernate
2017-12-06 15:07:14,838 244  [main] INFO  org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl  - HHH000401: using driver [c
2017-12-06 15:07:14,838 244  [main] INFO  org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl  - HHH000046: Connection prop
2017-12-06 15:07:14,839 245  [main] INFO  org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl  - HHH000006: Autocommit mode
2017-12-06 15:07:14,840 246  [main] INFO  org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl  - HHH000115: Hibernate conne
Wed Dec 06 15:07:15 CST 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.
```

Here is an example of me running a test to select a car and logging it.



```
HHH000230: Schema export complete
The car you selected is: Jeep Cpmass
```

Here is an example of my error handling, this is when I was trying to make sure that I'm getting all the components of the route, and adding it to the database



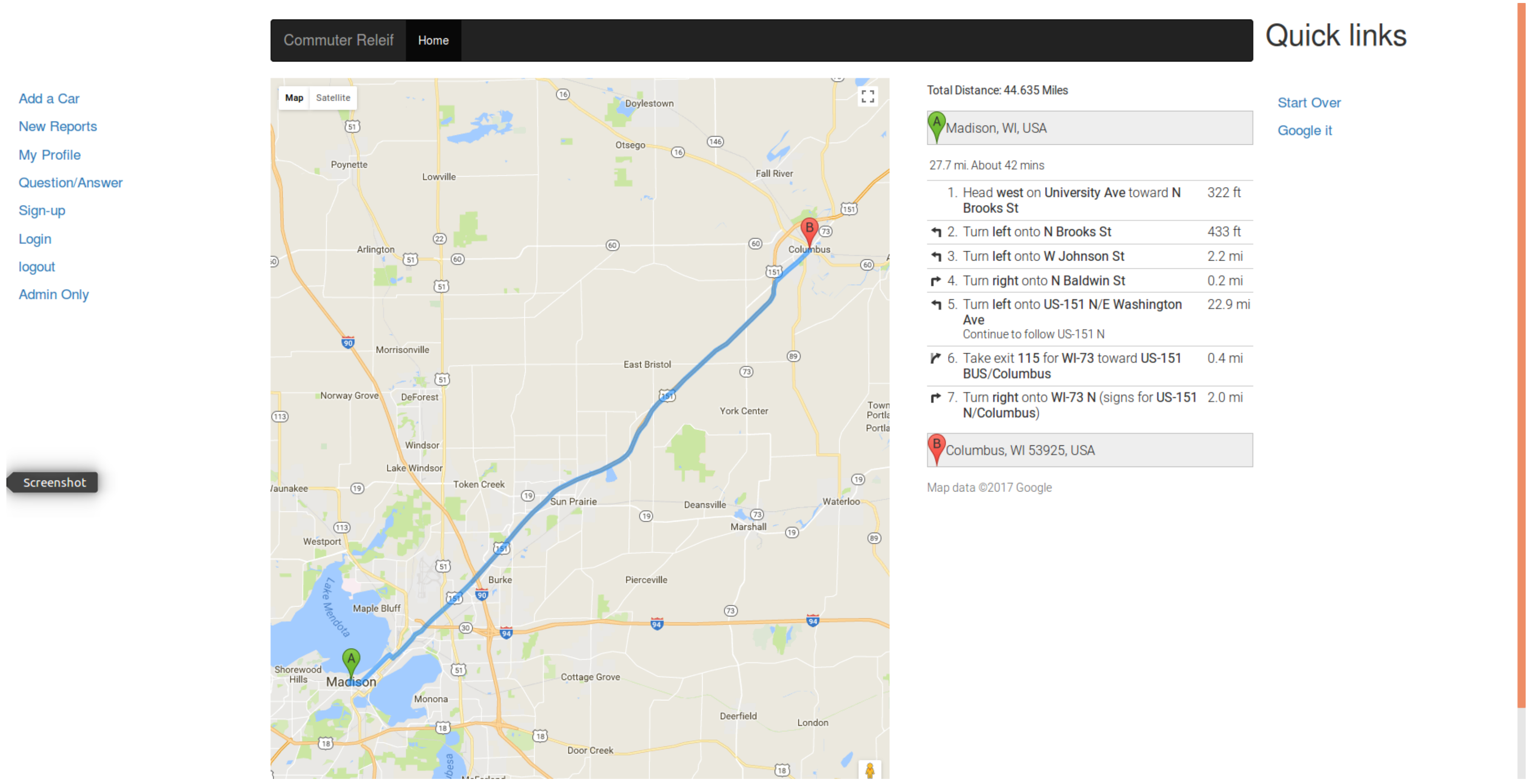
```
//double cityMPG = Double.parseDouble(addRoute.getMpgCity());
//double highMPG = Double.parseDouble(addRoute.getMpgHigh());
if (addRoute.getMpgCity() != null && addRoute.getMpgHigh() != null) {
    log.info("city miles:" + addRoute.getNumberOfCityMiles());
    log.info("city mpg: " + Double.parseDouble(addRoute.getMpgCity()));
    log.info("highwayMiles: " + addRoute.getNumberOfHighwayMiles());
    log.info("highway mpg: " + Double.parseDouble(addRoute.getMpgHigh()));
    log.info("gas price: "  + addRoute.getGasPrice());
    double totalMoney = (((addRoute.getNumberOfCityMiles() / Double.parseDouble(addRoute.getMpgCi
    log.info(totalMoney);
    decimalFormat.format(totalMoney);
    log.info(totalMoney);
    String totalMoneyString = String.valueOf(totalMoney);
    addRoute.setTotal(totalMoneyString);
```

My API Experience

It took forever to figure out after messing with the code I wrote from the API site for this that my bootstrap was messing up the map and wouldn't allow it to show up.

