EAPLI
Teórico-Prática

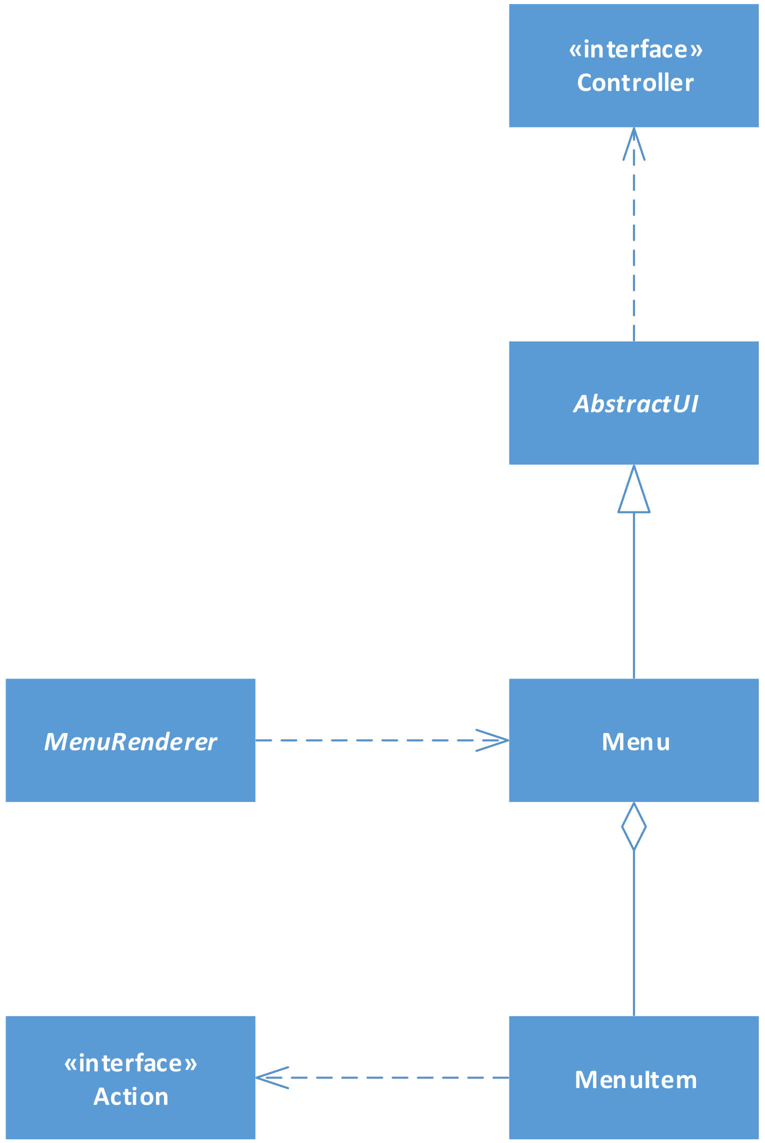
EAPLI Framework

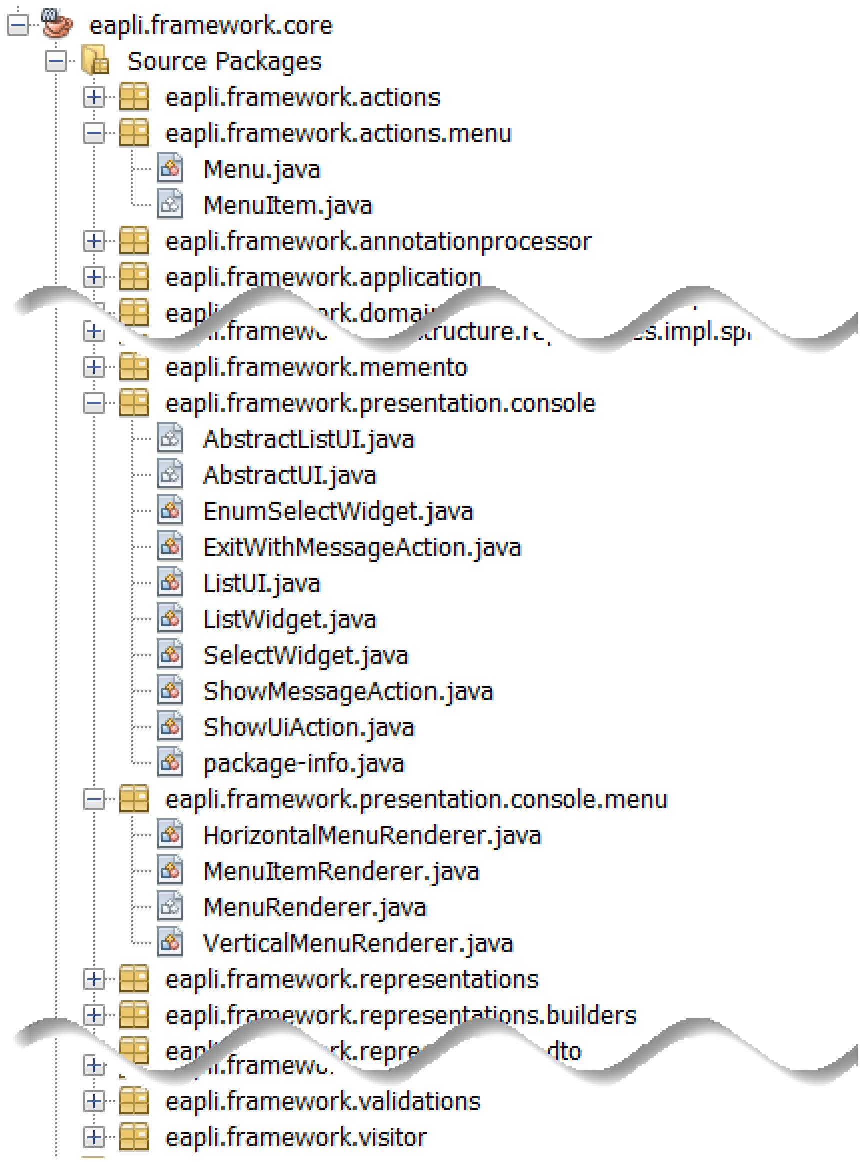
EAPLI framework

Core

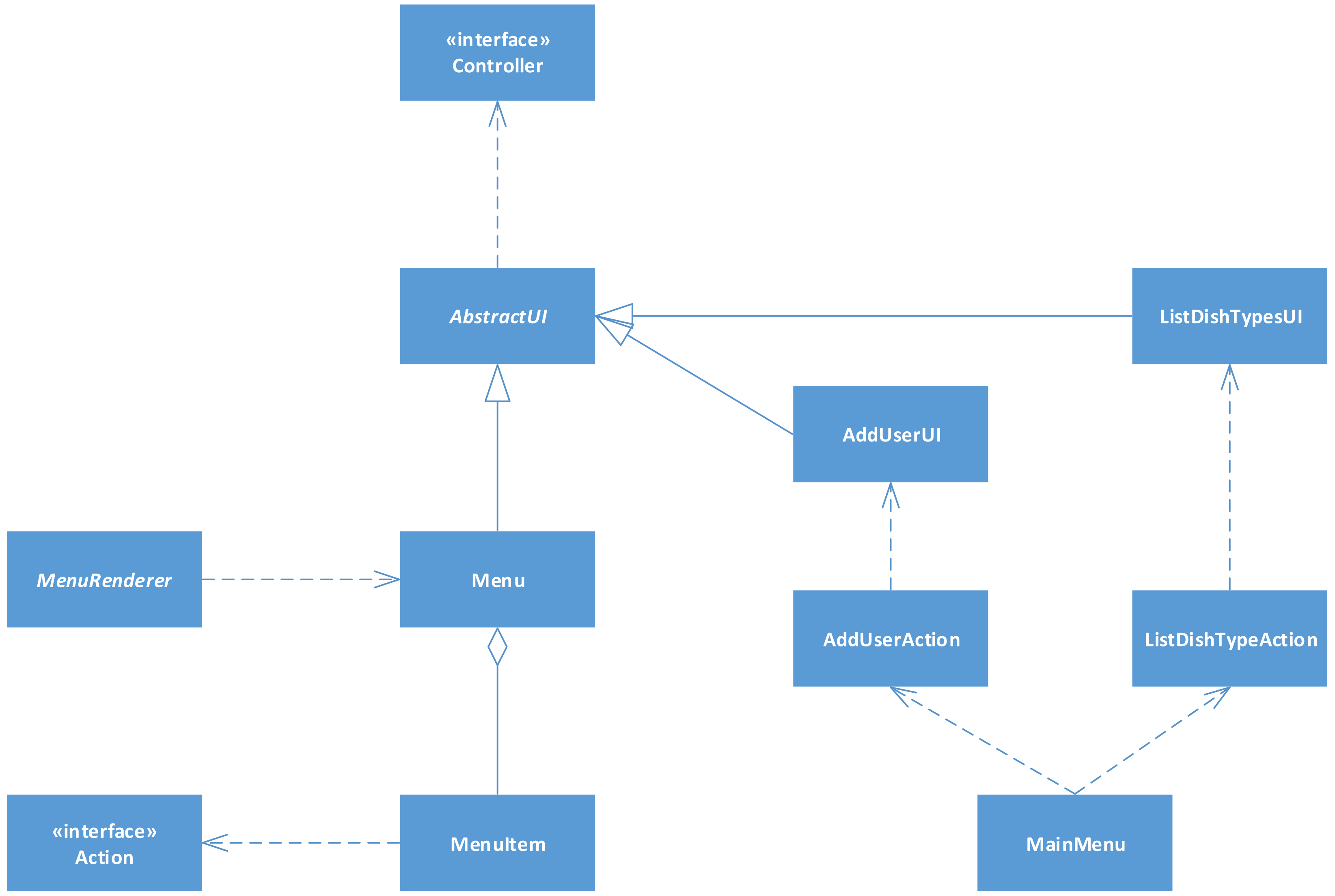
Presentation

Presentation

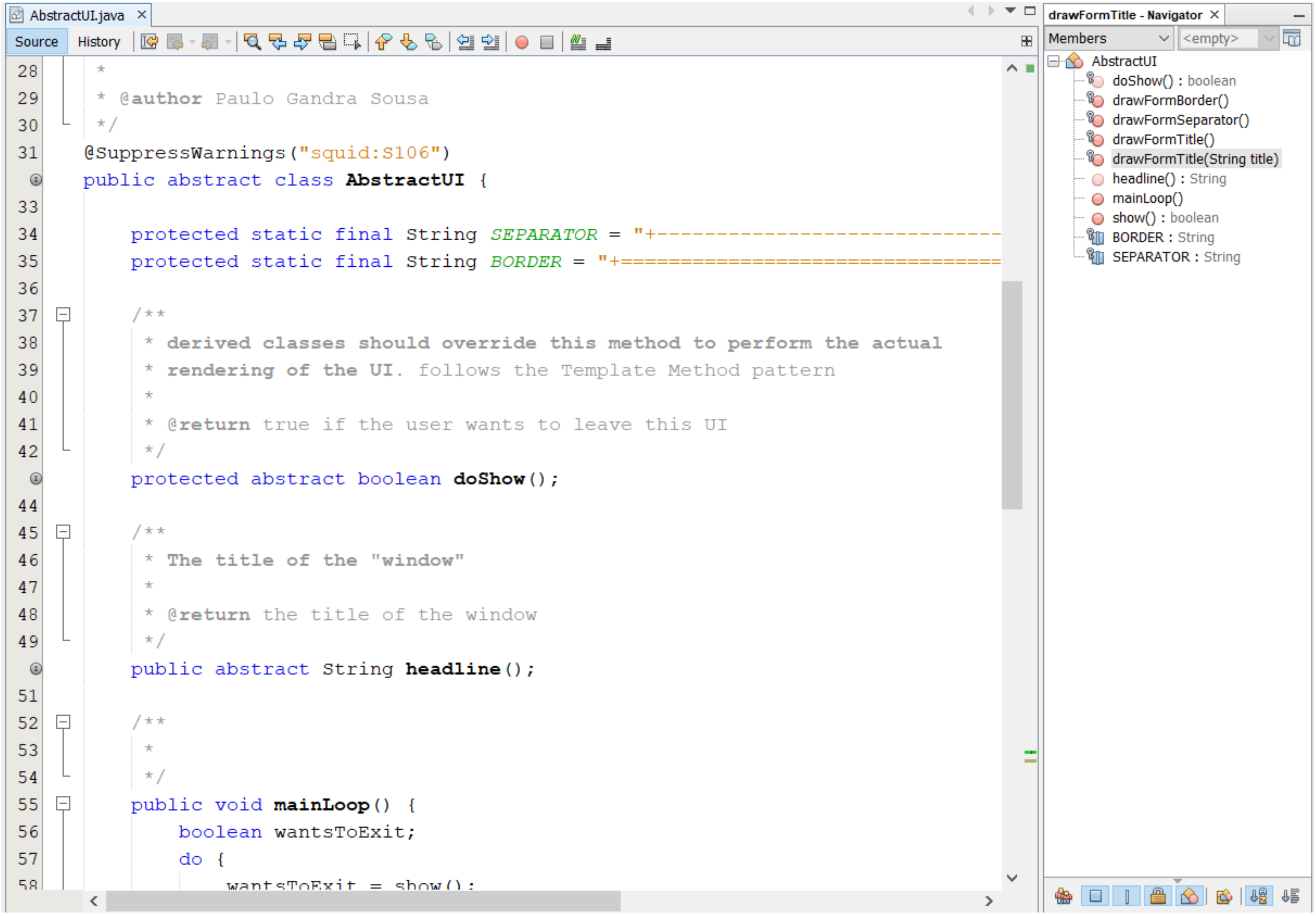



```
eapli.framework.core
  Source Packages
    eapli.framework.actions
    eapli.framework.actions.menu
      Menu.java
      MenuItem.java
    eapli.framework.annotationprocessor
    eapli.framework.application
    eapl    rk.domai
    .i.framew    .ructure.r    s.impl.sp
    eapli.framework.memento
    eapli.framework.presentation.console
      AbstractListUI.java
      AbstractUI.java
      EnumSelectWidget.java
      ExitWithMessageAction.java
      ListUI.java
      ListWidget.java
      SelectWidget.java
      ShowMessageAction.java
      ShowUiAction.java
      package-info.java
    eapli.framework.presentation.console.menu
      HorizontalMenuRenderer.java
      MenuItemRenderer.java
      MenuRenderer.java
      VerticalMenuRenderer.java
    eapli.framework.representations
    eapli.framework.representations.builders
    eap    k.repre    dto
    .i.framewo.
    eapli.framework.validations
    eapli.framework.visitor
```

