

EAPLI

Teórico-Prática

# **Apresentação Projeto PL Base eCafeteria**

# eCafeteria

- Cafeteria management
- Users, Kitchen and Menu managers, Cashiers
- User app
- Backoffice app
  - Kitchen management
  - Menu management
  - Delivery station


# Scope of this lesson

- Project structure
- Architecture (same as proposed in classes)
- No business discussion
- Overview of existing codebase
- Focus on “User” and “Dish Type” domain concepts
- Two use cases
  - Register a new user
  - List all dish types

pag\_isep / EAPLI-201 x

Atlassian, Inc. [US] https://bitbucket.org/pag\_isep/eapli-2016-ecafeteria/src

BitbucketTeamsProjectsRepositoriesSnippets



EAPLI-2016-eCafeteria

ACTIONS

Clone

Create branch

Create pull request

Compare

Fork

NAVIGATION

Overview

Source

Commits

Branches

Pull requests

Downloads

Settings

Paulo Gandra de Sousa / EAPLI-2016-eCafeteria

Source

masterEAPLI-2016-eCafeteria /

.idea

backoffice.consoleapp

consoleapp.common

documentation

ecafeteria.bootstrapapp

ecafeteria.core

framework

utente.consoleapp

util

.gitignore	2.4 KB	3 days ago	added structure for menus of User App
Notes.txt	7.1 KB	2016-03-14	added notes documentation
README.md	378 B	2016-03-14	added notes documentation
build-console.bat	89 B	2016-03-14	updated script files templates
pom.xml	1.1 KB	3 hours ago	Comment tests ensureRoleTypeListIsNotEmpty, ensureInvalidAccessWithEmptyMemoryDatabase and ensureAuth
run-console.bat	455 B	2016-03-14	updated script files templates

Personal Expenses

Polythecnic of Porto, School of Engineering

EAPLI

=====

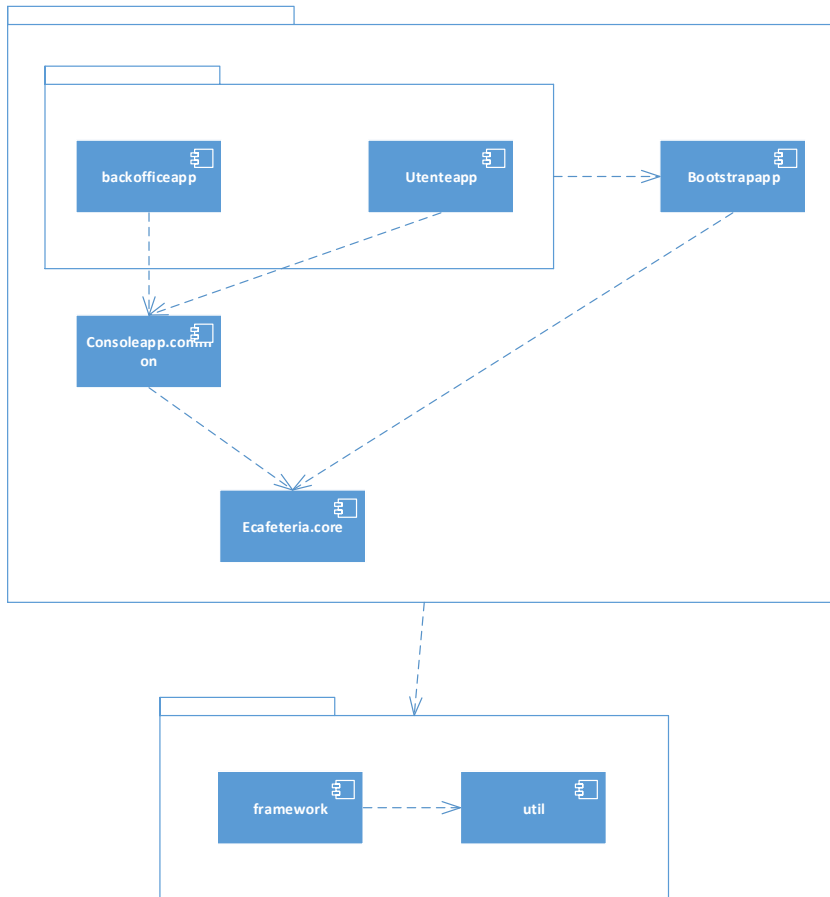
This application is part of the lab project for the course EAPLI.

Please check notes.txt for a discussion on design decissions.

BUILD

make sure JDK 1.7 is on the path

# Components (a.k.a. projects)



- Backofficeapp, utenteapp
- Core, console.common
- bootstrap
- Framework
  - Utility classes for DDD applications with JPA in EAPLI context
- Util
  - Generic utility classes

