Automation and framework review

Plan for selenium bootcamp

- **Framework and automation review. —> me**
  - Today -> automation interview questions
  - Next one —> framework interview questions
- **Selenium, Cucumber review —> Jamal, 2 days**

- **CI/CD, jenkins, grid, docker —> Vasyl, 2 days**

How did you work remotely?

- Core hours: **9 am EST to 3 pm EST**
- response time: **any message must be answered within 40 mins**
- Slack status: **never offline, either available, in a meeting**

Integration tests

- Type of tests where tests the different modules of a software/application. Unit tests can focus on one functionality. Smoke test is done to get quick feedback about the general well being of the application. Regression is done to test the application extensively.
- Integration is a test that done specifically when we have a new functionality and we test that new functionality together with the other functionalities that depend on it/interact with with.
- Integration tests can be done with Selenium and also API. We should do more integration tests with API

Integration tests: scenario

- **UI EXAMPLE**
  In vytrack app, new functionality is create calendar event. Once we test create calendar events, we also test dashboard page which uses the calendar events. Dashboard page depends on the calendar events.

- **API**
  In vytrack app, new functionality calendar events, we build apis for creating calendar events. Then we test the dashboard page related apis to see if that can get the same calendar events.

End to end testing

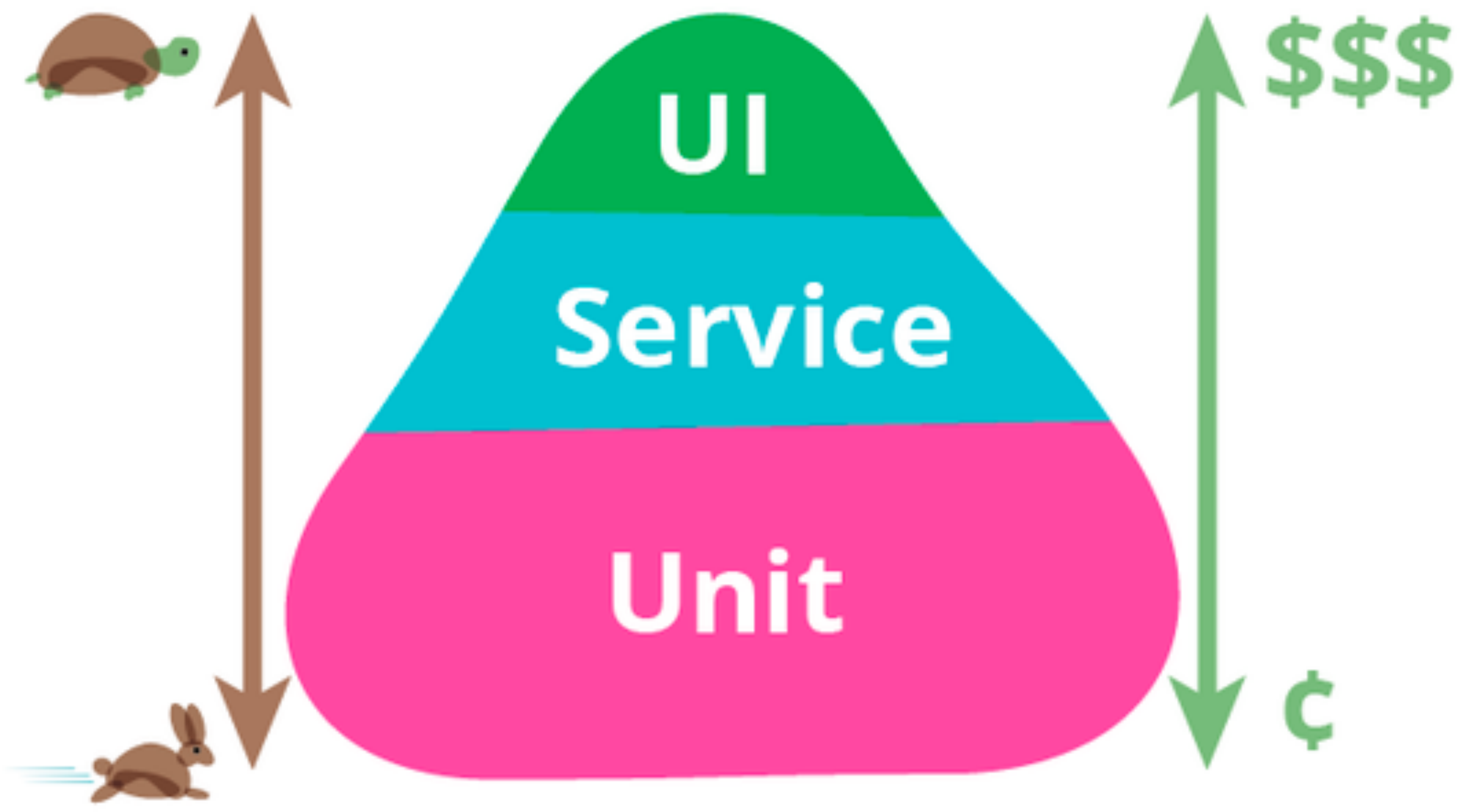
- A type of testing where we test the application from beginning to end. It can get really big if have application with many functionalities. That is because in e2e testing we test every module, every system, every functionality of the application.
- **Horizontal e2e testing:**
  - We test the user interface, but we focus on the functionality. If we take example amazon, e2e test will be register, login, search for product, verify search results page, select product, verify product details, add to card verify cart details, pay verify payment, select shipping options verify shipping.
- **Vertical e2e testing:**
  - In vertical e2e testing we test the components of the applications. Search for product, get result for product. Get the same product info from API and compare the results with UI, get the same info from DB and compare the results.

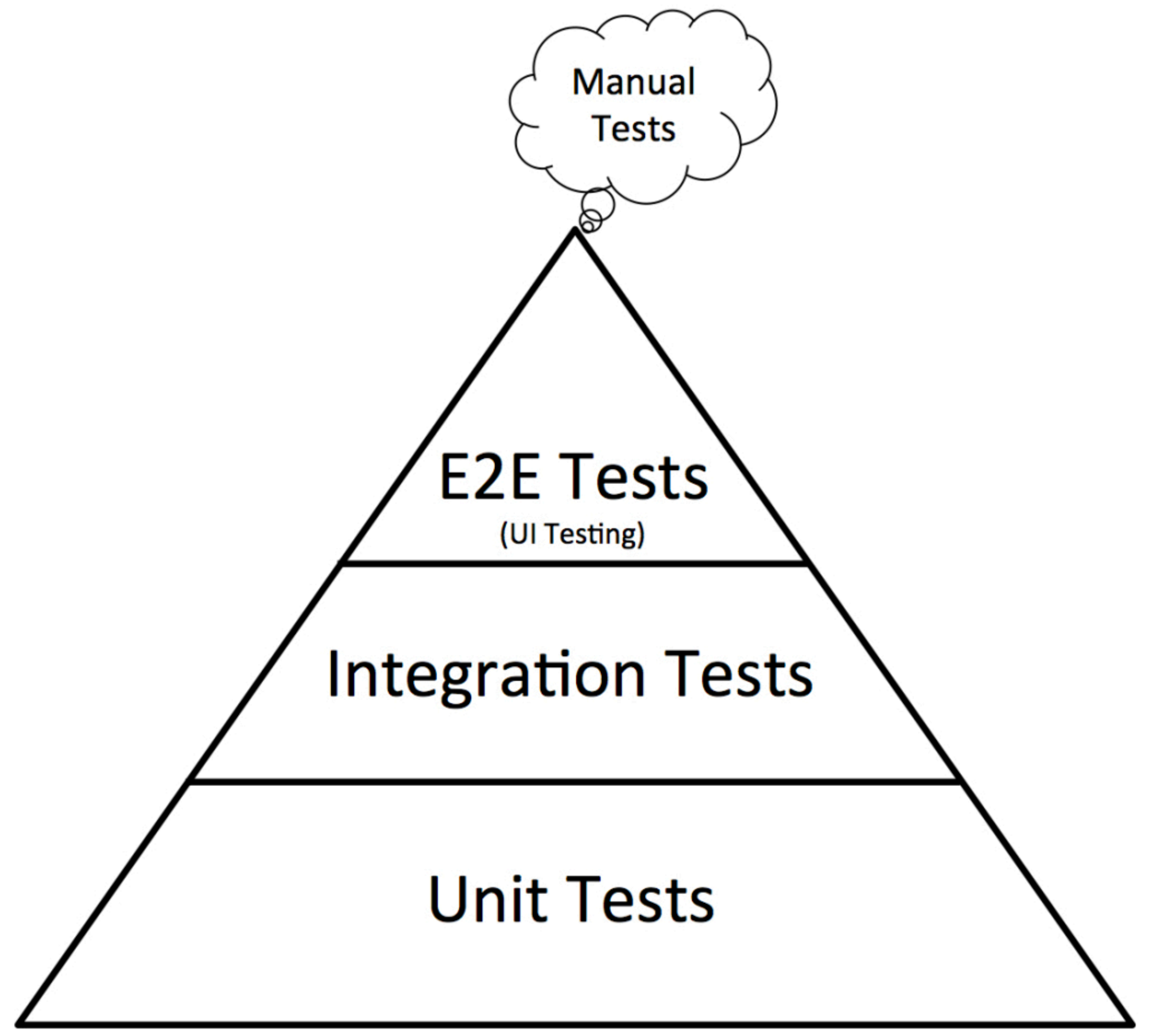
From Ruslan to Everyone:
process of buying wooden spoon from Etsy