And as you can see I had to pixel push this entire page because my bootstrap kept screwing it up so I tried my best to make it look like the bootstrap I used.

I am also aware that this API consumes javascript, but I really liked this if I could only get it to take the info and make a report out of it. (coming soon)



Total Distance: 44.635 Miles

Madison, WI, USA

27.7 mi. About 42 mins

1.	Head **west** on **University Ave** toward **N Brooks St**	322 ft
2.	Turn **left** onto **N Brooks St**	433 ft
3.	Turn **left** onto **W Johnson St**	2.2 mi
4.	Turn **right** onto **N Baldwin St**	0.2 mi
5.	Turn **left** onto **US-151 N/E Washington Ave** Continue to follow US-151 N	22.9 mi
6.	Take exit **115** for **WI-73** toward **US-151 BUS/Columbus**	0.4 mi
7.	Turn **right** onto **WI-73 N** (signs for **US-151 N/Columbus**)	2.0 mi

Columbus, WI 53925, USA

Map data ©2017 Google

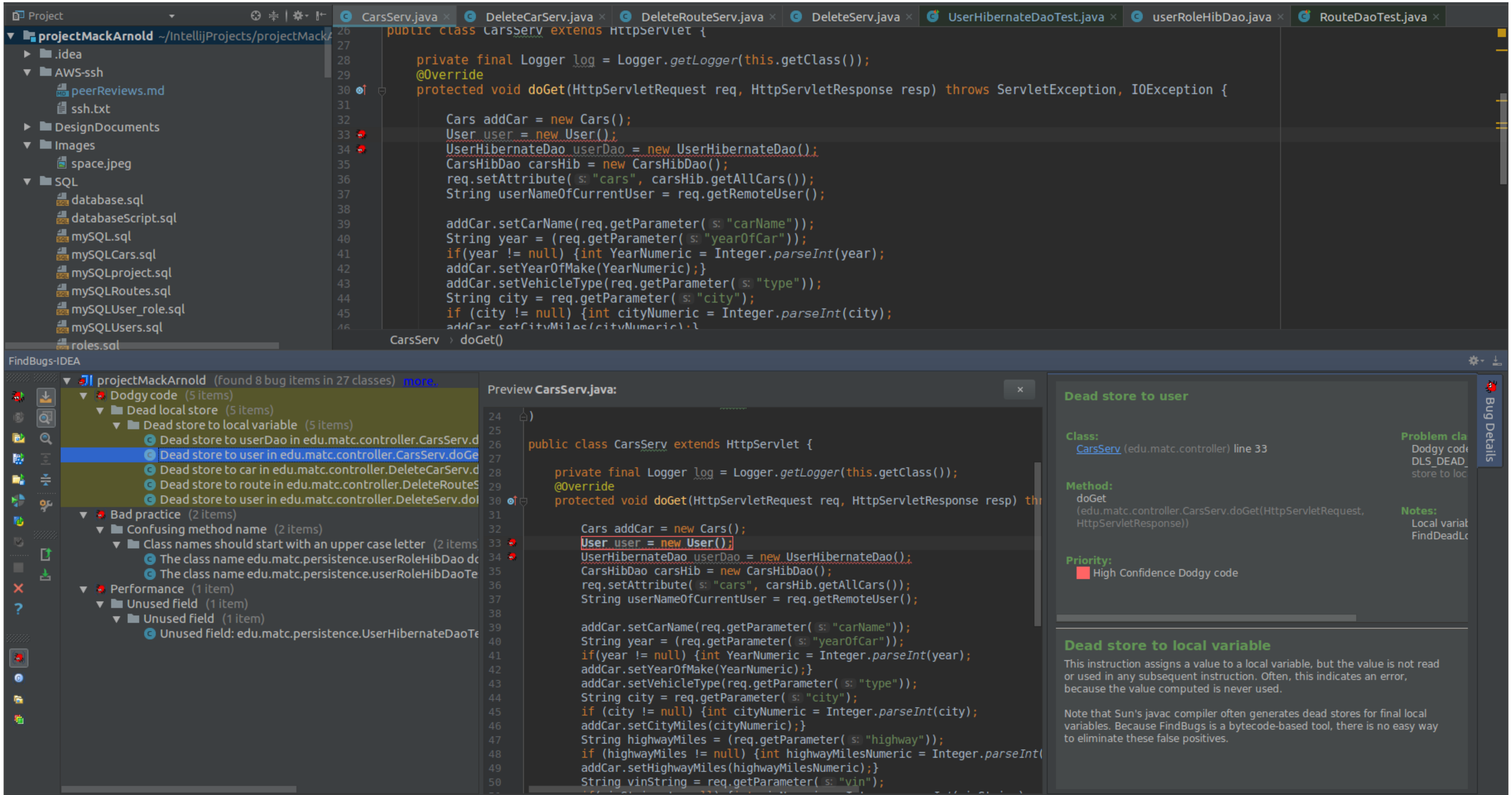
Lessons learned

- Using the logger is a great use, but every once and a while clear out the myAppLog. It gets huge and really confusing if your logging a ton of stuff over and over.
- Constantly uploading to AWS is a good idea. I went a month without touching the AWS account, and when I deployed some things worked but not everything. So deploying consistently is a good idea.
- Do the @before or import.sql for testing is a great idea. It saves time and makes less of a hassle in the future.
- Also make the tests right away so when you get to implementing and using hibernate, make sure tests pass so you know your methods work.

My individual research topic

- I implemented a code quality plugin to ensure that my code isn’t as bad as it was
- The next couple slides will show that there weren’t that many errors.
    - Just some unused objects I created, and some names that it didn’t understand that I will explain when I get to it.

This is an example of the plugin finding an object I made and never used, which intellij sees as well, but this helped me delete redundant code.