Presentation



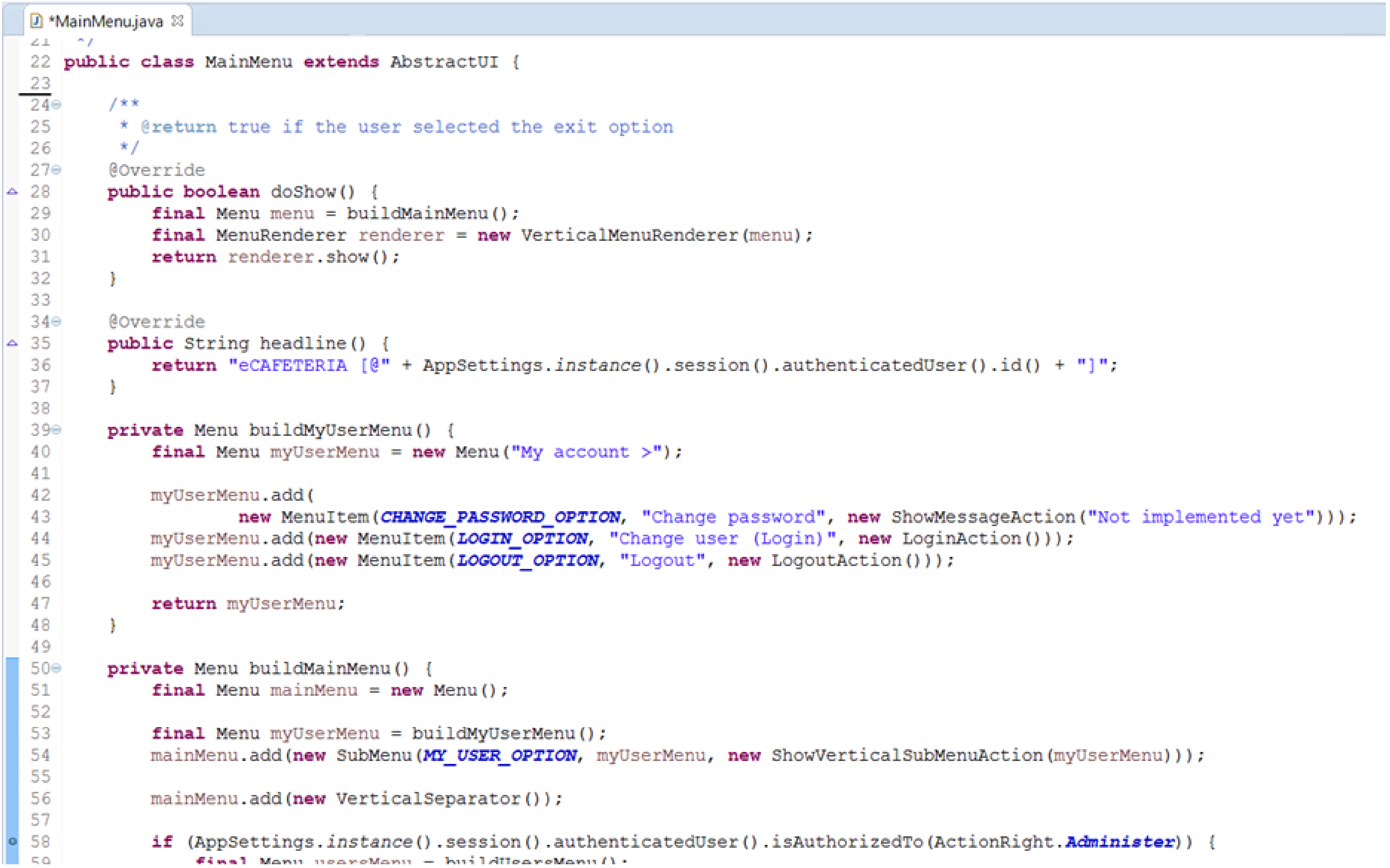
AbstractUI.java



```java
 *
 * @author Paulo Gandra Sousa
 */
@SuppressWarnings("squid:S106")
public abstract class AbstractUI {

    protected static final String SEPARATOR = "+----------------------------
    protected static final String BORDER = "+================================

    /**
     * derived classes should override this method to perform the actual
     * rendering of the UI. follows the Template Method pattern
     *
     * @return true if the user wants to leave this UI
     */
    protected abstract boolean doShow();

    /**
     * The title of the "window"
     *
     * @return the title of the window
     */
    public abstract String headline();

    /**
     *
     */
    public void mainLoop() {
        boolean wantsToExit;
        do {
            wantsToExit = show();
```

```
AbstractUI
    doShow() : boolean
    drawFormBorder()
    drawFormSeparator()
    drawFormTitle()
    drawFormTitle(String title)
    headline() : String
    mainLoop()
    show() : boolean
    BORDER : String
    SEPARATOR : String
```

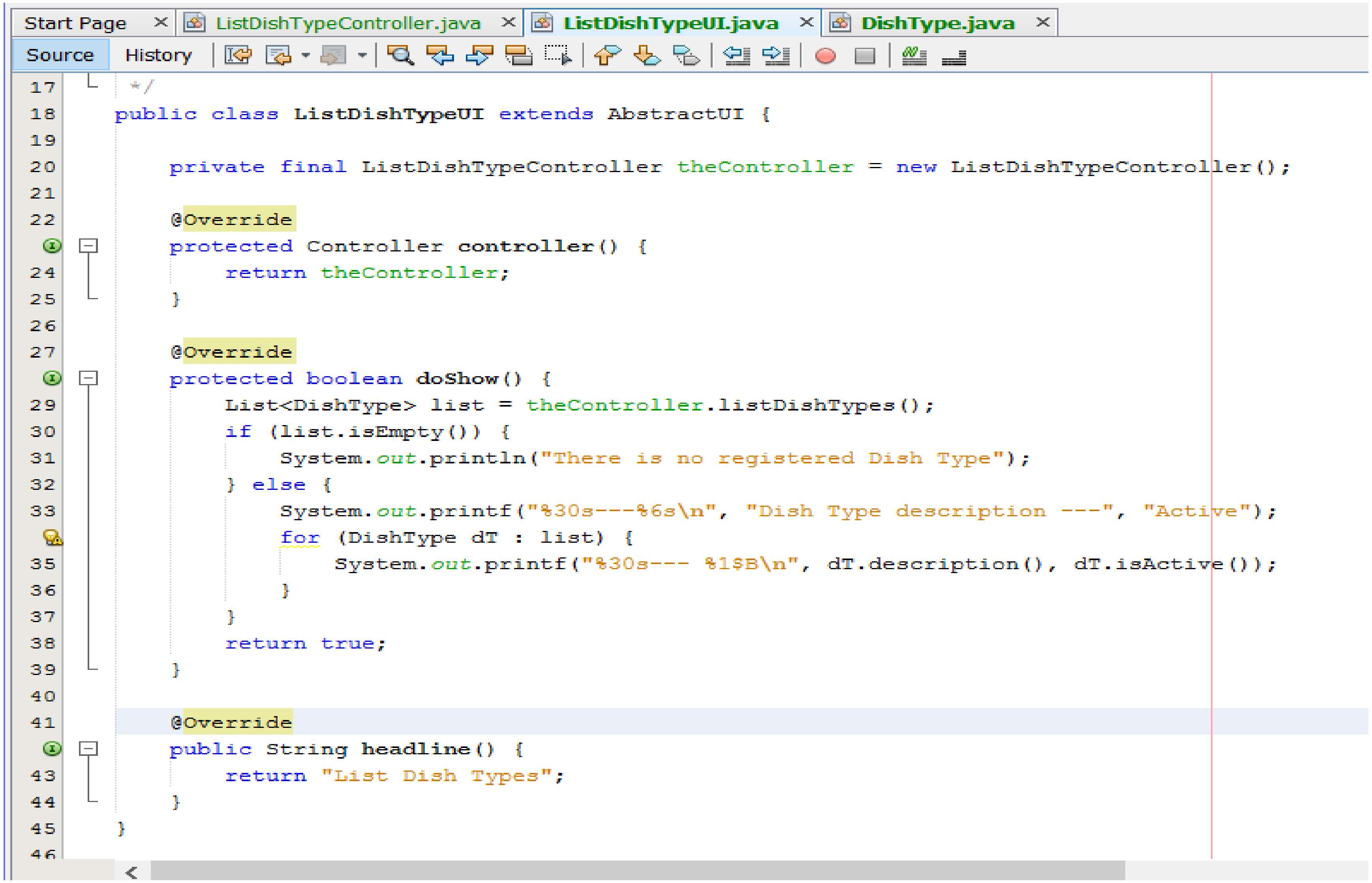
Example



*MainMenu.java

```
public class MainMenu extends AbstractUI {

    /**
     * @return true if the user selected the exit option
     */
    @Override
    public boolean doShow() {
        final Menu menu = buildMainMenu();
        final MenuRenderer renderer = new VerticalMenuRenderer(menu);
        return renderer.show();
    }

    @Override
    public String headline() {
        return "eCAFETERIA [@" + AppSettings.instance().session().authenticatedUser().id() + "]";
    }

    private Menu buildMyUserMenu() {
        final Menu myUserMenu = new Menu("My account >");

        myUserMenu.add(
                new MenuItem(CHANGE_PASSWORD_OPTION, "Change password", new ShowMessageAction("Not implemented yet")));
        myUserMenu.add(new MenuItem(LOGIN_OPTION, "Change user (Login)", new LoginAction()));
        myUserMenu.add(new MenuItem(LOGOUT_OPTION, "Logout", new LogoutAction()));

        return myUserMenu;
    }

    private Menu buildMainMenu() {
        final Menu mainMenu = new Menu();

        final Menu myUserMenu = buildMyUserMenu();
        mainMenu.add(new SubMenu(MY_USER_OPTION, myUserMenu, new ShowVerticalSubMenuAction(myUserMenu)));

        mainMenu.add(new VerticalSeparator());

        if (AppSettings.instance().session().authenticatedUser().isAuthorizedTo(ActionRight.Administer)) {
```