End to end testing with selenium

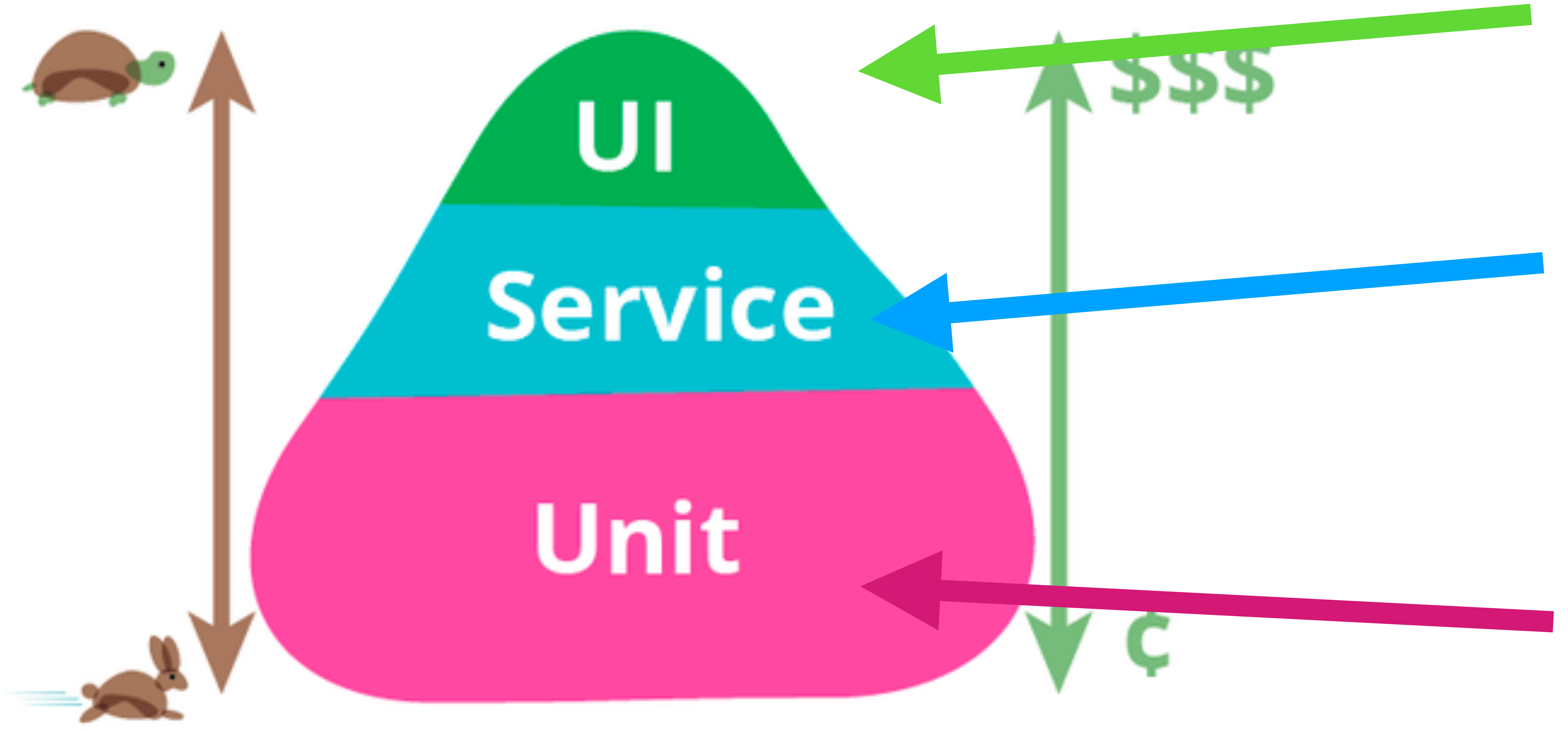
- For e2e scenarios that test only include UI we use selenium
- For e2e scenarios that test app layers of the application, selenium (UI), rest assured (API), databases (jdbc)

Testing pyramid





Testing pyramid



Testers / qa / sdet

Testers / qa / sdet do it.
developers can also do this

Only Developers do it

- **Test plan**: a document that details everything that is related to testing **certain application**: scope, tools, timing, who does what, exit criteria…. Usually written by **lead, manager**. Before any testing starts.
- **Plan strategy**: can be one separate document and one for all company. It can also be a part of test plan. It outlines how we approach testing.
- **Test approach**: part if the test plan. Defines how approach testing such as reactive (build first test later) , proactive (test as software is build)

- BREAK come back later 3.09 pm est

Why did you use selenium, not proctractor

- In my company we use java, so wanted to have a tools that works with java
- **Popularity**: for selenium there is a lot of community support. We can learn more about best practices of selenium easier. More resources.
- **Maturity**: selenium has been around for sometime, it has come a long way. Has had many upgrades bug fixes etc..
- **Integration** with other tools: there any available reporting tools, CI/CD integration options for selenium

Why did you use java, not …..?

- Same toolset in the company
- Popularity : many libraries, tools for reporting, testing are available in java.
- Compatibility with other tools

How do you handle flaky tests?

- Flake test can happen due to wait issues, application instability, test case complexity, tests with dependencies.
- 1. Synchronization (implicit and explicit and fluent waits)
- 2. Reduce the number of test case (atomic testing)
- 3. Use api, db where we can in the test to create data
- 4. **Do not use those flaky tests in you smoke/regression tests**.

What synchronization issue did you face?

- In my framework we have pages that used ui overlay when loading information. These overlay had different load time. Sometimes it is visible for 1 setons, other time for 5 secons. Another time it is visible, then disappears then visible again.
- I used Synchronization (implicit and explicit and fluent waits) to address issue
-

How did you do UI, API and Db testing in your framework?

- In current project I had test tests with ui, api and db.
- In some tests cases even though I did not have API in test steps, I used api for test data creation.
- I used api, and db when cleaning up the database after tests.
- I used Selenium for UI, rest assured for api, jdbc for database.

Did you have same framework for UI and API?

- There are project where they have API and UI framework separate. In those project it means that we do not have test vases where we test UI and API together.
- There are project where UI and API is in the same framework.
- Some applications 100 percent API, so they have only 1 framework.

- BREAK come back later 4.04 pm est

What to you pay attention to in your CODE REVIEWS

- In my company we coding standards documents. We use it as a refecence in your code review. It has standards about variables names, formatting etc.
- One of the SOLID Principles is **single responsibility**. It means each class will do one job. Page objects —> locators. Tests —> test steps. We do not have any locators in the test classes.
- DRY —> any repetition should be another method or utility. (dont repeat yourself)
- I make sure that scenario steps from jira is fully reflected in the test (requirements).

What are the common exceptions you get and to handle them?