```
public class CarsServ extends HttpServlet {

    private final Logger log = Logger.getLogger(this.getClass());
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        Cars addCar = new Cars();
        User user = new User();
        UserHibernateDao userDao = new UserHibernateDao();
        CarsHibDao carsHib = new CarsHibDao();
        req.setAttribute( s: "cars", carsHib.getAllCars());
        String userNameOfCurrentUser = req.getRemoteUser();

        addCar.setCarName(req.getParameter( s: "carName"));
        String year = (req.getParameter( s: "yearOfCar"));
        if(year != null) {int YearNumeric = Integer.parseInt(year);
        addCar.setYearOfMake(YearNumeric);}
        addCar.setVehicleType(req.getParameter( s: "type"));
        String city = req.getParameter( s: "city");
        if (city != null) {int cityNumeric = Integer.parseInt(city);
        addCar.setCityMiles(cityNumeric);}
```

```
projectMackArnold  (found 8 bug items in 27 classes)  more..
  Dodgy code  (5 items)
    Dead local store  (5 items)
      Dead store to local variable  (5 items)
        Dead store to userDao in edu.matc.controller.CarsServ.d
        Dead store to user in edu.matc.controller.CarsServ.doGe
        Dead store to car in edu.matc.controller.DeleteCarServ.d
        Dead store to route in edu.matc.controller.DeleteRouteS
        Dead store to user in edu.matc.controller.DeleteServ.doI
  Bad practice  (2 items)
    Confusing method name  (2 items)
      Class names should start with an upper case letter  (2 items
        The class name edu.matc.persistence.userRoleHibDao dc
        The class name edu.matc.persistence.userRoleHibDaoTe
  Performance  (1 item)
    Unused field  (1 item)
      Unused field  (1 item)
        Unused field: edu.matc.persistence.UserHibernateDaoTe
```