Example

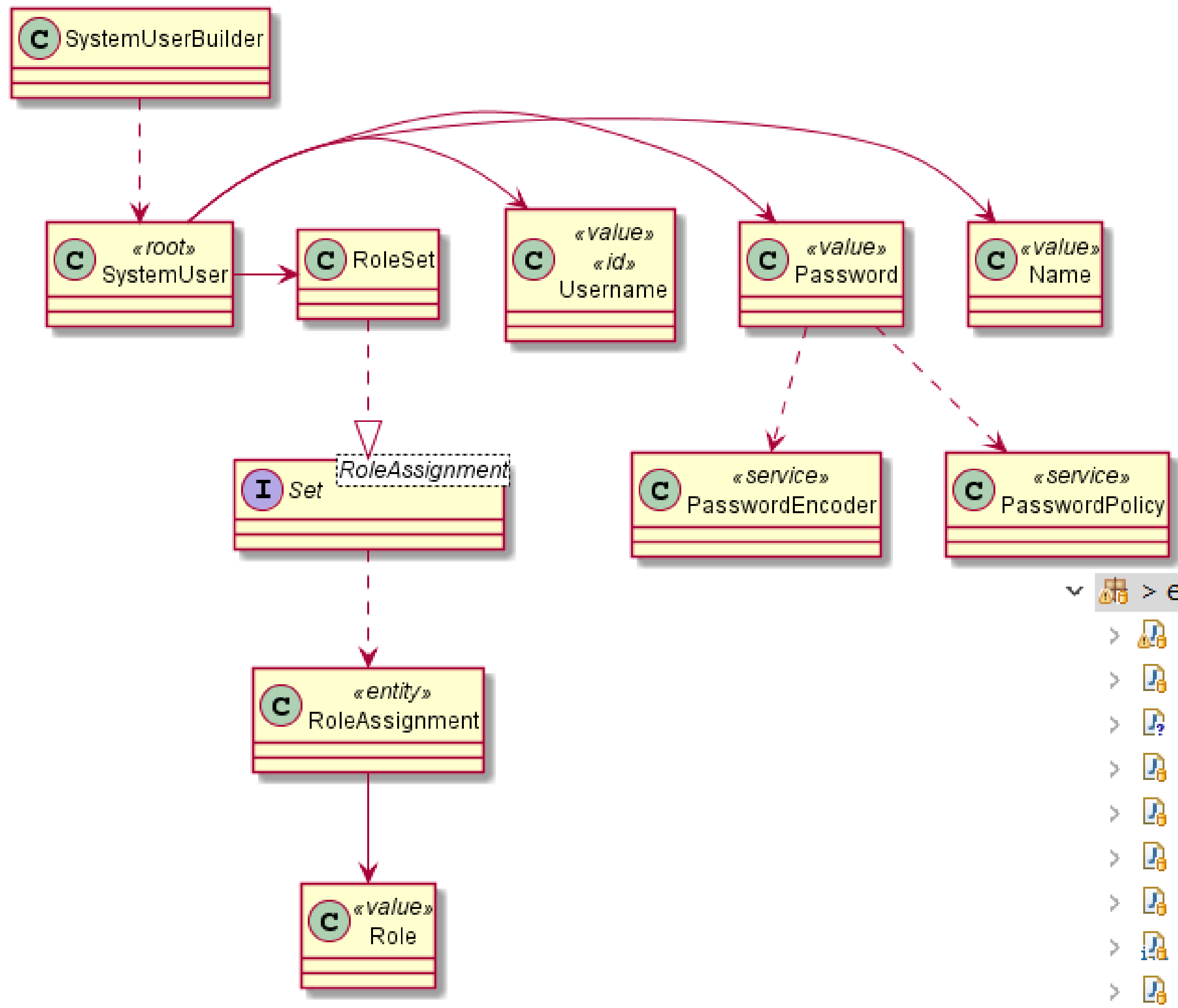


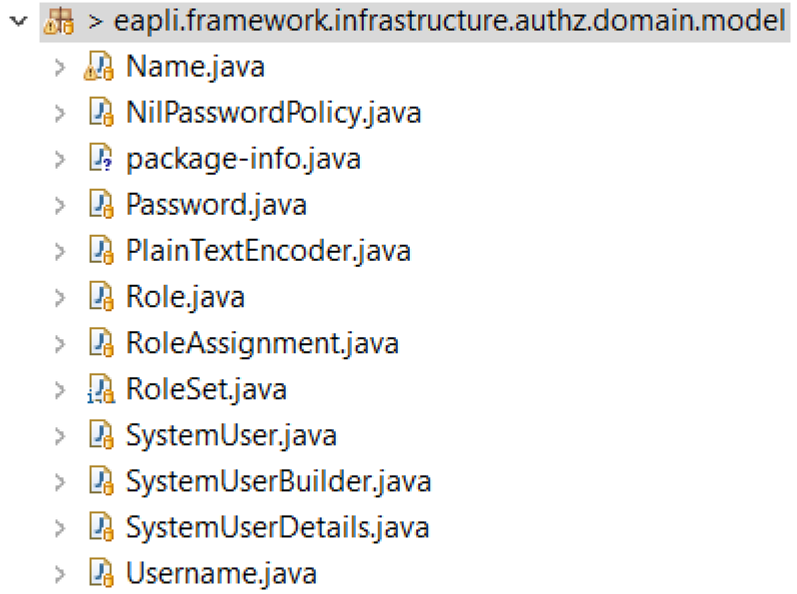
```java
    */
public class ListDishTypeUI extends AbstractUI {

    private final ListDishTypeController theController = new ListDishTypeController();

    @Override
    protected Controller controller() {
        return theController;
    }

    @Override
    protected boolean doShow() {
        List<DishType> list = theController.listDishTypes();
        if (list.isEmpty()) {
            System.out.println("There is no registered Dish Type");
        } else {
            System.out.printf("%30s---%6s\n", "Dish Type description ---", "Active");
            for (DishType dT : list) {
                System.out.printf("%30s--- %1$B\n", dT.description(), dT.isActive());
            }
        }
        return true;
    }

    @Override
    public String headline() {
        return "List Dish Types";
    }
}
```