- NoSuchElementeException (findELement, when locating)
- ElementeNotVisibleException / ElementeNotInteractibleException (click, sendKeys , when we try to interact with an element)
- TimeOutException (explicits waits)
- StaleElementException (click, sendKeys , when we try to interact with an element)

What happens if we use explicit and implicit wait at the same time?

- It wait for both of the given duration (at least)

- Implicit = 0;
- wait.until(explicit condition)
- Implicit = 5;
-

Difference between driver.get(“https://google.com”) and driver.navigate().to(“https://google.com”)

- We used to say that driver.get waits for page to load fully, navigate to does not. In internet you will find many resources that say this.
- It is not like that. They both wait. **They same when it comes to waiting for page load**

Locators strategy
Show me 5 different ways of location certain element

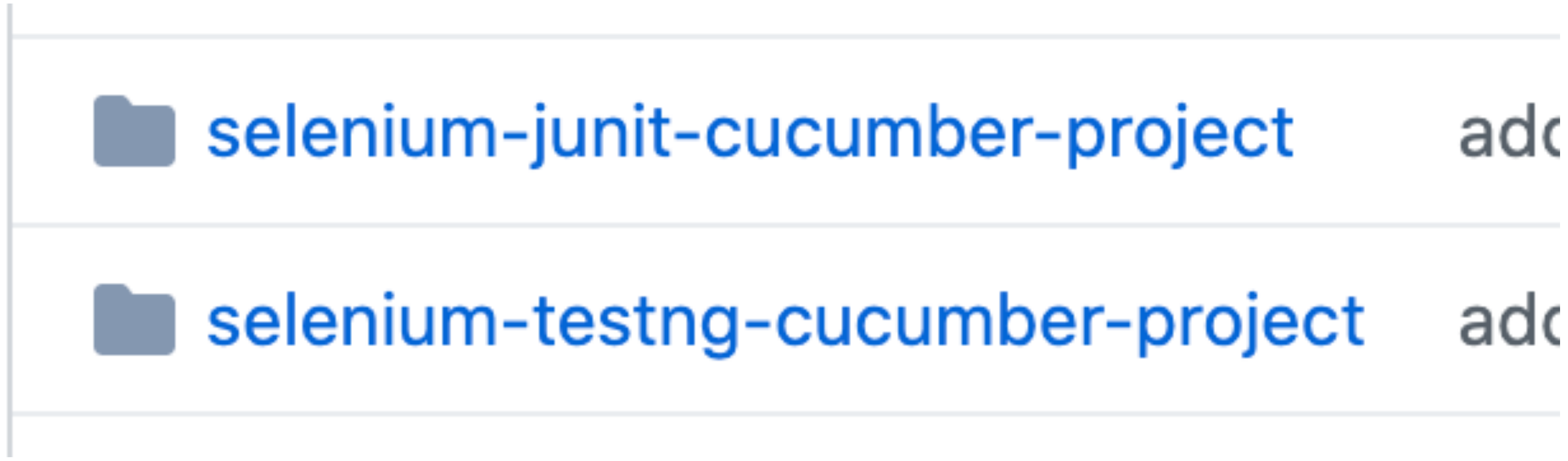
- When choosing locator for element, this is my order
  - **Id**
  - **Name**
  - **Tag name**
  - **Class name**
  - **Css** or **xpath**
- If my element is a link I start with **linkText** and **partialLinkText**, if they are not suitable the I go to the list above.

What is javascript executor and when you need it?

- It is a class in selenium api what helps us send/inject javascript code to the browser.

- We can use it to do actions which is not working working regular selenium methods.

Github project

- Update your existing re-bootcamp-spring-2020 repository.
- There 2 new projects under the same repository
- url: https://github.com/CybertekSchool/re-bootcamp-spring-2020
- 

selenium-junit-cucumber-project adc

selenium-testng-cucumber-project adc

How do you choose tests case for automaton in your project?

- **Backlog grooming** we discuss the story: requirements and test scenarios. In my project during that discussion as an automation engineer I decided if the story should be automated or not. Then based on that we vote on the size/point of story.
- Don’t automate everything.
- Smoke tests
- Business need: tests cases that has most business value
- What users use mostly?
- Most critical:
- Data driven tests
- Complex test cases
- Test data generation must be automated as well. (this is not about test case)
- Regression, integration

How do you choose tests case for automaton in your project?
Part two

- Don’t automate everything.
- Features without complete requirements and that may change soon
- flaky/unstable scenarios
- visual/ look and feel test case
- Buggy/broken scenarios
- Some experts also argue against automating long/complex test cases: hard to build, hard to verify if they fail,
- Too many edge cases