```
)

public class CarsServ extends HttpServlet {

    private final Logger log = Logger.getLogger(this.getClass());
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thr

        Cars addCar = new Cars();
        User user = new User();
        UserHibernateDao userDao = new UserHibernateDao();
        CarsHibDao carsHib = new CarsHibDao();
        req.setAttribute( s: "cars", carsHib.getAllCars());
        String userNameOfCurrentUser = req.getRemoteUser();

        addCar.setCarName(req.getParameter( s: "carName"));
        String year = (req.getParameter( s: "yearOfCar"));
        if(year != null) {int YearNumeric = Integer.parseInt(year);
        addCar.setYearOfMake(YearNumeric);}
        addCar.setVehicleType(req.getParameter( s: "type"));
        String city = req.getParameter( s: "city");
        if (city != null) {int cityNumeric = Integer.parseInt(city);
        addCar.setCityMiles(cityNumeric);}
        String highwayMiles = (req.getParameter( s: "highway"));
        if (highwayMiles != null) {int highwayMilesNumeric = Integer.parseInt(
        addCar.setHighwayMiles(highwayMilesNumeric);}
        String vinString = req.getParameter( s: "vin");
```

Dead store to user

Class: CarsServ (edu.matc.controller) line 33

Method: doGet (edu.matc.controller.CarsServ.doGet(HttpServletRequest, HttpServletResponse))

Priority: 🟥 High Confidence Dodgy code

Problem cla Dodgy code DLS_DEAD_ store to loc

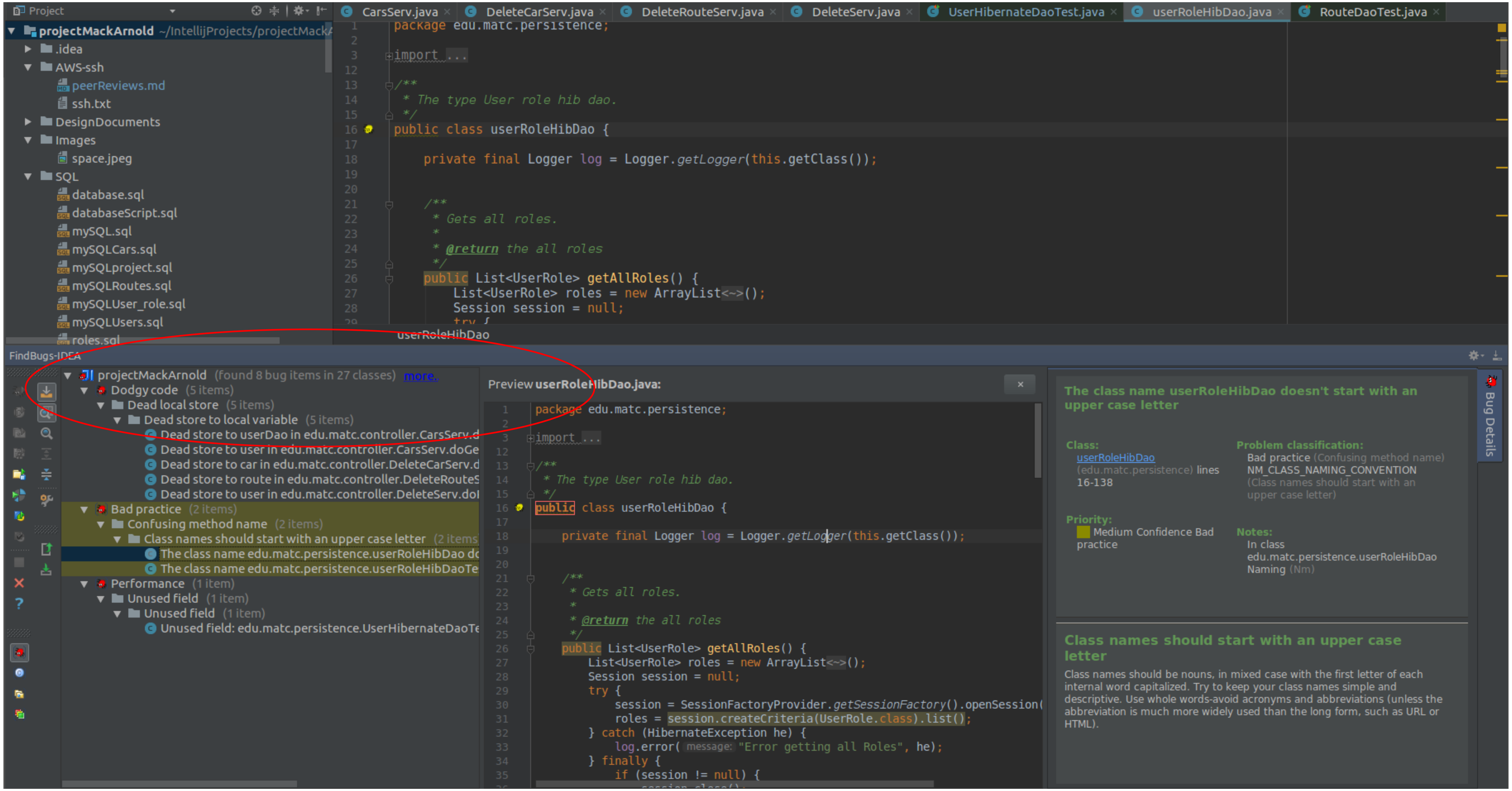
Notes: Local variab FindDeadLc

Dead store to local variable

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because FindBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

This is to show you that when you run the bug finder, it will say how many bugs in how many classes it analyzed.



Project
- projectMackArnold ~/IntellijProjects/projectMackA
  - .idea
  - AWS-ssh
    - peerReviews.md
    - ssh.txt
  - DesignDocuments
  - Images
    - space.jpeg
  - SQL
    - database.sql
    - databaseScript.sql
    - mySQL.sql
    - mySQLCars.sql
    - mySQLproject.sql
    - mySQLRoutes.sql
    - mySQLUser_role.sql
    - mySQLUsers.sql
    - roles.sql

CarsServ.java | DeleteCarServ.java | DeleteRouteServ.java | DeleteServ.java | UserHibernateDaoTest.java | userRoleHibDao.java | RouteDaoTest.java