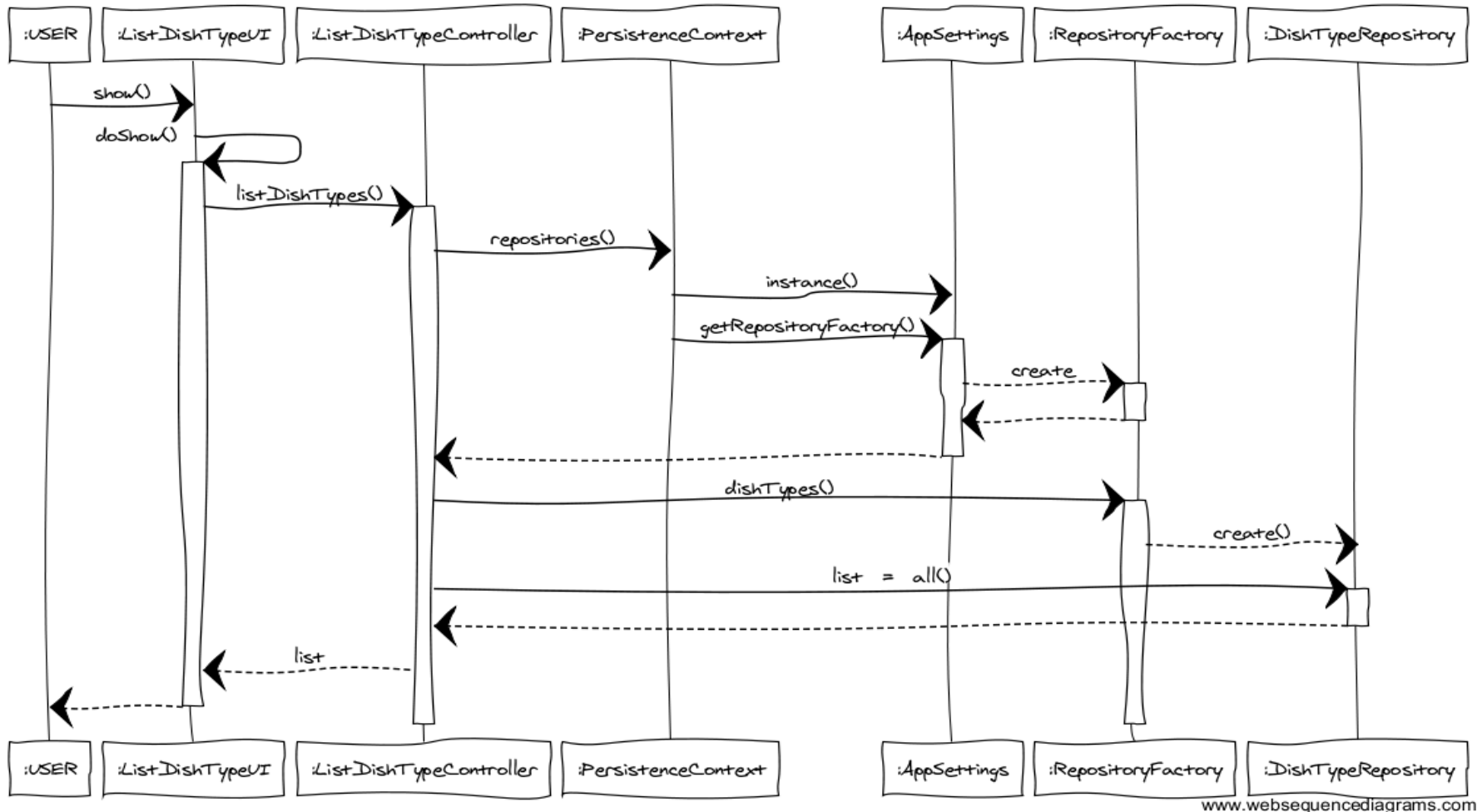
# eCafeteria design decisions

- Layers
  - Presentation
  - Application
  - Domain
  - Persistence
- Domain objects travel to UI for output