Automation framework → TOOLS

Java

Selenium WebDriver

IntelliJ

Cucumber BDD

JUnit

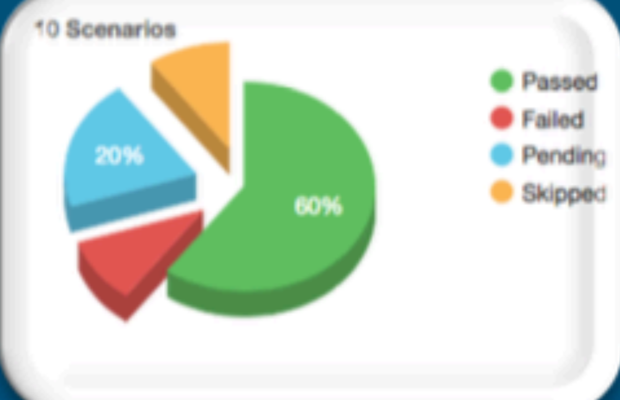
Maven

REST Assured

JDBC

Apache POI

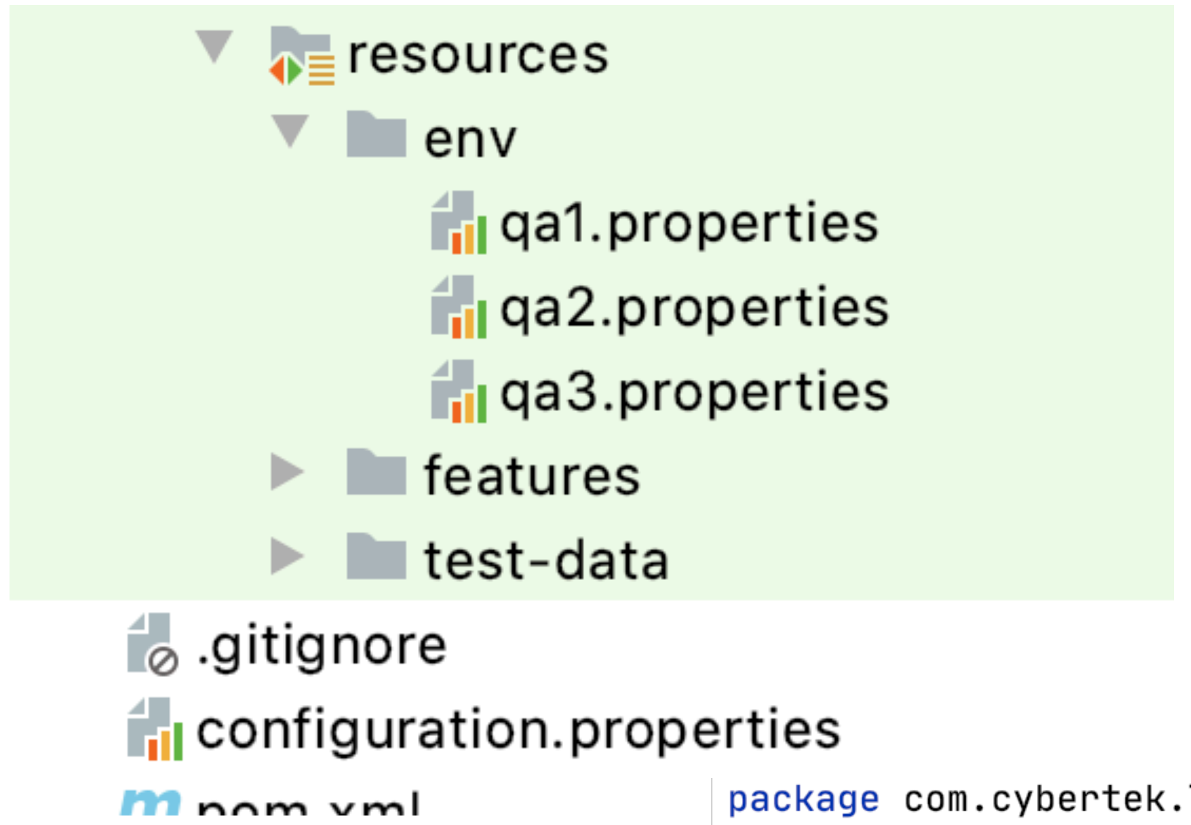
Git



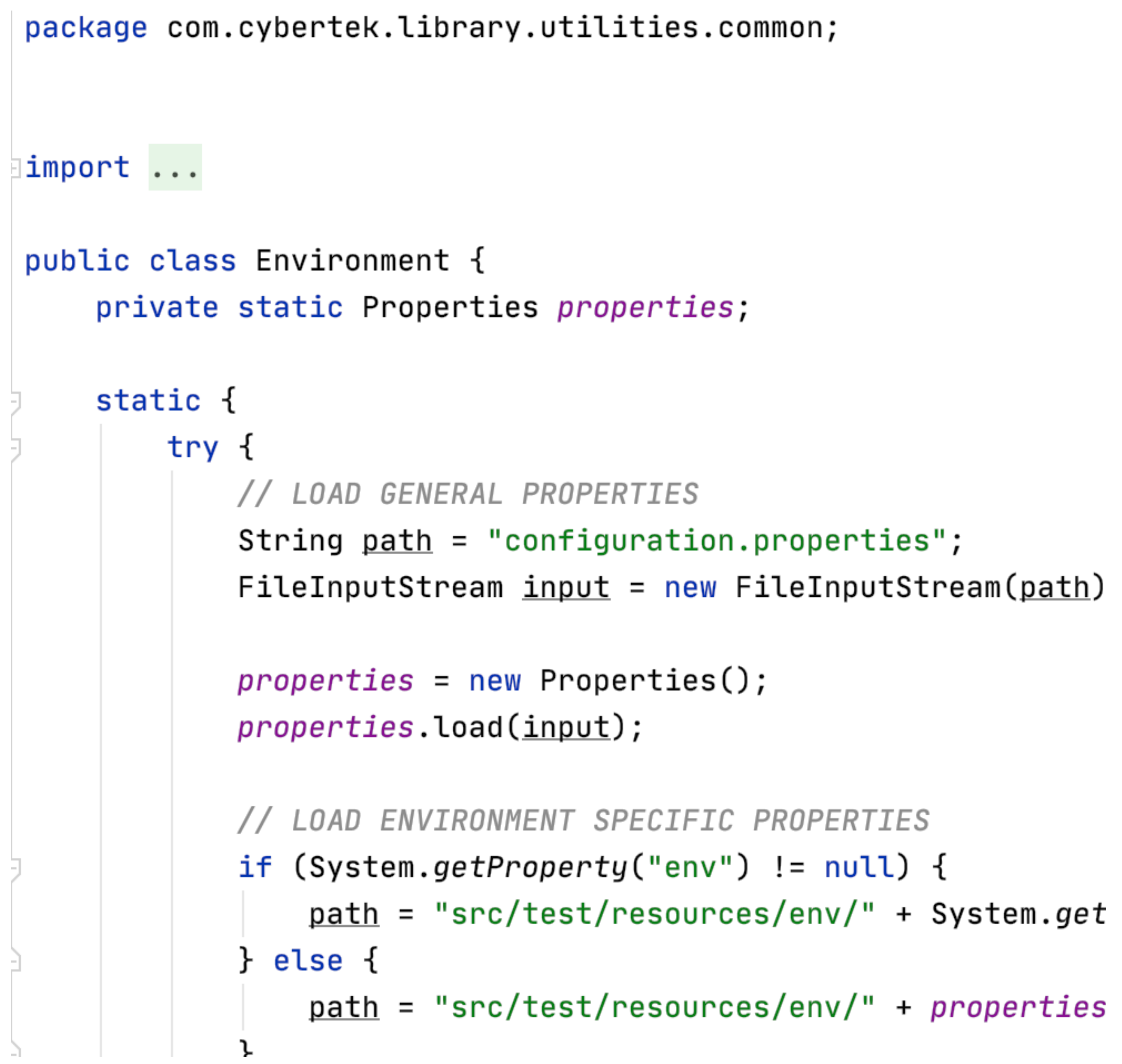
HTML Reporting

How did you handle environments in your projects:

I did 2 things.
1. I had a Environment utility class which helped me handle the information related to 3 different environments. This class loads the information specific to each information from the properties file for that environment. I can pass the environment type form the main configuration.properties file or I can pass it from the command line.
2. I created steps to autogenerate test data using api. This way my test cases don't have dependency on environment specific test data.
•



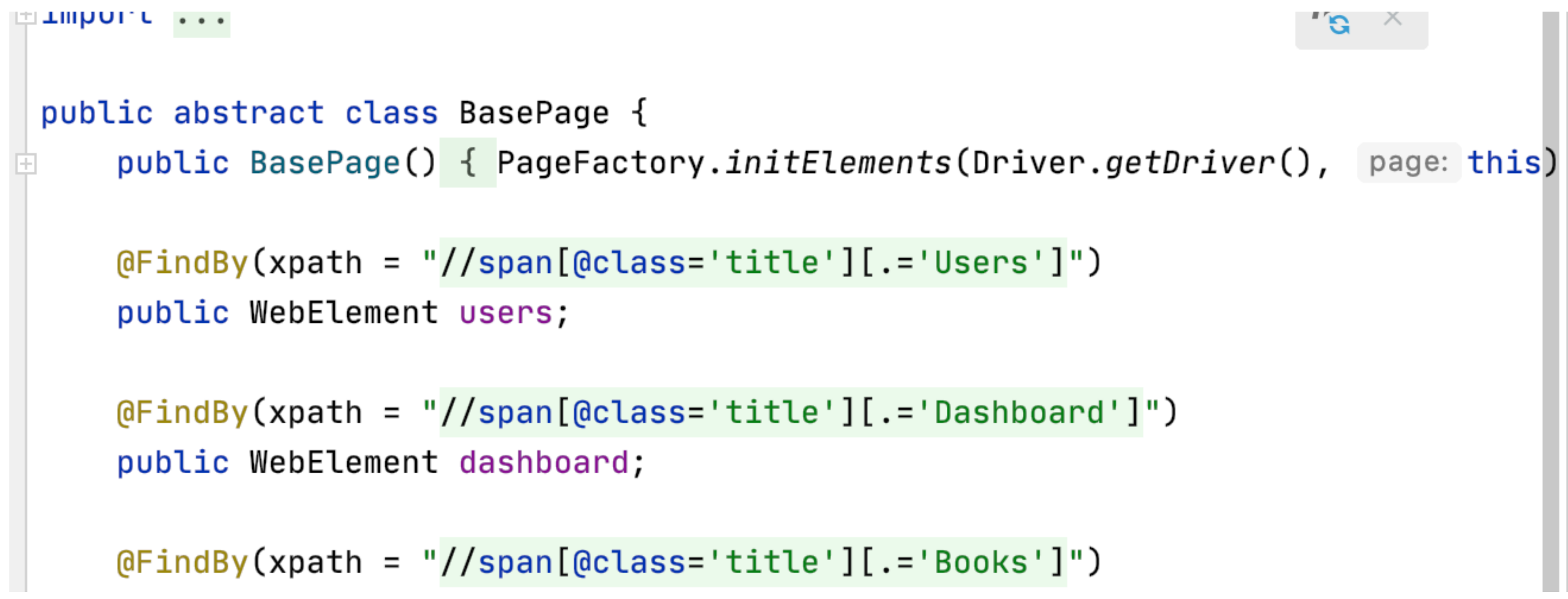
```
▼ resources
  ▼ env
      qa1.properties
      qa2.properties
      qa3.properties
  ▶ features
  ▶ test-data
.gitignore
configuration.properties
pom.xml
```