```java
package edu.matc.persistence;

import ...

/**
 * The type User role hib dao.
 */
public class userRoleHibDao {

    private final Logger log = Logger.getLogger(this.getClass());


    /**
     * Gets all roles.
     *
     * @return the all roles
     */
    public List<UserRole> getAllRoles() {
        List<UserRole> roles = new ArrayList<~>();
        Session session = null;
        try {
```

userRoleHibDao

FindBugs-IDEA

- projectMackArnold (found 8 bug items in 27 classes) more..
  - Dodgy code (5 items)
    - Dead local store (5 items)
      - Dead store to local variable (5 items)
        - Dead store to userDao in edu.matc.controller.CarsServ.d
        - Dead store to user in edu.matc.controller.CarsServ.doGe
        - Dead store to car in edu.matc.controller.DeleteCarServ.d
        - Dead store to route in edu.matc.controller.DeleteRouteS
        - Dead store to user in edu.matc.controller.DeleteServ.doI
  - Bad practice (2 items)
    - Confusing method name (2 items)
      - Class names should start with an upper case letter (2 items
        - The class name edu.matc.persistence.userRoleHibDao do
        - The class name edu.matc.persistence.userRoleHibDaoTe
  - Performance (1 item)
    - Unused field (1 item)
      - Unused field (1 item)
        - Unused field: edu.matc.persistence.UserHibernateDaoTe

Preview **userRoleHibDao.java:**

```java
package edu.matc.persistence;

import ...