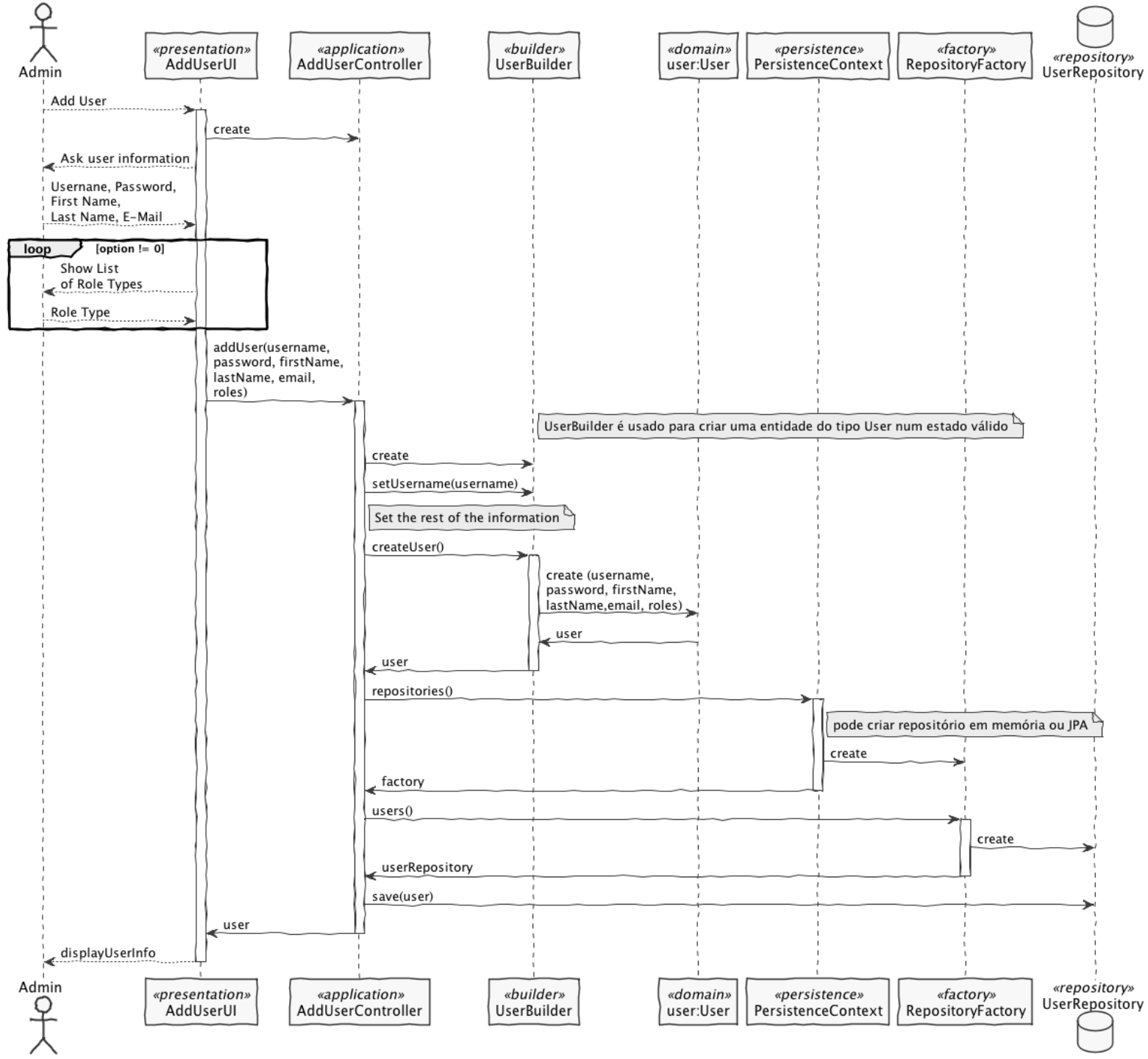
More on this  
next lesson

# List Dish Types

SD - List All Dish Types

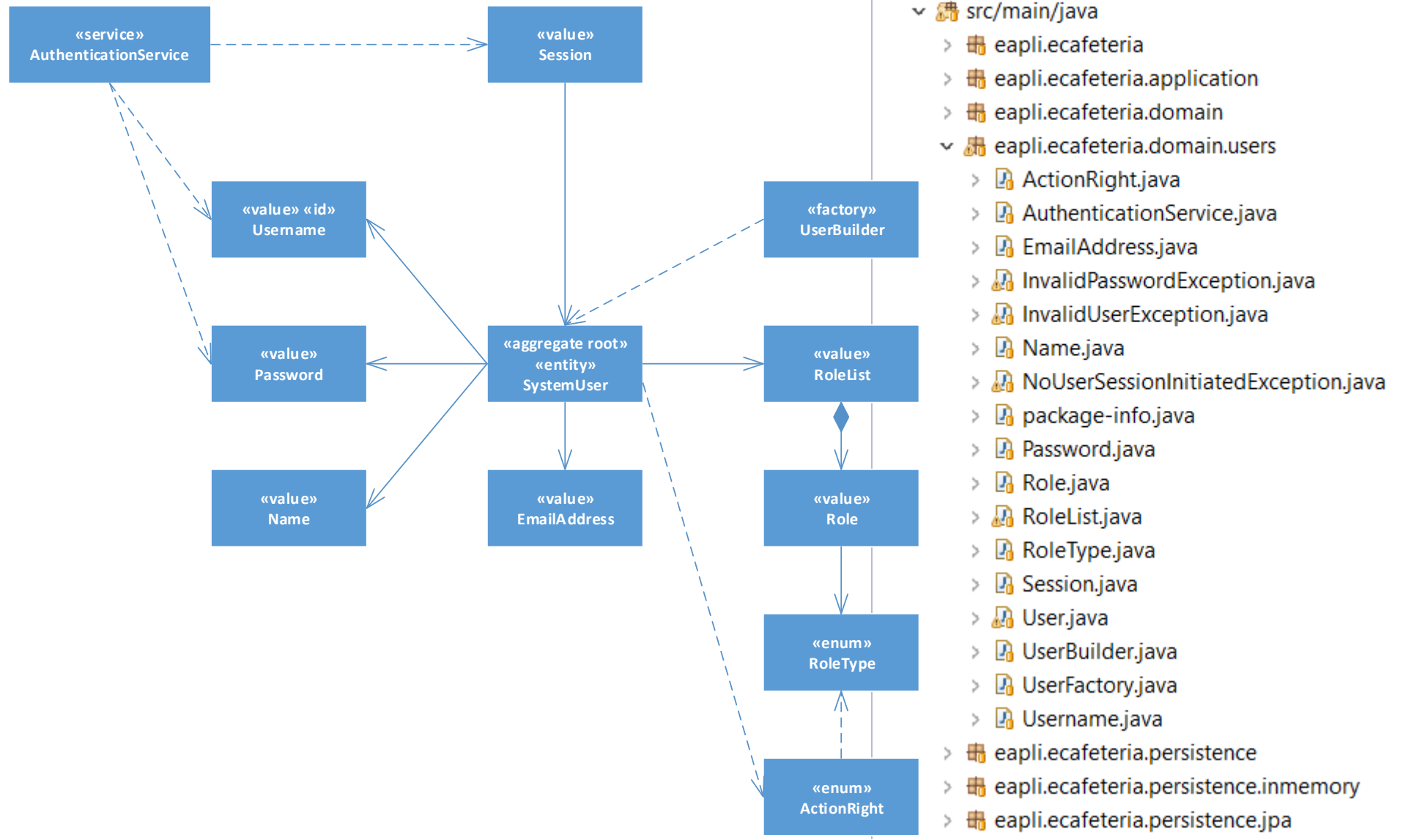


# Register New User





# User model



# Embeddable vs regular fields

```
@Entity
class SystemUser
{
```

```
    @Id
    String username;
    String password;
    ...
}
```

VS

```
@Entity
class SystemUser
{
```

```
    @EmbeddedId
    Username username;
    Password password;
    ...
}
```

```
@Embeddable
class Username
{
```

```
    String username;
```

```
    {
        @Embeddable
        class Password
        {
            String password;
        }
    }
}
```

PK		
Username	Password	- - -

# Domain invariants

```
@Test
public void ensurePasswordHasAtLeastOneDigitAnd6CharactersLong()
{
    new Password("abcdefgh1");
}

@Test(expected = IllegalArgumentException.class)
public void ensurePasswordsSmallerThan6CharactersAreNotAllowed()
{
    new Password("ab1c");
}

@Test(expected = IllegalArgumentException.class)
public void ensurePasswordsWithoutDigitsAreNotAllowed() {
    new Password("abcdefgh");
}
```