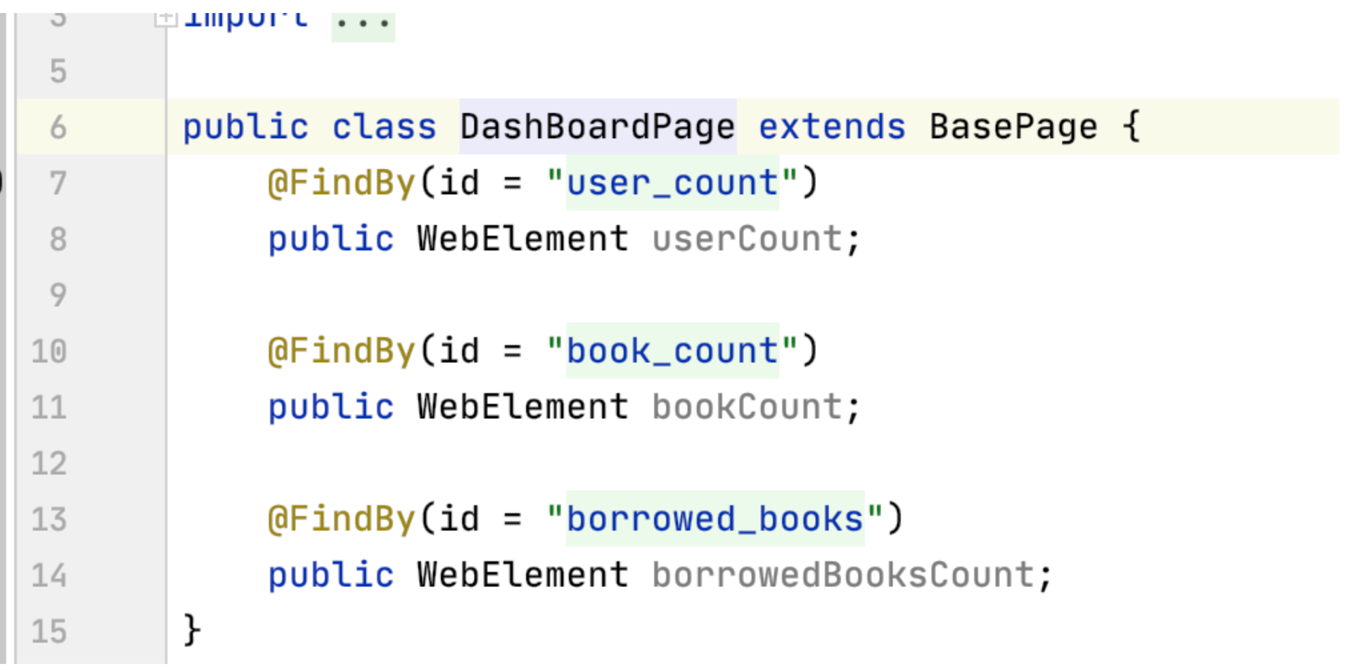
```
package com.cybertek.library.utilities.common;


import ...

public class Environment {
    private static Properties properties;

    static {
        try {
            // LOAD GENERAL PROPERTIES
            String path = "configuration.properties";
            FileInputStream input = new FileInputStream(path)

            properties = new Properties();
            properties.load(input);

            // LOAD ENVIRONMENT SPECIFIC PROPERTIES
            if (System.getProperty("env") != null) {
                path = "src/test/resources/env/" + System.get
            } else {
                path = "src/test/resources/env/" + properties
            }
```

HOW DID YOU USE OOP CONCEPTS IN YOUR FRAMEWORK

- **Inheritance** Page object classe extend the base page which has the common elements



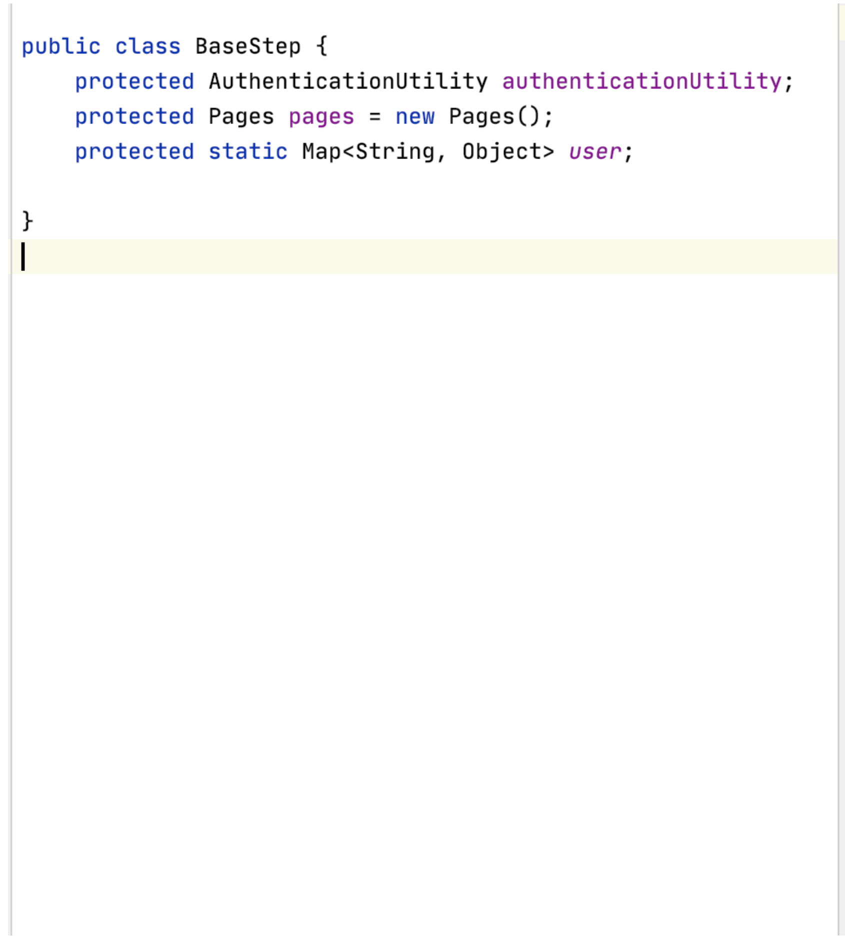
```java
import ...

public abstract class BasePage {
    public BasePage() { PageFactory.initElements(Driver.getDriver(), page: this)

    @FindBy(xpath = "//span[@class='title'][.='Users']")
    public WebElement users;

    @FindBy(xpath = "//span[@class='title'][.='Dashboard']")
    public WebElement dashboard;

    @FindBy(xpath = "//span[@class='title'][.='Books']")
```



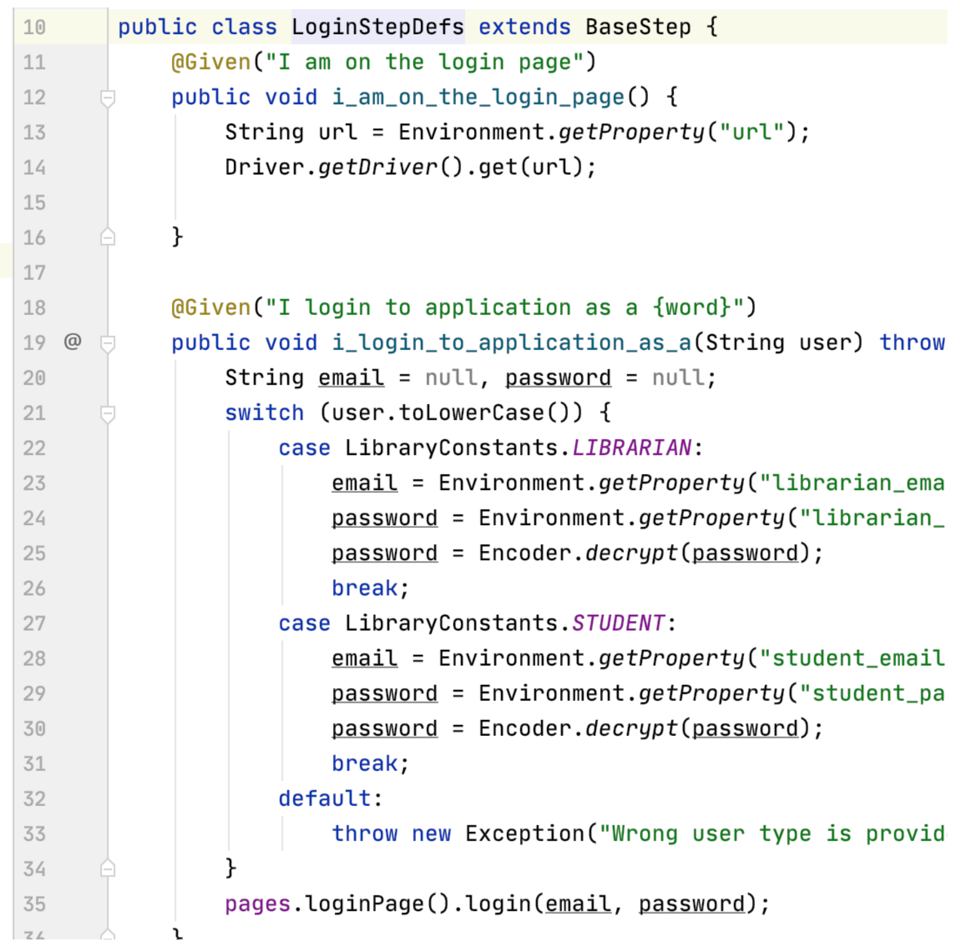
```java
import ...

public class DashBoardPage extends BasePage {
    @FindBy(id = "user_count")
    public WebElement userCount;

    @FindBy(id = "book_count")
    public WebElement bookCount;

    @FindBy(id = "borrowed_books")
    public WebElement borrowedBooksCount;
}
```