/**
 * The type User role hib dao.
 */
public class userRoleHibDao {

    private final Logger log = Logger.getLogger(this.getClass());


    /**
     * Gets all roles.
     *
     * @return the all roles
     */
    public List<UserRole> getAllRoles() {
        List<UserRole> roles = new ArrayList<~>();
        Session session = null;
        try {
            session = SessionFactoryProvider.getSessionFactory().openSession(
            roles = session.createCriteria(UserRole.class).list();
        } catch (HibernateException he) {
            log.error( message: "Error getting all Roles", he);
        } finally {
            if (session != null) {
```

The class name userRoleHibDao doesn't start with an upper case letter

Class:
userRoleHibDao
(edu.matc.persistence) lines 16-138

Problem classification:
Bad practice (Confusing method name)
NM_CLASS_NAMING_CONVENTION
(Class names should start with an upper case letter)

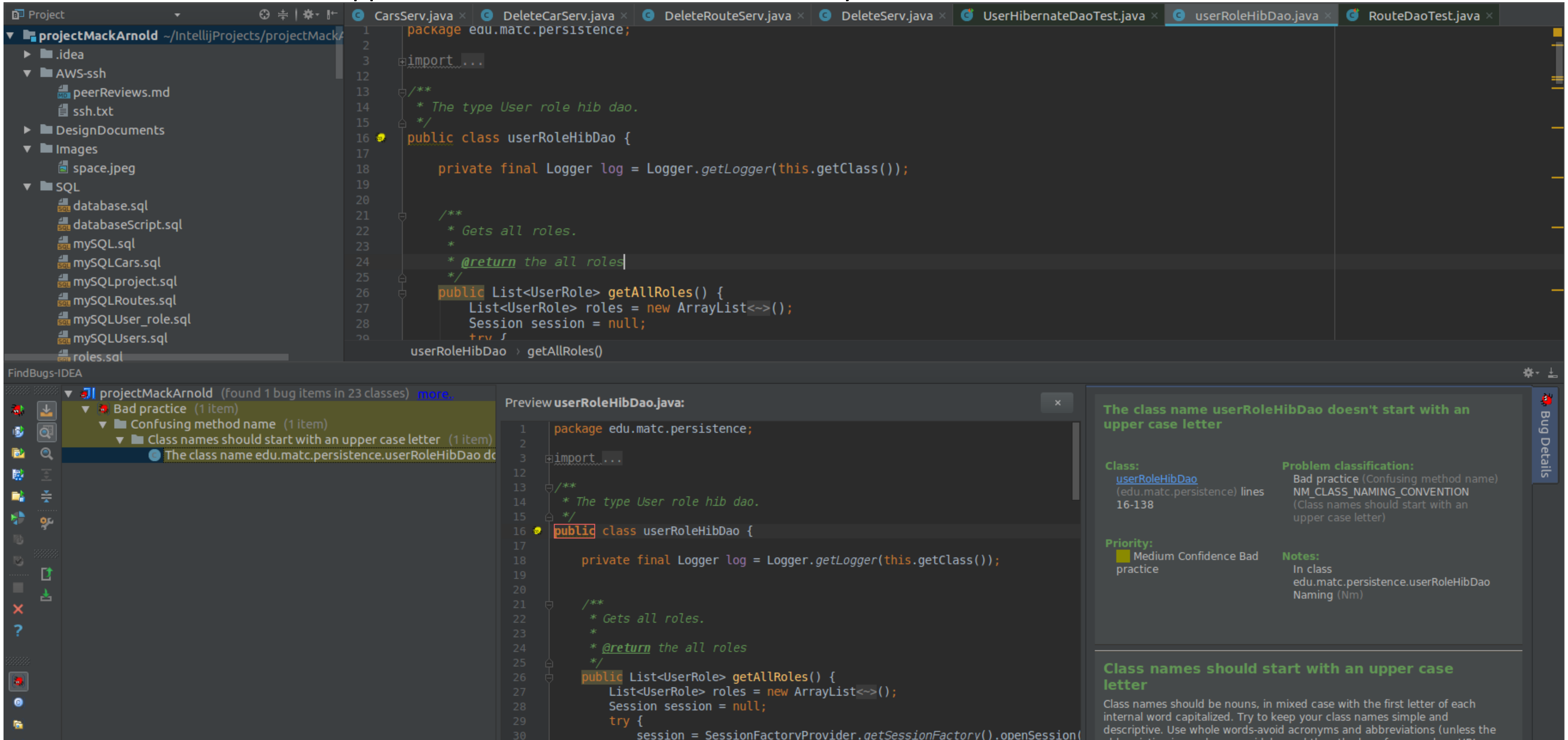
Priority:
🟨 Medium Confidence Bad practice

Notes:
In class
edu.matc.persistence.userRoleHibDao
Naming (Nm)

Class names should start with an upper case letter

Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).

And this is What I found that Paula pointed out to me that not everything it will tell you needs to be changed. For this example its complaining about my real weird name for my classes and in this case I don't have to change anything, its just the way I code and read it. It also complains when it sees that I don't have an uppercase class name which can be easily overlooked.



Project
projectMackArnold ~/IntellijProjects/projectMackA
.idea
AWS-ssh
peerReviews.md
ssh.txt
DesignDocuments
Images
space.jpeg
SQL
database.sql
databaseScript.sql
mySQL.sql
mySQLCars.sql
mySQLproject.sql
mySQLRoutes.sql
mySQLUser_role.sql
mySQLUsers.sql
roles.sql

CarsServ.java | DeleteCarServ.java | DeleteRouteServ.java | DeleteServ.java | UserHibernateDaoTest.java | userRoleHibDao.java | RouteDaoTest.java