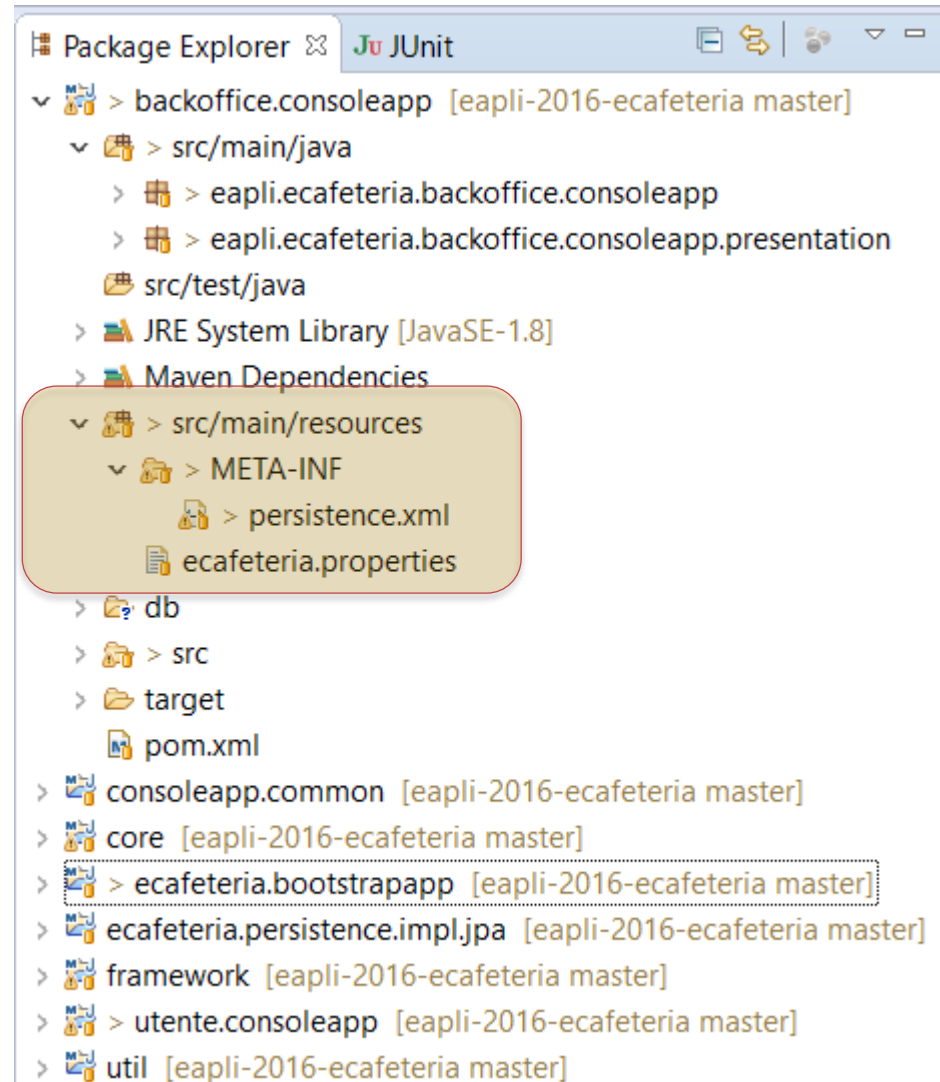
# Domain

```
public class Password {  
  
    ...  
  
    public Password(String password) {  
        if (!meetsMinimumRequirements(password)) {  
            throw new IllegalStateException();  
        }  
        thePassword = password;  
    }  
  
    private boolean meetsMinimumRequirements(String password) {  
        if (Strings.isNullOrEmpty(password)  
            || password.length() < 6  
            || !Strings.containsDigit(password))  
        {  
            return false;  
        }  
  
        return true;  
    }  
}
```

# Some additional design decisions

- Bootstrap data
- Support two repositories
  - In memory
  - Relational database
- Decide which repository implementation to use based on property file
- Simple main menu

# Resources



# bootstrap

```
public class ECafeteriaBootstrap implements Action {

    @Override
    public boolean execute() {
        // declare bootstrap actions
        final Action[] actions = { new UsersBootstrap(), };
        // execute all bootstrapping
        boolean ret = false;
        for (final Action boot : actions) {
            ret |= boot.execute();
        }
        return ret;
    }
}

public class UsersBootstrap implements Action {

    @Override
    public boolean execute() {
        registerAdmin();
        return false;
    }

    private void registerAdmin() {
        final String username = "admin";
        final String password = "admin";
        final String firstName = "John";
        final String lastName = "Doe";
        final String email = "john.doe@email.l.com";
        final List<RoleType> roles = new ArrayList<RoleType>();
        roles.add(RoleType.Admin);

        final UserRegisterController userController = new UserRegisterController();
        userController.registerUser(username, password, firstName, lastName, email, roles);
    }
}
```