EAPLI framework

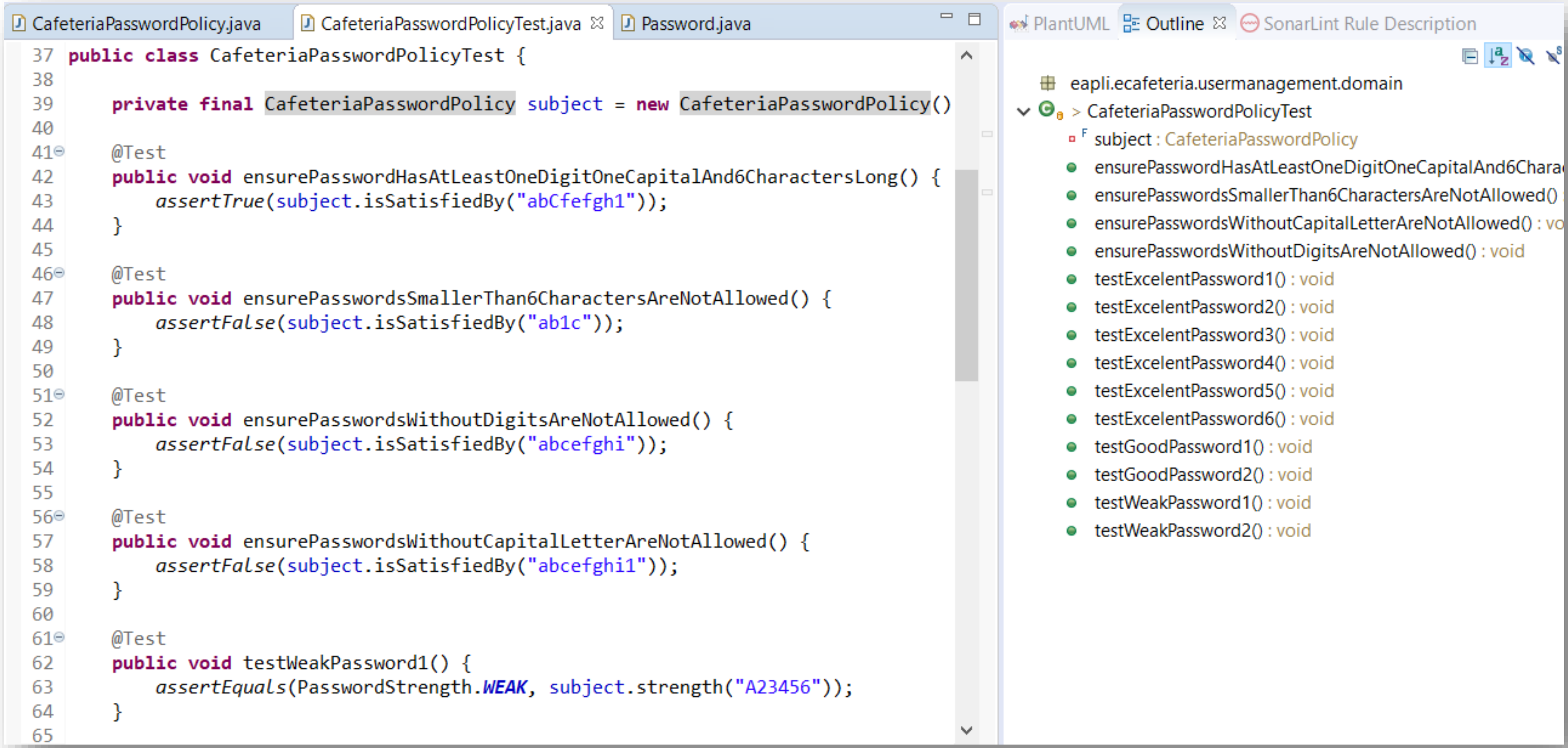
Infrastructure Authz

User model




> eapli.framework.infrastructure.authz.domain.model
Name.java
NilPasswordPolicy.java
package-info.java
Password.java
PlainTextEncoder.java
Role.java
RoleAssignment.java
RoleSet.java
SystemUser.java
SystemUserBuilder.java
SystemUserDetails.java
Username.java

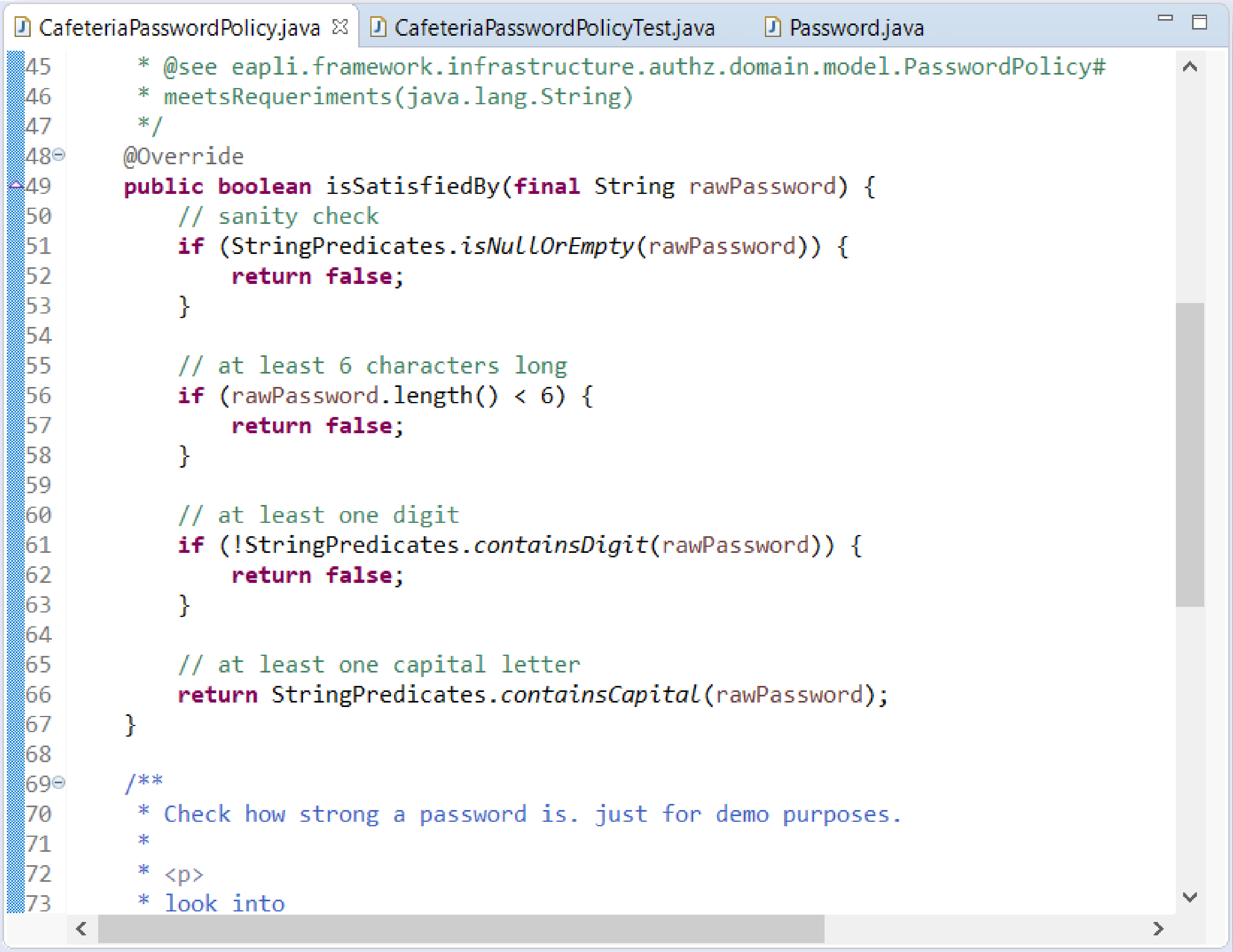
Domain Invariants


CafeteriaPasswordPolicy.java
CafeteriaPasswordPolicyTest.java
Password.java
PlantUML
Outline
SonarLint Rule Description

```java
public class CafeteriaPasswordPolicyTest {

    private final CafeteriaPasswordPolicy subject = new CafeteriaPasswordPolicy()

    @Test
    public void ensurePasswordHasAtLeastOneDigitOneCapitalAnd6CharactersLong() {
        assertTrue(subject.isSatisfiedBy("abCfefgh1"));
    }

    @Test
    public void ensurePasswordsSmallerThan6CharactersAreNotAllowed() {
        assertFalse(subject.isSatisfiedBy("ab1c"));
    }

    @Test
    public void ensurePasswordsWithoutDigitsAreNotAllowed() {
        assertFalse(subject.isSatisfiedBy("abcefghi"));
    }

    @Test
    public void ensurePasswordsWithoutCapitalLetterAreNotAllowed() {
        assertFalse(subject.isSatisfiedBy("abcefghi1"));
    }

    @Test
    public void testWeakPassword1() {
        assertEquals(PasswordStrength.WEAK, subject.strength("A23456"));
    }

```

```
eapli.ecafeteria.usermanagement.domain
  CafeteriaPasswordPolicyTest
    subject : CafeteriaPasswordPolicy
    ensurePasswordHasAtLeastOneDigitOneCapitalAnd6Chara
    ensurePasswordsSmallerThan6CharactersAreNotAllowed()
    ensurePasswordsWithoutCapitalLetterAreNotAllowed() : vo
    ensurePasswordsWithoutDigitsAreNotAllowed() : void
    testExcelentPassword1() : void
    testExcelentPassword2() : void
    testExcelentPassword3() : void
    testExcelentPassword4() : void
    testExcelentPassword5() : void
    testExcelentPassword6() : void
    testGoodPassword1() : void
    testGoodPassword2() : void
    testWeakPassword1() : void
    testWeakPassword2() : void
```