```java
package edu.matc.persistence;

import ...

/**
 * The type User role hib dao.
 */
public class userRoleHibDao {

    private final Logger log = Logger.getLogger(this.getClass());


    /**
     * Gets all roles.
     *
     * @return the all roles
     */
    public List<UserRole> getAllRoles() {
        List<UserRole> roles = new ArrayList<~>();
        Session session = null;
        try {
```

userRoleHibDao › getAllRoles()

FindBugs-IDEA

projectMackArnold (found 1 bug items in 23 classes) more..
Bad practice (1 item)
Confusing method name (1 item)
Class names should start with an upper case letter (1 item)
The class name edu.matc.persistence.userRoleHibDao dc

Preview userRoleHibDao.java:

```java
package edu.matc.persistence;

import ...

/**
 * The type User role hib dao.
 */
public class userRoleHibDao {

    private final Logger log = Logger.getLogger(this.getClass());


    /**
     * Gets all roles.
     *
     * @return the all roles
     */
    public List<UserRole> getAllRoles() {
        List<UserRole> roles = new ArrayList<~>();
        Session session = null;
        try {
            session = SessionFactoryProvider.getSessionFactory().openSession(
```

The class name userRoleHibDao doesn't start with an upper case letter

Class:
userRoleHibDao
(edu.matc.persistence) lines 16-138

Problem classification:
Bad practice (Confusing method name)
NM_CLASS_NAMING_CONVENTION
(Class names should start with an upper case letter)

Priority:
🟨 Medium Confidence Bad practice

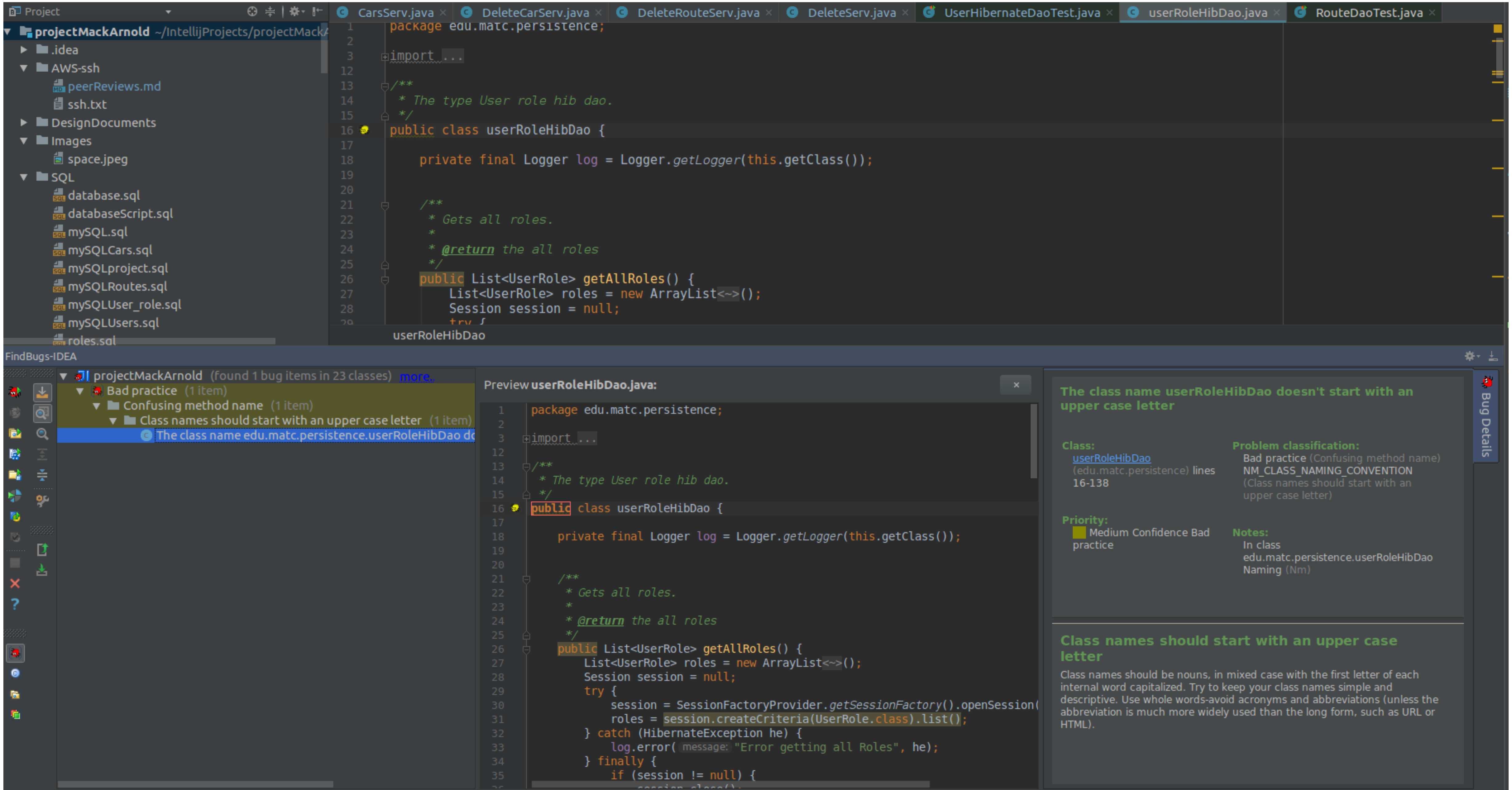
Notes:
In class
edu.matc.persistence.userRoleHibDao
Naming (Nm)

Class names should start with an upper case letter

Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the

Bug Details

This is another example from the last slide where the class didn't start with a capital letter, but when you hover over the name it says couldn't comprehend the name, but I couldn't hover and screenshot at the same time.



```java
package edu.matc.persistence;