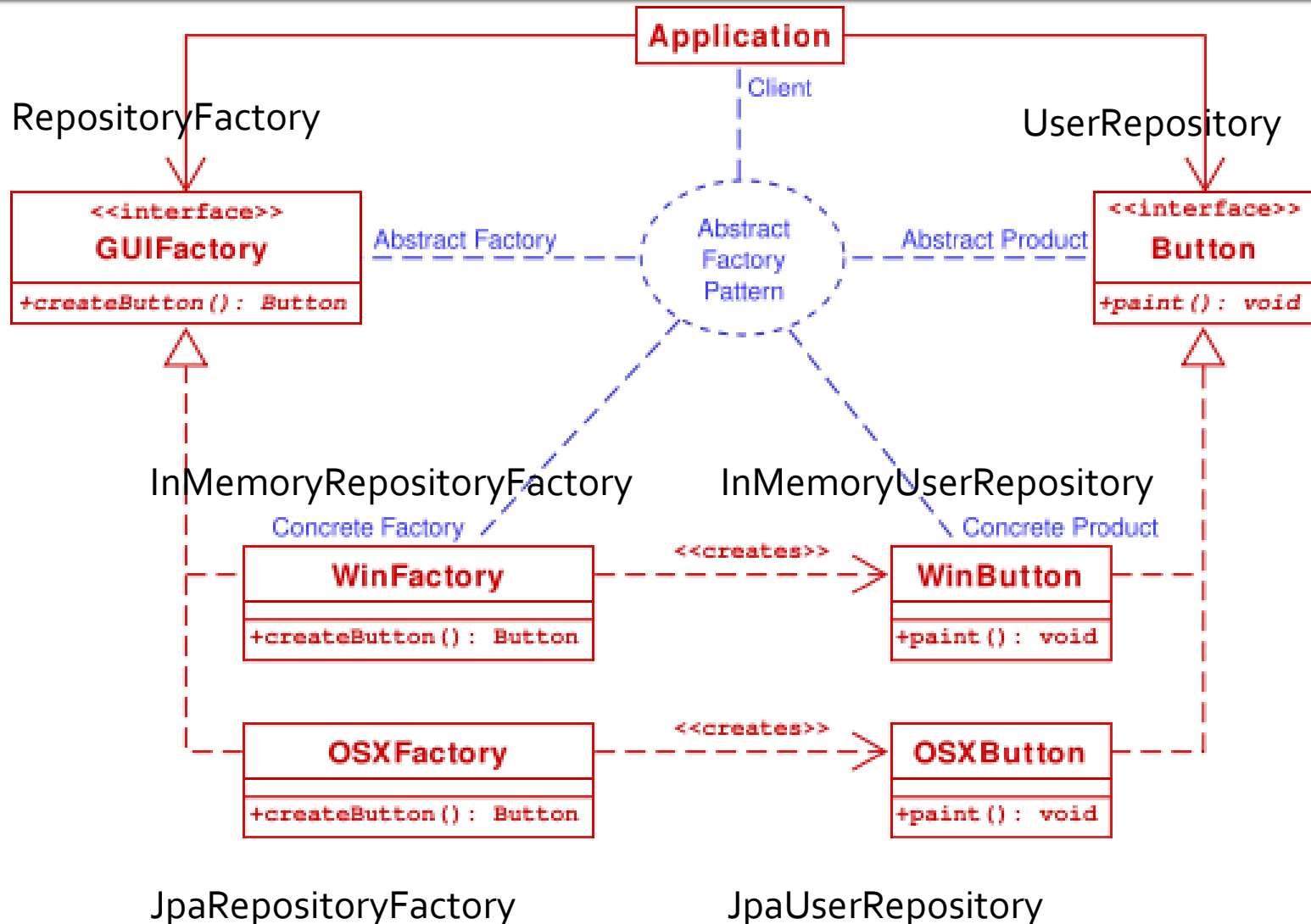
# Persistence

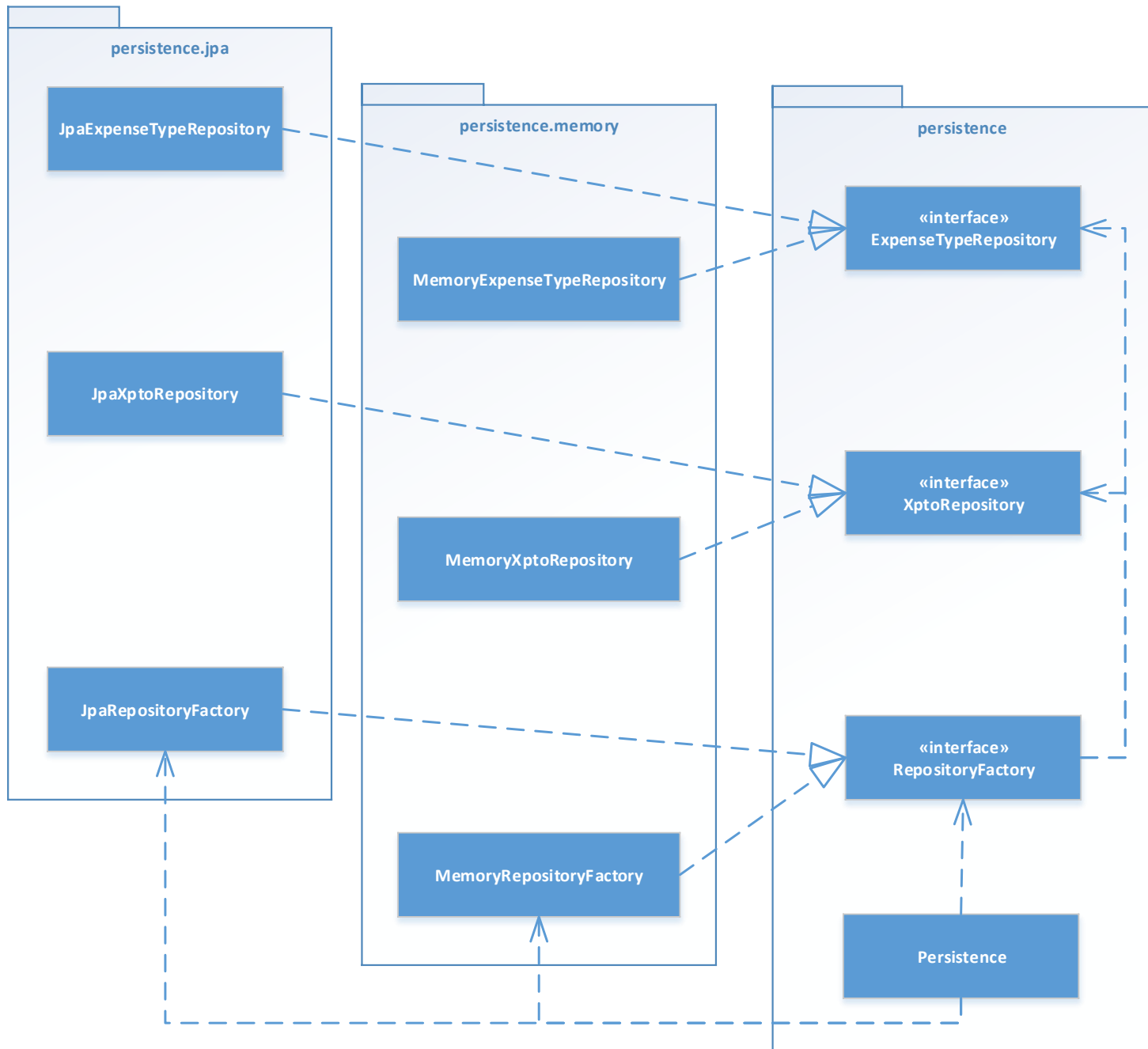
- Controller needs to access the repositories
- But we want to have two repository implementations
  - In memory
  - Relational database
- Hide persistence details from rest of the code
  - Interfaces
  - Factories



# Abstract Factory Pattern

PersistenceFactory





# Persistence Context

```
1. public final class PersistenceContext {
2.
3.     public static RepositoryFactory repositories() {
4.
5.         String factoryClassName = AppSettings.instance().getRepositoryFactory();
6.
7.         try {
8.
9.             return (RepositoryFactory) Class.forName(factoryClassName).newInstance();
10.
11.         } catch (ClassNotFoundException | IllegalAccessException | InstantiationException
12. ex) {
13.             Logger.getLogger(PersistenceContext.class.getName()).log(Level.SEVERE, null,
14. ex);
15.             return null;
16.         }
17.     }
18. }
```

# Persistence Context Usage

```
1. public class UserRegisterController {
2.
3.     public User registerUser(String username, String password, String
4.         firstName, String lastName, String email, List<RoleType> roles) {
5.         ....
6.         final UserBuilder userBuilder = new UserBuilder();
7.         userBuilder.setUsername(username);
8.         ....
9.         final User newUser = userBuilder.createUser();
10.
11.         final UserRepository userRepository =
12.             PersistenceContext.repositories().users();
13.         userRepository.save(newUser);
14.     }
15. }
```

# User Repository JPA / InMemory

## JPA Implementation

```
1. public class JpaUserRepository extends JpaRepository<User, Username>
   implements UserRepository {
2.
3.     @Override
4.     protected String persistenceUnitName\(\) {
5.         return PersistenceSettings.PERSISTENCE_UNIT_NAME;
6.     }
7. }
```

## In Memory Implementation

```
1. public class InMemoryUserRepository extends InMemoryRepository<User,
   Username> implements UserRepository {
2. }
```

# Build

- Maven
  - Dependency manager
  - Artifact (jar) repository
  - Build automation
  - Other tasks, e.g., deploy, run
- Works for Eclipse, IntelliJ, Netbeans
- Pom.xml