Cafeteria Password Policy

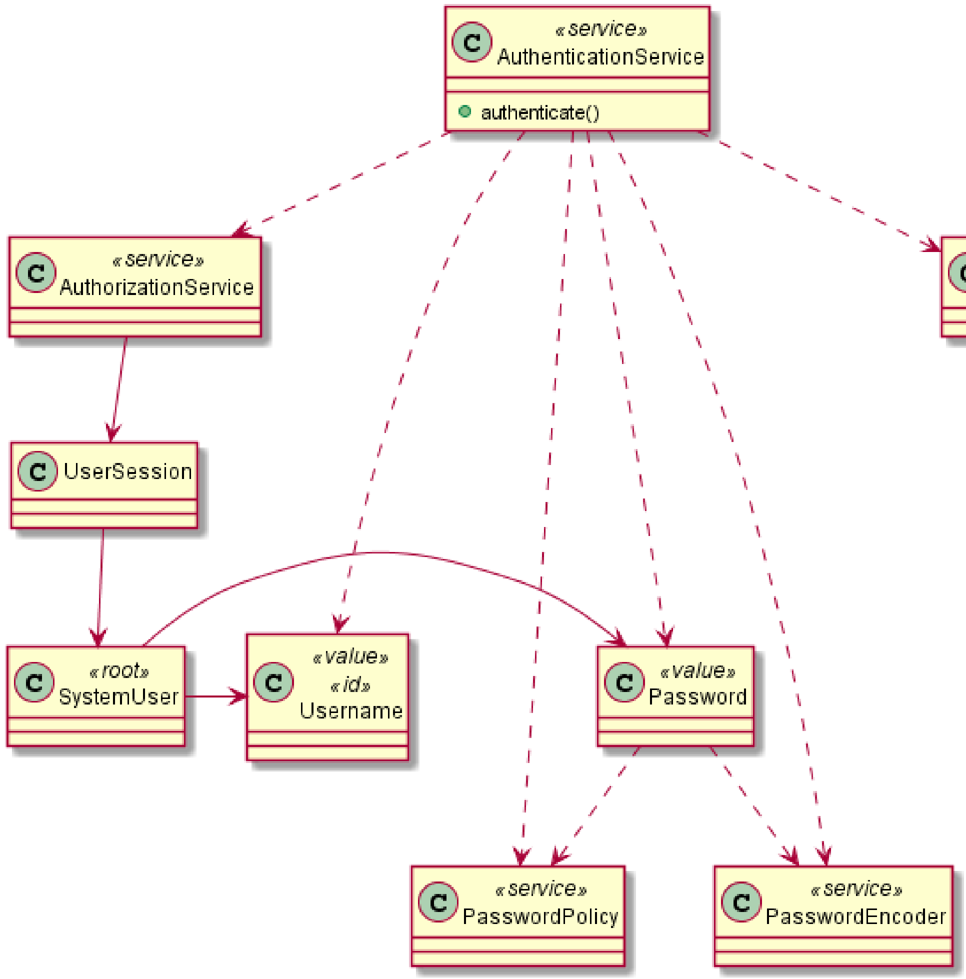


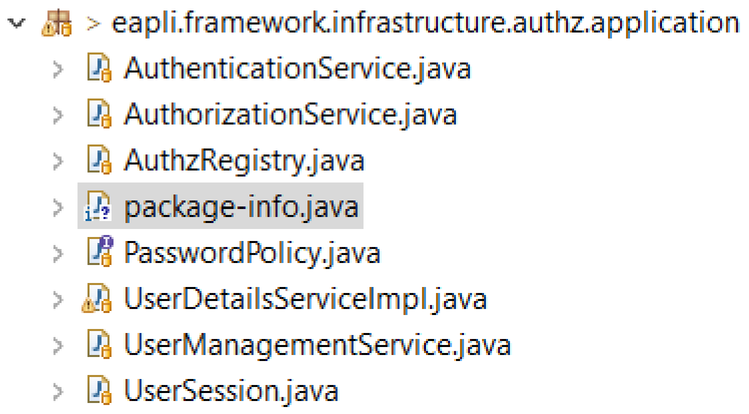
CafeteriaPasswordPolicy.java   CafeteriaPasswordPolicyTest.java   Password.java

```
     * @see eapli.framework.infrastructure.authz.domain.model.PasswordPolicy#
     * meetsRequeriments(java.lang.String)
     */
    @Override
    public boolean isSatisfiedBy(final String rawPassword) {
        // sanity check
        if (StringPredicates.isNullOrEmpty(rawPassword)) {
            return false;
        }

        // at least 6 characters long
        if (rawPassword.length() < 6) {
            return false;
        }

        // at least one digit
        if (!StringPredicates.containsDigit(rawPassword)) {
            return false;
        }

        // at least one capital letter
        return StringPredicates.containsCapital(rawPassword);
    }

    /**
     * Check how strong a password is. just for demo purposes.
     *
     * <p>
     * look into
```

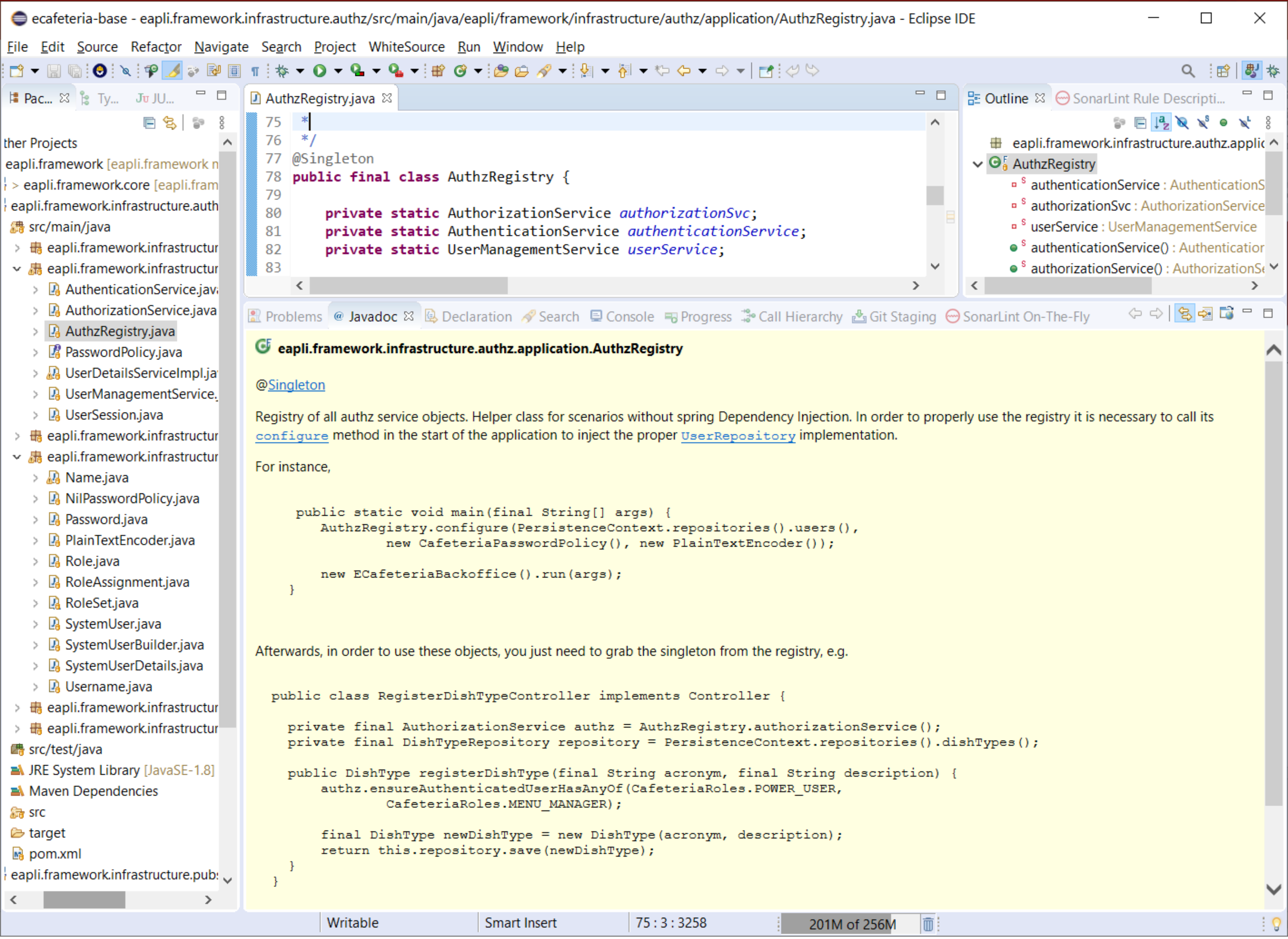
EAPLI framework

Using the authz framework





eapli.framework.infrastructure.authz.application
- AuthenticationService.java
- AuthorizationService.java
- AuthzRegistry.java
- package-info.java
- PasswordPolicy.java
- UserDetailsServiceImpl.java
- UserManagementService.java
- UserSession.java