- Step defs extend the common

- base Step class



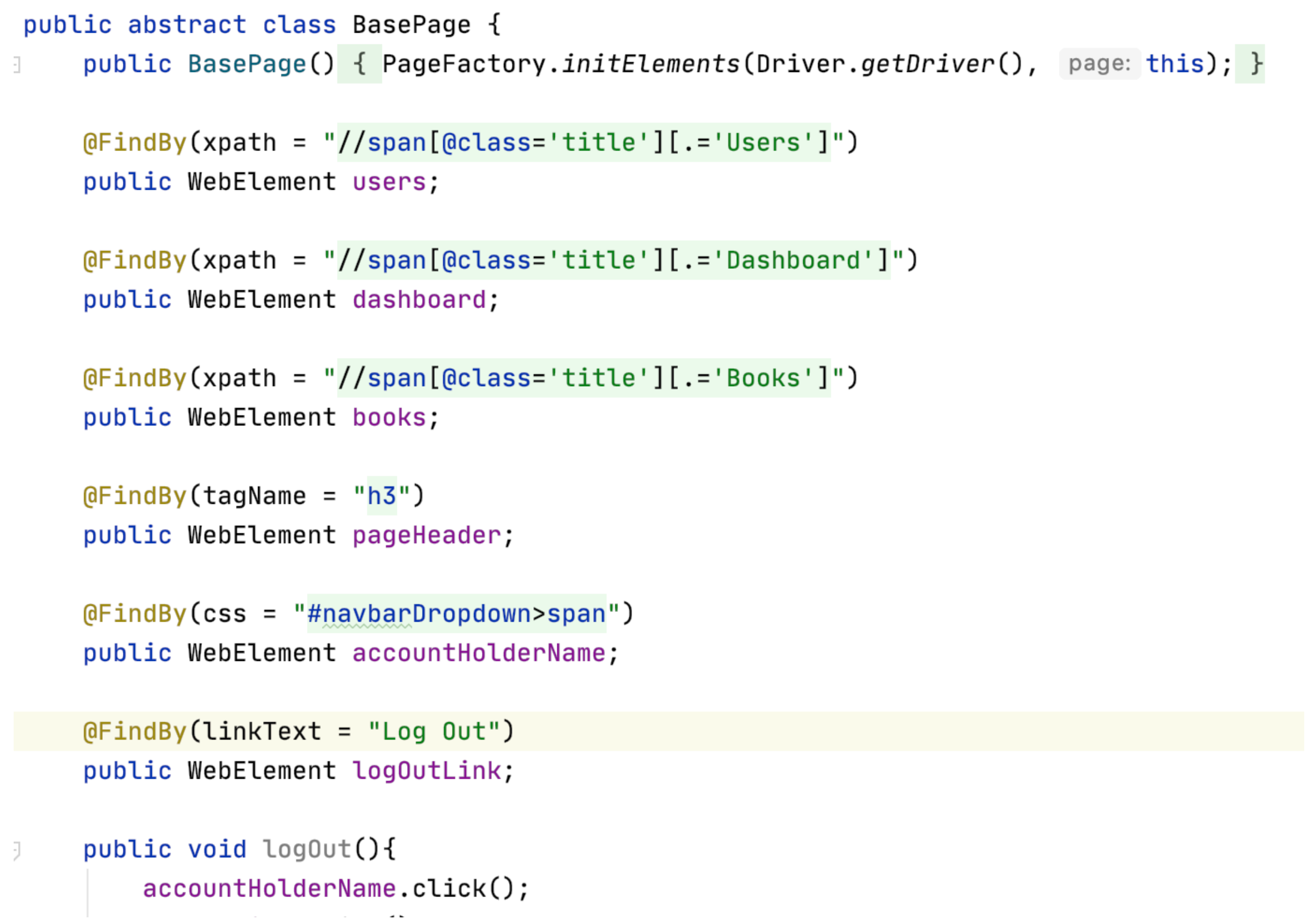
```java
public class BaseStep {
    protected AuthenticationUtility authenticationUtility;
    protected Pages pages = new Pages();
    protected static Map<String, Object> user;

}
```



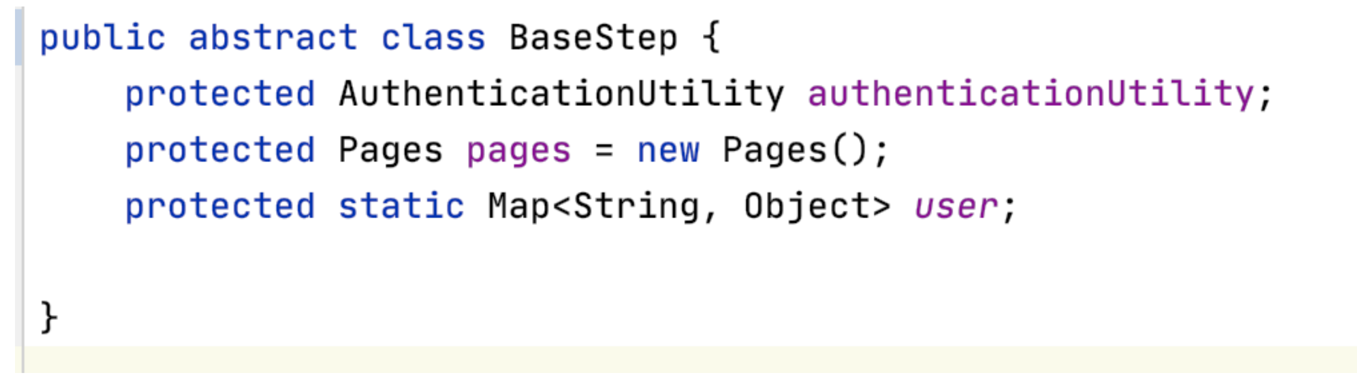
```java
public class LoginStepDefs extends BaseStep {
    @Given("I am on the login page")
    public void i_am_on_the_login_page() {
        String url = Environment.getProperty("url");
        Driver.getDriver().get(url);

    }

    @Given("I login to application as a {word}")
    public void i_login_to_application_as_a(String user) throw
        String email = null, password = null;
        switch (user.toLowerCase()) {
            case LibraryConstants.LIBRARIAN:
                email = Environment.getProperty("librarian_ema
                password = Environment.getProperty("librarian_
                password = Encoder.decrypt(password);
                break;
            case LibraryConstants.STUDENT:
                email = Environment.getProperty("student_email
                password = Environment.getProperty("student_pa
                password = Encoder.decrypt(password);
                break;
            default:
                throw new Exception("Wrong user type is provid
        }
        pages.loginPage().login(email, password);
    }
```