AddRoleType... AddUserActio... AddUserUI.java ECafeteriaBo... ECafeteriaBa... ECafeteriaU... UsersBootst... MainMenu.java AbstractUIJa

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>eapli</groupId>
6     <artifactId>ecafeteria.backoffice.consoleapp</artifactId>
7     <version>1.0-SNAPSHOT</version>
8     <packaging>jar</packaging>
9
10    <properties>
11        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
13        <maven.compiler.source>1.8</maven.compiler.source>
14        <maven.compiler.target>1.8</maven.compiler.target>
15    </properties>
16
17    <name>ecafeteria.backoffice.consoleapp</name>
18
19    <dependencies>
20        <dependency>
21            <groupId>eapli</groupId>
22            <artifactId>framework</artifactId>
23            <version>1.0-SNAPSHOT</version>
24        </dependency>
25        <dependency>
26            <groupId>eapli</groupId>
27            <artifactId>ecafeteria.bootstrapapp</artifactId>
28            <version>0.0.1-SNAPSHOT</version>
29        </dependency>
30        <dependency>
31            <groupId>org.eclipse.persistence</groupId>
32            <artifactId>org.eclipse.persistence.jpa</artifactId>
33            <version>2.6.2</version>
34        </dependency>
35        <dependency>
36            <groupId>com.h2database</groupId>
37            <artifactId>h2</artifactId>
38            <version>1.4.191</version>
39        </dependency>
40    </dependencies>
41 </project>
```

## Dependencies

Filter:

## Dependency Management

### Dependencies