ecafeteria-base - eapli.framework.infrastructure.authz/src/main/java/eapli/framework/infrastructure/authz/application/AuthzRegistry.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project WhiteSource Run Window Help
Pac... Ty... JU...
ther Projects
eapli.framework [eapli.framework n
> eapli.framework.core [eapli.fram
eapli.framework.infrastructure.auth
src/main/java
eapli.framework.infrastructur
eapli.framework.infrastructur
AuthenticationService.java
AuthorizationService.java
AuthzRegistry.java
PasswordPolicy.java
UserDetailsServiceImpl.ja
UserManagementService.
UserSession.java
eapli.framework.infrastructur
eapli.framework.infrastructur
Name.java
NilPasswordPolicy.java
Password.java
PlainTextEncoder.java
Role.java
RoleAssignment.java
RoleSet.java
SystemUser.java
SystemUserBuilder.java
SystemUserDetails.java
Username.java
eapli.framework.infrastructur
eapli.framework.infrastructur
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
eapli.framework.infrastructure.pub
AuthzRegistry.java
Outline
SonarLint Rule Descripti...
eapli.framework.infrastructure.authz.applic
AuthzRegistry
authenticationService : AuthenticationS
authorizationSvc : AuthorizationService
userService : UserManagementService
authenticationService() : Authenticatior
authorizationService() : AuthorizationS
Problems
Javadoc
Declaration
Search
Console
Progress
Call Hierarchy
Git Staging
SonarLint On-The-Fly
Writable
Smart Insert
75 : 3 : 3258
201M of 256M

```java
 *
 */
@Singleton
public final class AuthzRegistry {

    private static AuthorizationService authorizationSvc;
    private static AuthenticationService authenticationService;
    private static UserManagementService userService;
```

eapli.framework.infrastructure.authz.application.AuthzRegistry

@Singleton

Registry of all authz service objects. Helper class for scenarios without spring Dependency Injection. In order to properly use the registry it is necessary to call its `configure` method in the start of the application to inject the proper `UserRepository` implementation.

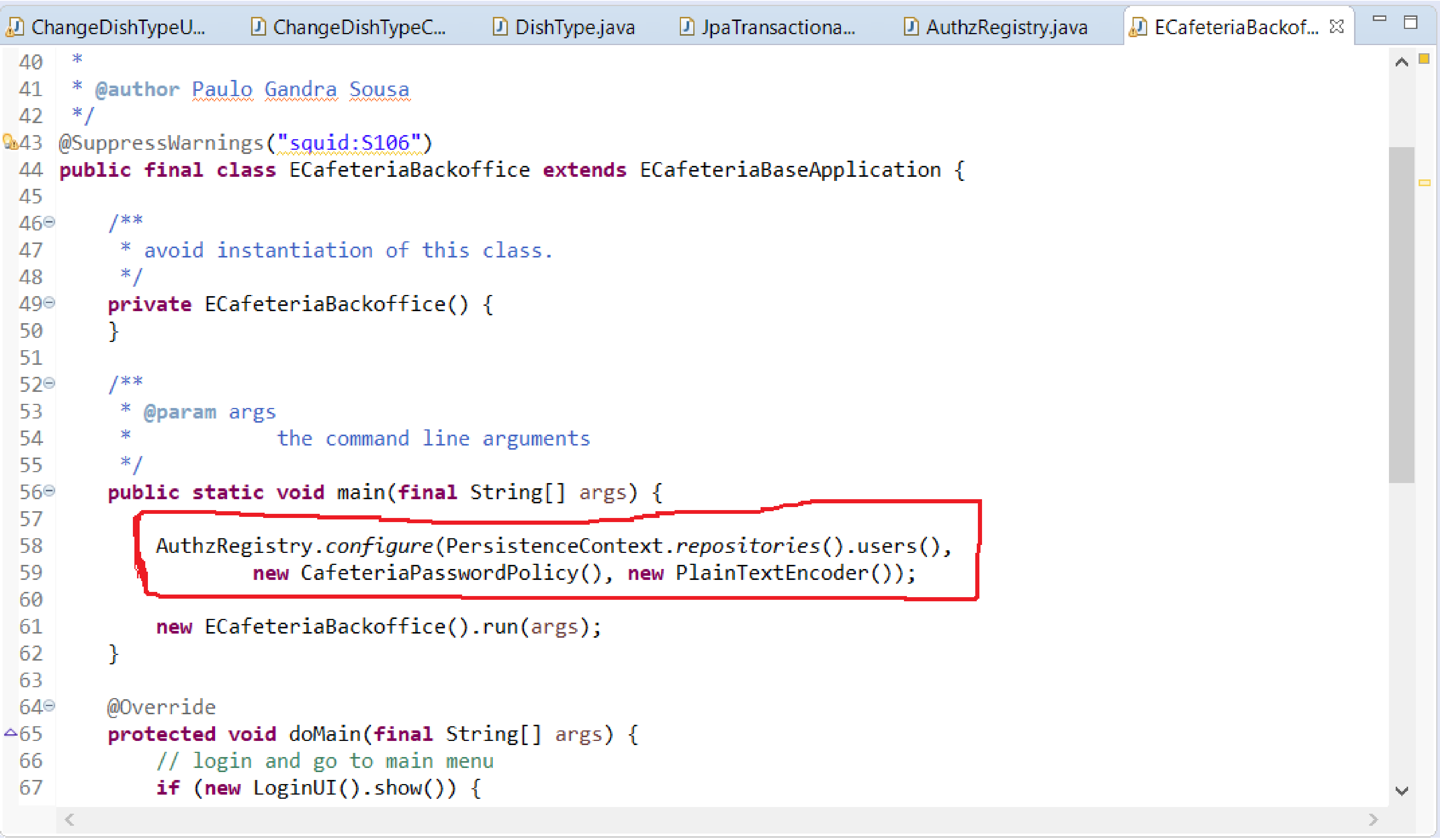
For instance,

```java
 public static void main(final String[] args) {
    AuthzRegistry.configure(PersistenceContext.repositories().users(),
            new CafeteriaPasswordPolicy(), new PlainTextEncoder());

    new ECafeteriaBackoffice().run(args);
}
```

Afterwards, in order to use these objects, you just need to grab the singleton from the registry, e.g.

```java
public class RegisterDishTypeController implements Controller {

  private final AuthorizationService authz = AuthzRegistry.authorizationService();
  private final DishTypeRepository repository = PersistenceContext.repositories().dishTypes();

  public DishType registerDishType(final String acronym, final String description) {
      authz.ensureAuthenticatedUserHasAnyOf(CafeteriaRoles.POWER_USER,
              CafeteriaRoles.MENU_MANAGER);

      final DishType newDishType = new DishType(acronym, description);
      return this.repository.save(newDishType);
  }
}
```