HOW DID YOU USE OOP CONCEPTS IN YOUR FRAMEWORK

- Abstraction **3.10**



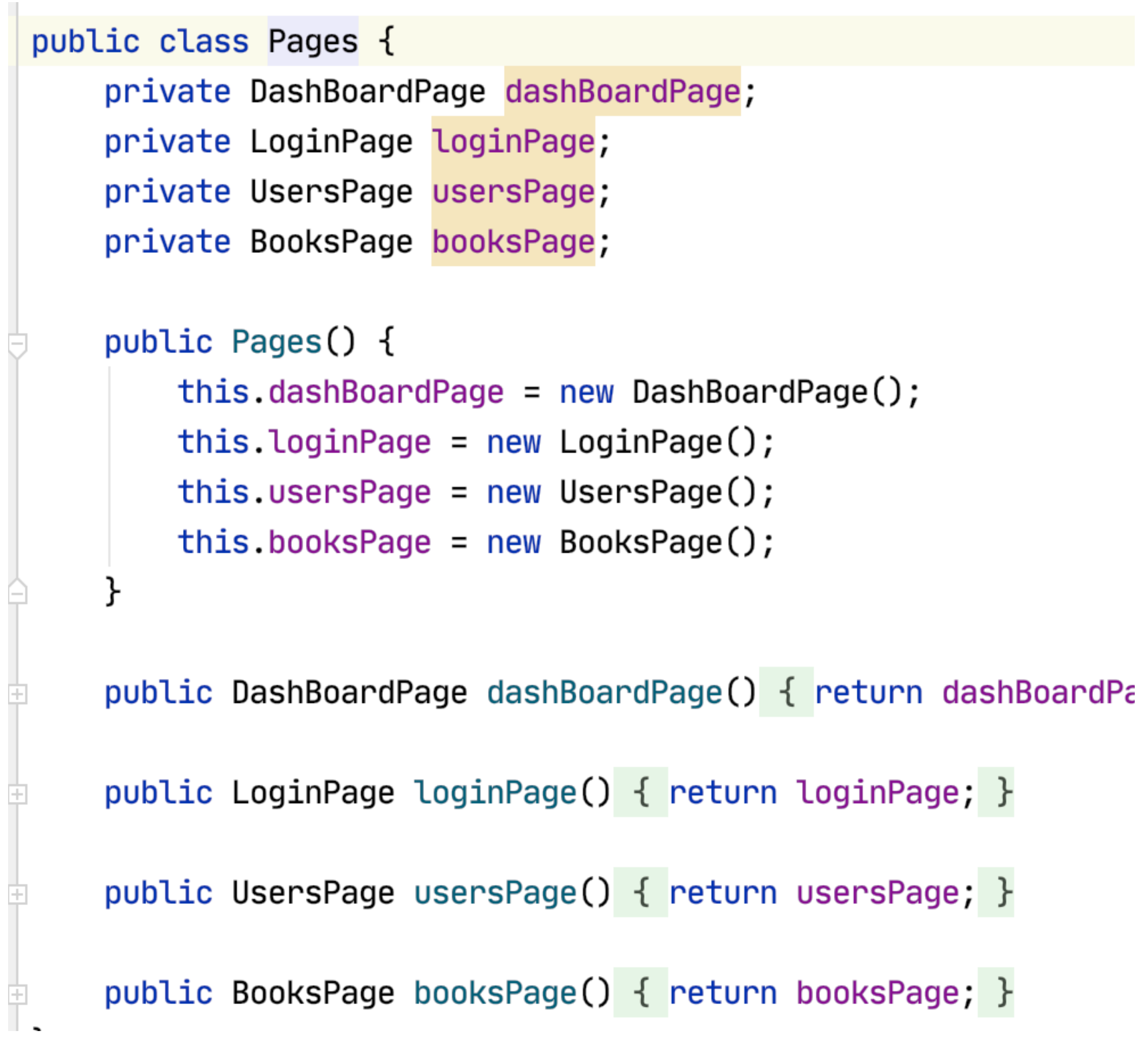
```java
public abstract class BasePage {
    public BasePage() { PageFactory.initElements(Driver.getDriver(),  page: this); }

    @FindBy(xpath = "//span[@class='title'][.='Users']")
    public WebElement users;

    @FindBy(xpath = "//span[@class='title'][.='Dashboard']")
    public WebElement dashboard;

    @FindBy(xpath = "//span[@class='title'][.='Books']")
    public WebElement books;

    @FindBy(tagName = "h3")
    public WebElement pageHeader;

    @FindBy(css = "#navbarDropdown>span")
    public WebElement accountHolderName;

    @FindBy(linkText = "Log Out")
    public WebElement logOutLink;

    public void logOut(){
        accountHolderName.click();
```



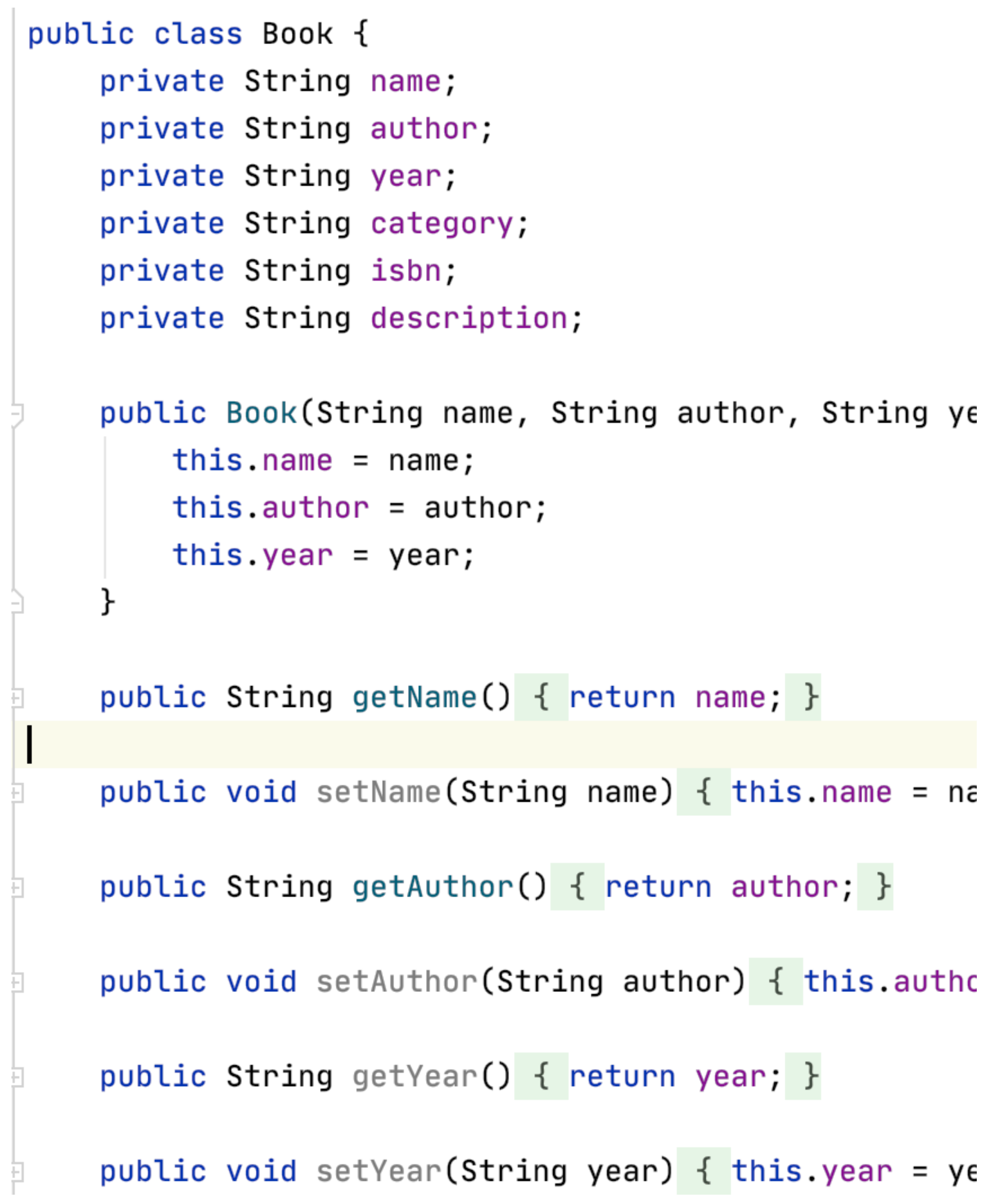
```java
public abstract class BaseStep {
    protected AuthenticationUtility authenticationUtility;
    protected Pages pages = new Pages();
    protected static Map<String, Object> user;

}
```