```
framework : 1.0-SNAPSHOT
ecafeteria.bootstrapapp : 0.0.1-SNAPSHOT
org.eclipse.persistence.jpa : 2.6.2
h2 : 1.4.191
```

Add...

Remove

Properties...

Manage...

 Remove from Build Path Configure Build Path...

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.

[Overview](#) [Dependencies](#) [Dependency Hierarchy](#) [Effective POM](#) [pom.xml](#)

Problems @ Javadoc Declaration Console Call Hierarchy History

0 errors, 29 warnings, 0 others

### Description

Resource

> ⚠ Warnings (29 items)



# Use cases implemented in base project

- Add user
- List users
- Deactivate user
- Check permissions
- Signup
- Approve new user
- Add dish type
- Edit dish type
- Deactivate dish type
- List dish types
- Register organic unit

# Next steps

1. Read project description
2. Discuss and clear assumptions in PL
3. Clone class' repository
  - One for each PL class
4. Study base code
5. Analyse – design – code – test – document

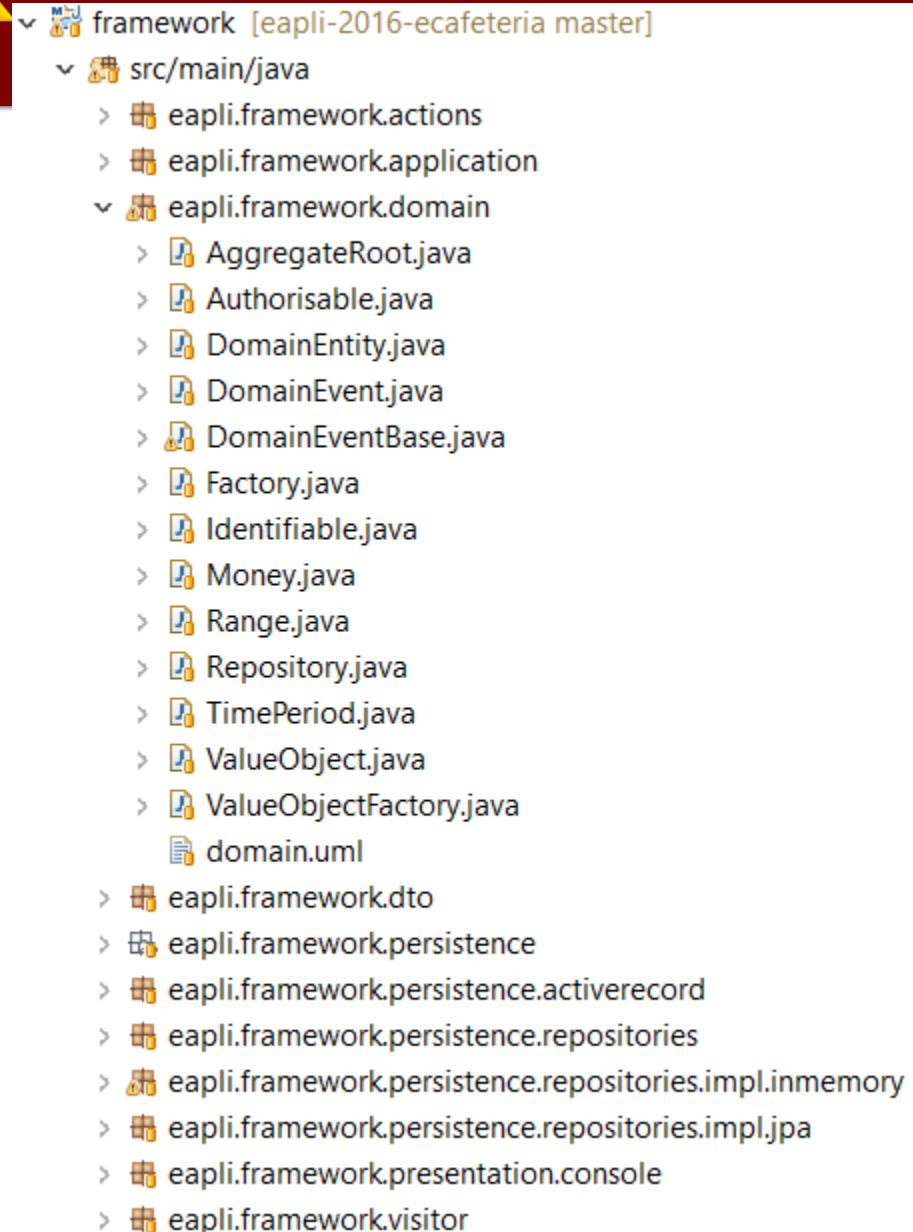
# EAPLI Framework

# Eapli.Util

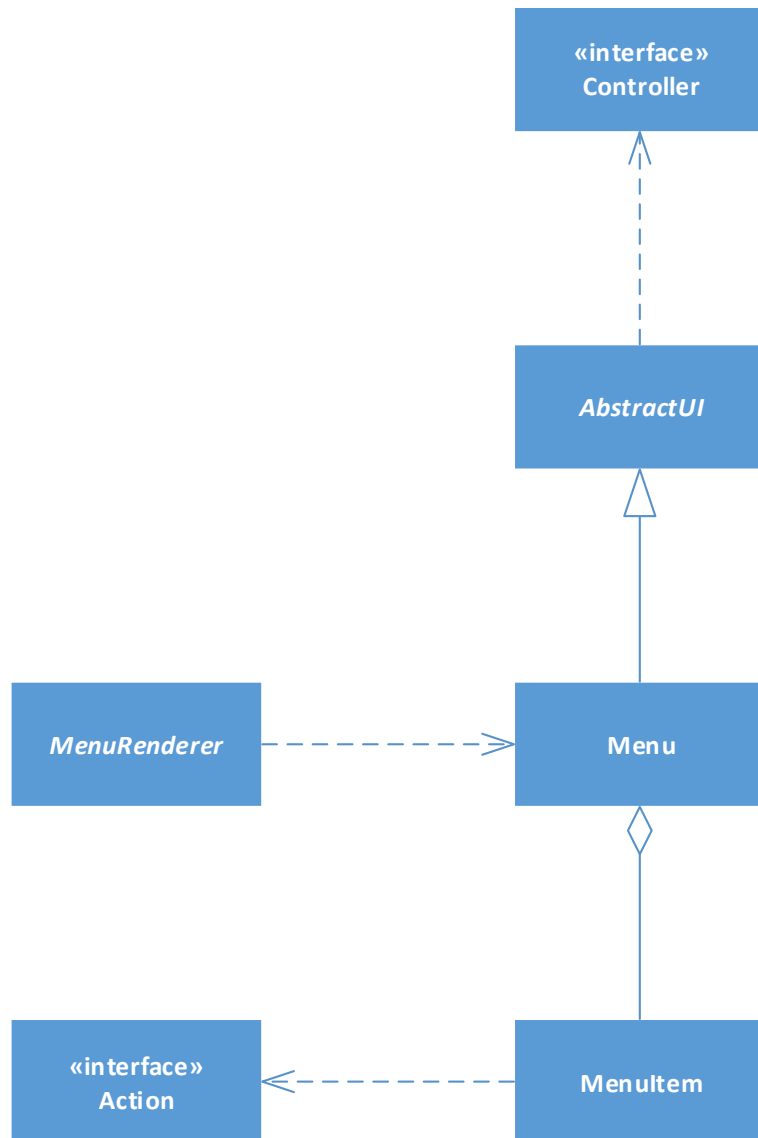
- Console
  - Helper console reading functions
- DateTime
  - Simplifies manipulation of dates and times thru java Calendar
- Files
  - File manipulation helper
- Math
  - Sample math utility
- RomanNumeral
  - Represents a decimal number as a Roman numeral
- Strings
  - String manipulation

# Eapli.Framework

- Domain objects
  - Money
  - Range
  - Time period
- DDD pattern interfaces
  - ValueObject
  - DomainEntity
  - AggregateRoot



# Eapli.Framework



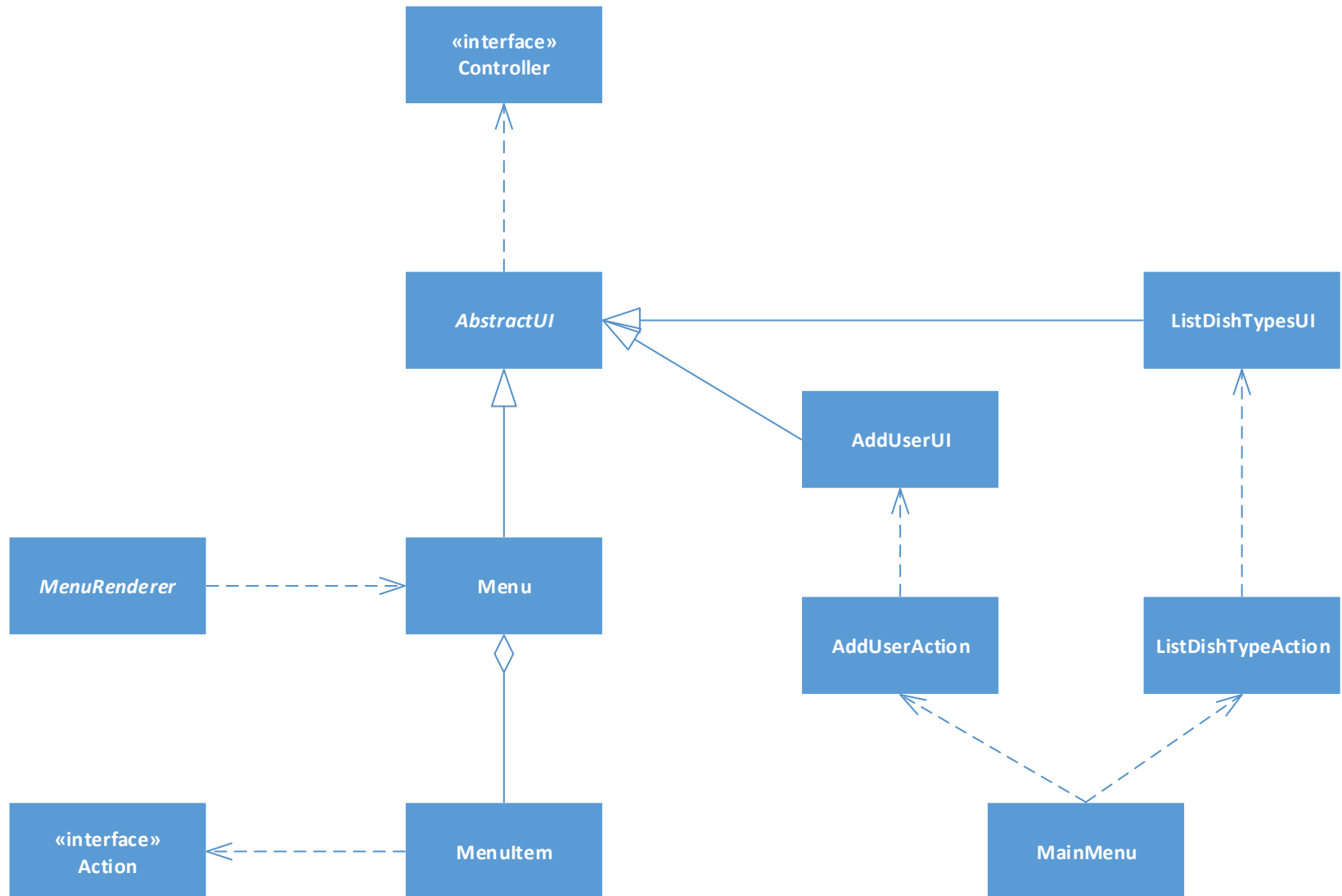
- src/main/java
  - eapli.framework.actions
    - Action.java
    - CompoundAction.java
    - ExitAction.java
    - IfThenAction.java
    - NullAction.java
    - ReturnAction.java
    - ShowMessageAction.java
  - eapli.framework.application
  - eapli.framework.domain
  - eapli.framework.dto
  - eapli.framework.persistence
    - eapli.framework.persistence.activerecord
    - eapli.framework.persistence.repositories
      - eapli.framework.persistence.repositories.impl.inmemory
      - eapli.framework.persistence.repositories.impl.jpa
  - eapli.framework.presentation.console
    - AbstractUI.java
    - HorizontalMenuRenderer.java
    - ListWidget.java
    - Menu.java
    - MenuItem.java
    - MenuRenderer.java
    - SelectWidget.java
    - ShowUiAction.java
    - ShowVerticalSubMenuAction.java
    - SubMenu.java
    - VerticalMenuRenderer.java
    - VerticalSeparator.java
  - eapli.framework.visitor

# Eapli.Framework

- Persistence
  - Repository interfaces
- Implementations
  - JPA
  - InMemory list