import ...

/**
 * The type User role hib dao.
 */
public class userRoleHibDao {

    private final Logger log = Logger.getLogger(this.getClass());


    /**
     * Gets all roles.
     *
     * @return the all roles
     */
    public List<UserRole> getAllRoles() {
        List<UserRole> roles = new ArrayList<~>();
        Session session = null;
        try {
```

```
projectMackArnold (found 1 bug items in 23 classes) more...
    Bad practice (1 item)
        Confusing method name (1 item)
            Class names should start with an upper case letter (1 item)
                The class name edu.matc.persistence.userRoleHibDao dc
```

Preview userRoleHibDao.java:

```java
package edu.matc.persistence;

import ...

/**
 * The type User role hib dao.
 */
public class userRoleHibDao {

    private final Logger log = Logger.getLogger(this.getClass());


    /**
     * Gets all roles.
     *
     * @return the all roles
     */
    public List<UserRole> getAllRoles() {
        List<UserRole> roles = new ArrayList<~>();
        Session session = null;
        try {
            session = SessionFactoryProvider.getSessionFactory().openSession(
            roles = session.createCriteria(UserRole.class).list();
        } catch (HibernateException he) {
            log.error( message: "Error getting all Roles", he);
        } finally {
            if (session != null) {
```

The class name userRoleHibDao doesn't start with an upper case letter

Class:
userRoleHibDao
(edu.matc.persistence) lines
16-138

Problem classification:
Bad practice (Confusing method name)
NM_CLASS_NAMING_CONVENTION
(Class names should start with an upper case letter)

Priority:
🟨 Medium Confidence Bad practice

Notes:
In class
edu.matc.persistence.userRoleHibDao
Naming (Nm)

Class names should start with an upper case letter

Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).

This is the end of my final Individual presentation for Enterprise Java.

I will not be continuing this project….