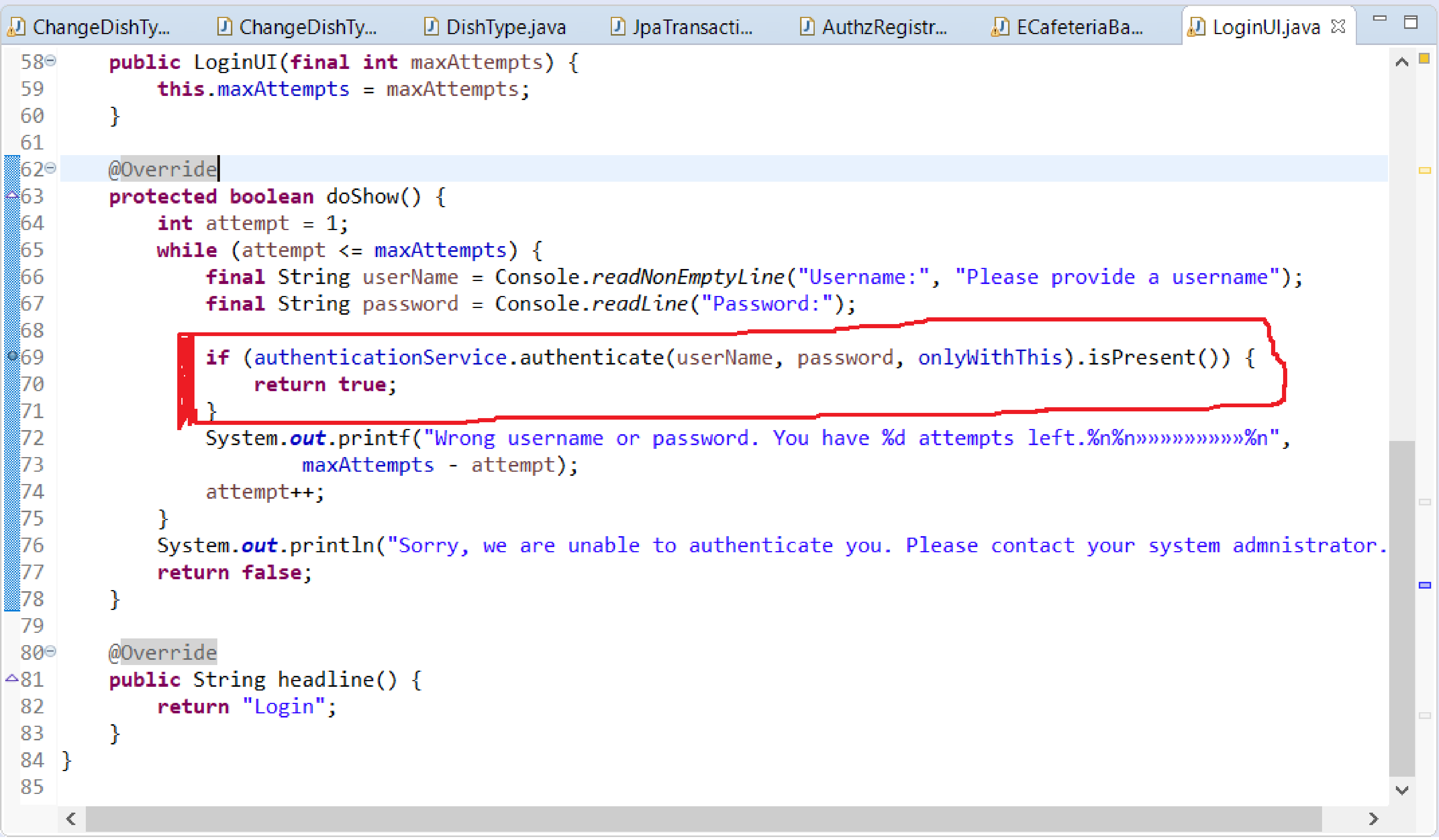
Main method



ChangeDishTypeU... ChangeDishTypeC... DishType.java JpaTransactiona... AuthzRegistry.java ECafeteriaBackof...

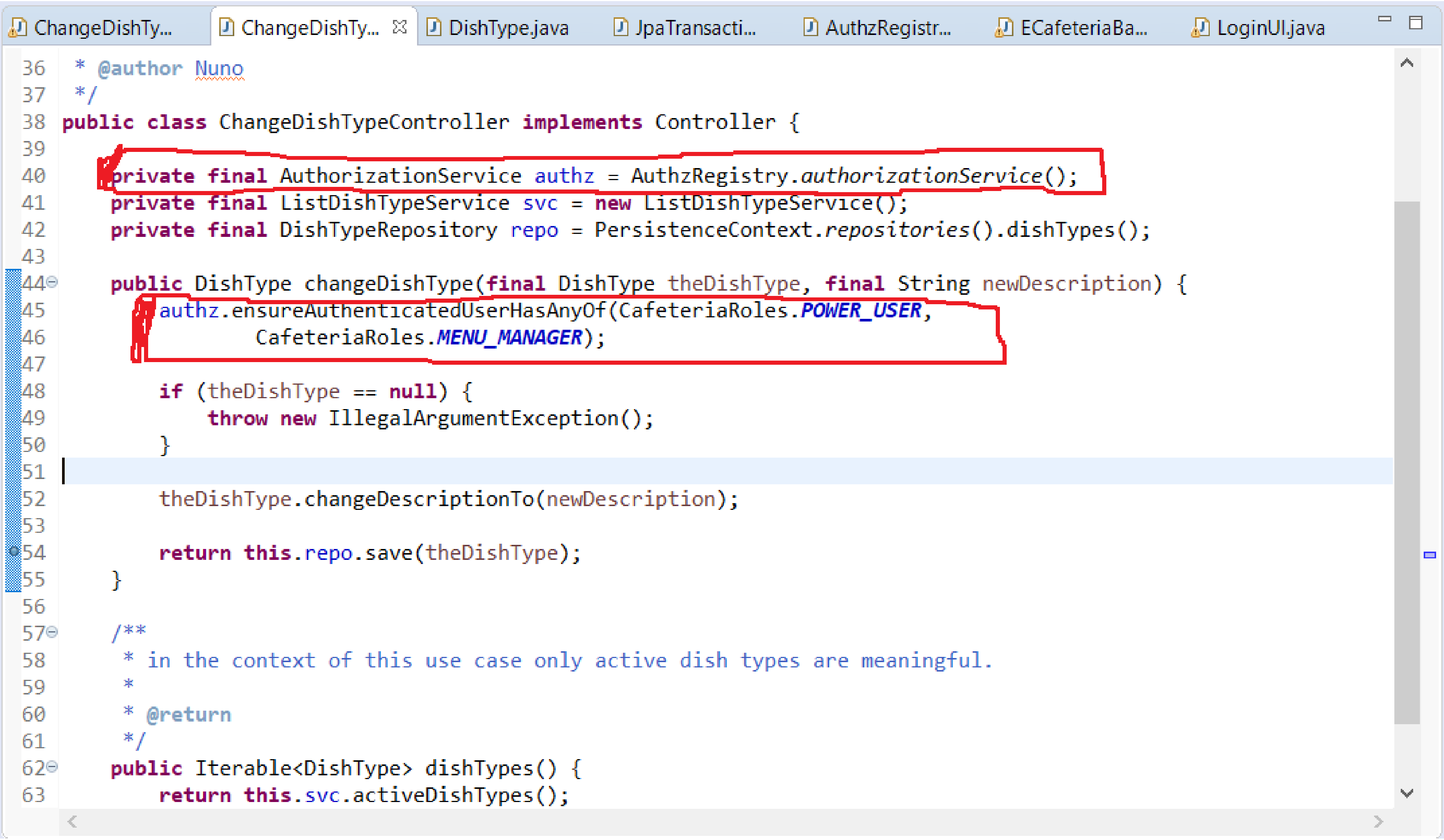
```
 *
 * @author Paulo Gandra Sousa
 */
@SuppressWarnings("squid:S106")
public final class ECafeteriaBackoffice extends ECafeteriaBaseApplication {

    /**
     * avoid instantiation of this class.
     */
    private ECafeteriaBackoffice() {
    }

    /**
     * @param args
     *            the command line arguments
     */
    public static void main(final String[] args) {

        AuthzRegistry.configure(PersistenceContext.repositories().users(),
                new CafeteriaPasswordPolicy(), new PlainTextEncoder());

        new ECafeteriaBackoffice().run(args);
    }

    @Override
    protected void doMain(final String[] args) {
        // login and go to main menu
        if (new LoginUI().show()) {
```

Perform authentication



```java
    public LoginUI(final int maxAttempts) {
        this.maxAttempts = maxAttempts;
    }

    @Override
    protected boolean doShow() {
        int attempt = 1;
        while (attempt <= maxAttempts) {
            final String userName = Console.readNonEmptyLine("Username:", "Please provide a username");
            final String password = Console.readLine("Password:");

            if (authenticationService.authenticate(userName, password, onlyWithThis).isPresent()) {
                return true;
            }
            System.out.printf("Wrong username or password. You have %d attempts left.%n%n»»»»»»»»»%n",
                    maxAttempts - attempt);
            attempt++;
        }
        System.out.println("Sorry, we are unable to authenticate you. Please contact your system admnistrator.
        return false;
    }

    @Override
    public String headline() {
        return "Login";
    }
}
```

Authorization in each controller



```java
 * @author Nuno
 */
public class ChangeDishTypeController implements Controller {

    private final AuthorizationService authz = AuthzRegistry.authorizationService();
    private final ListDishTypeService svc = new ListDishTypeService();
    private final DishTypeRepository repo = PersistenceContext.repositories().dishTypes();

    public DishType changeDishType(final DishType theDishType, final String newDescription) {
        authz.ensureAuthenticatedUserHasAnyOf(CafeteriaRoles.POWER_USER,
                CafeteriaRoles.MENU_MANAGER);

        if (theDishType == null) {
            throw new IllegalArgumentException();
        }

        theDishType.changeDescriptionTo(newDescription);

        return this.repo.save(theDishType);
    }

    /**
     * in the context of this use case only active dish types are meaningful.
     *
     * @return
     */
    public Iterable<DishType> dishTypes() {
        return this.svc.activeDishTypes();
```