```
▼ framework [eapli-2016-ecafeteria master]
  ▼ src/main/java
    > eapli.framework.actions
    > eapli.framework.application
    > eapli.framework.domain
    > eapli.framework.dto
    > eapli.framework.persistence
    > eapli.framework.persistence.activerecord
    ▼ eapli.framework.persistence.repositories
      > DeleteableRepository.java
      > IterableRepository.java
      > package-info.java
      > Repository.java
    ▼ eapli.framework.persistence.repositories.impl.inmemory
      > InMemoryRepository.java
      > NotFoundException.java
    ▼ eapli.framework.persistence.repositories.impl.jpa
      > JpaRepository.java
      > package-info.java
    > eapli.framework.presentation.console
    > eapli.framework.visitor
```

# presentation





```
12  */
13  public abstract class AbstractUI {
14
15      public static final String SEPARATOR = "+-----+";
16      public static final String BORDER    = "+=====+";
17
18      * derived classes should provide the Controller object. an example of the
19      protected abstract Controller controller();
20
21      /**
22       * derived classes should override this method to perform the actual
23       * rendering of the UI. follows the Template Method pattern
24       *
25       * @return true if the user wants to leave this UI
26       */
27      protected abstract boolean doShow();
28
29      * derived classes should override this method to provide the title of the
30      public abstract String headline();
31
32      public void mainLoop() {
33          boolean wantsToExit;
34          do {
35              wantsToExit = show();
36          } while (!wantsToExit);
37      }
38
39      /**
40       *
41       * @return true if the user wants to leave this UI
42       */
43      public boolean show() {
44          drawFormTitle();
45          final boolean wantsToExit = doShow();
46          drawFormBorder();
47          // Console.waitForKey("Press any key.");
48
49          return wantsToExit;
50      }
51
52      protected void drawFormTitle() {
53          System.out.println();
54          drawFormTitle(headline());
55      }
56  }
```



AddRoleType2List.java AddUserAction.java AddUserUI.java ECafeteriaBootstrap.java ECafeteriaBackoffice.java ECafeteriaUtenteApp.java UsersBoots

```
21  ~/  
22  public class MainMenu extends AbstractUI {  
23  
24  /**  
25   * @return true if the user selected the exit option  
26   */  
27  @Override  
28  public boolean doShow() {  
29      final Menu menu = buildMainMenu();  
30      final MenuRenderer renderer = new VerticalMenuRenderer(menu);  
31      return renderer.show();  
32  }  
33  
34  @Override  
35  public String headline() {  
36      return "eCAFETERIA [" + AppSettings.instance().session().authenticatedUser().id() + "]";  
37  }  
38  
39  private Menu buildMyUserMenu() {  
40      final Menu myUserMenu = new Menu("My account >");  
41  
42      myUserMenu.add(  
43          new MenuItem(CHANGE_PASSWORD_OPTION, "Change password", new ShowMessageAction("Not implemented yet"));  
44      myUserMenu.add(new MenuItem(LOGIN_OPTION, "Change user (Login)", new LoginAction()));  
45      myUserMenu.add(new MenuItem(LOGOUT_OPTION, "Logout", new LogoutAction()));  
46  
47      return myUserMenu;  
48  }  
49  
50  private Menu buildMainMenu() {  
51      final Menu mainMenu = new Menu();  
52  
53      final Menu myUserMenu = buildMyUserMenu();  
54      mainMenu.add(new SubMenu(MY_USER_OPTION, myUserMenu, new ShowVerticalSubMenuAction(myUserMenu)));  
55  
56      mainMenu.add(new VerticalSeparator());  
57  
58      if (AppSettings.instance().session().authenticatedUser().isAuthorizedTo(ActionRight.Administer)) {  
59          final Menu usersMenu = buildUsersMenu();  
60          mainMenu.add(new SubMenu(USERS_OPTION, usersMenu, new ShowVerticalSubMenuAction(usersMenu)));  
61  
62          final Menu organicUnitsMenu = buildOrganicUnitsMenu();  
63          mainMenu.add(new SubMenu(ORGANIC_UNITS_OPTION, organicUnitsMenu,
```

```
17  */
18  public class ListDishTypeUI extends AbstractUI {
19
20      private final ListDishTypeController theController = new ListDishTypeController();
21
22      @Override
23      protected Controller controller() {
24          return theController;
25      }
26
27      @Override
28      protected boolean doShow() {
29          List<DishType> list = theController.listDishTypes();
30          if (list.isEmpty()) {
31              System.out.println("There is no registered Dish Type");
32          } else {
33              System.out.printf("%30s---%6s\n", "Dish Type description ---", "Active");
34              for (DishType dT : list) {
35                  System.out.printf("%30s--- %1sB\n", dT.description(), dT.isActive());
36              }
37          }
38          return true;
39      }
40
41      @Override
42      public String headline() {
43          return "List Dish Types";
44      }
45  }
46
```