HOW DID YOU USE OOP CONCEPTS IN YOUR FRAMEWORK

- Encapsulation
- Pojos/beans, Pages class,
- Page object classes,
- driver class



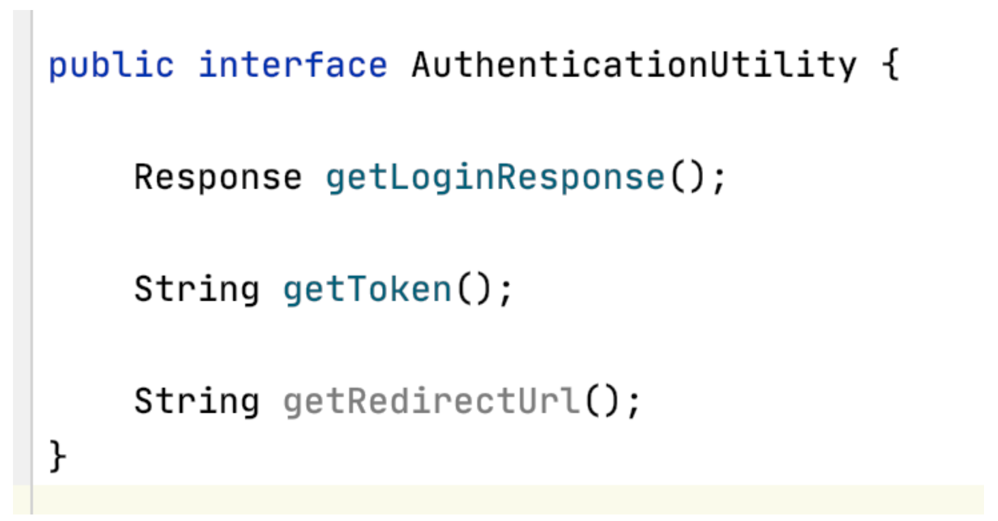
```java
public class Pages {
    private DashBoardPage dashBoardPage;
    private LoginPage loginPage;
    private UsersPage usersPage;
    private BooksPage booksPage;

    public Pages() {
        this.dashBoardPage = new DashBoardPage();
        this.loginPage = new LoginPage();
        this.usersPage = new UsersPage();
        this.booksPage = new BooksPage();
    }

    public DashBoardPage dashBoardPage() { return dashBoardPa

    public LoginPage loginPage() { return loginPage; }

    public UsersPage usersPage() { return usersPage; }

    public BooksPage booksPage() { return booksPage; }
```



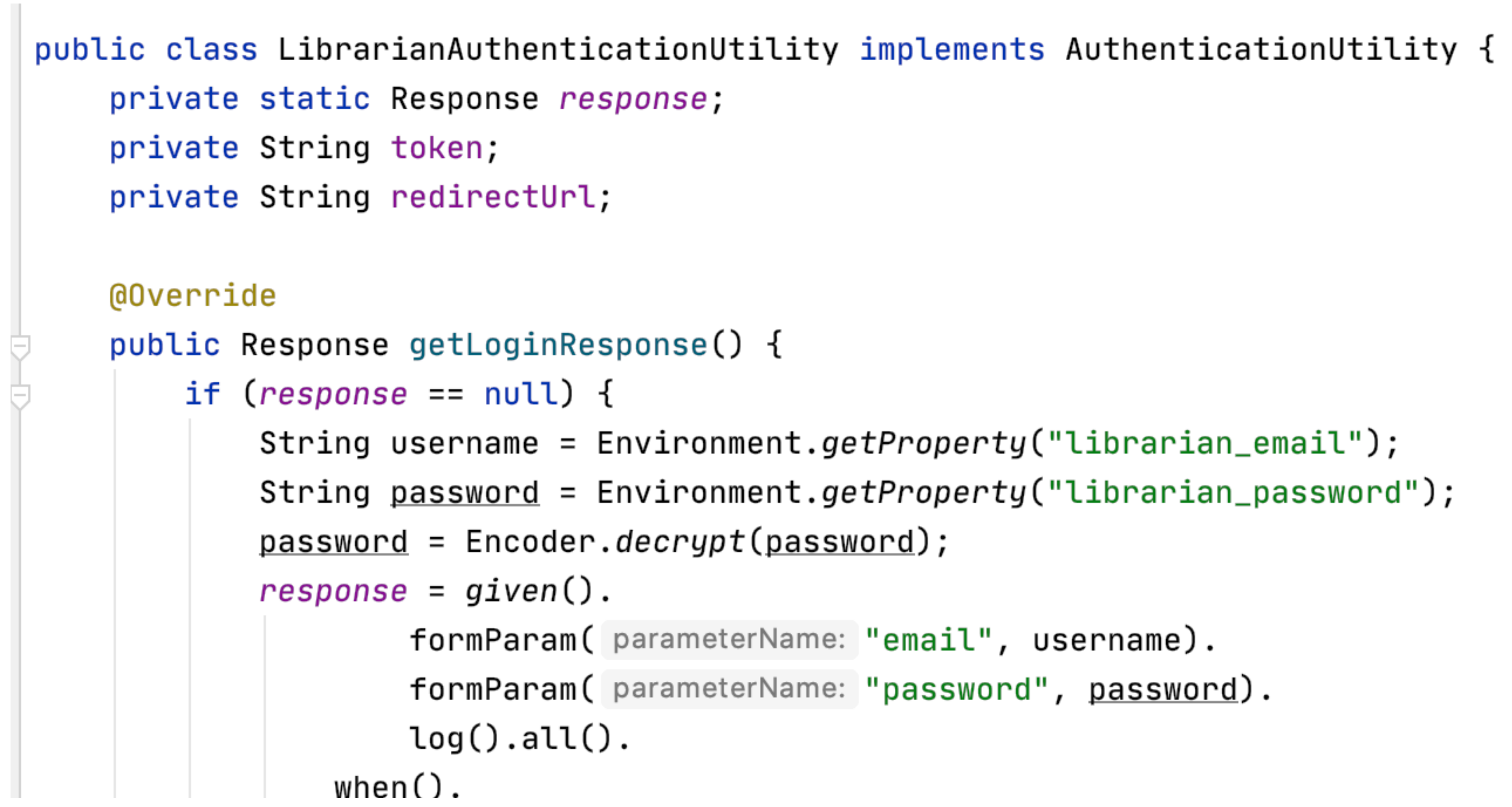
```java
public class Book {
    private String name;
    private String author;
    private String year;
    private String category;
    private String isbn;
    private String description;

    public Book(String name, String author, String ye
        this.name = name;
        this.author = author;
        this.year = year;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = na

    public String getAuthor() { return author; }

    public void setAuthor(String author) { this.autho

    public String getYear() { return year; }

    public void setYear(String year) { this.year = ye
```

HOW DID YOU USE OOP CONCEPTS IN YOUR FRAMEWORK

- Polymorphism
- I use interface for AuthenticationUtility. Of each type of user, I have different inmplementation of that interface AuthenticationUtility



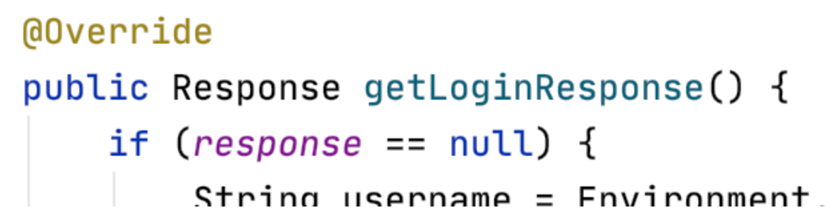
```java
public interface AuthenticationUtility {

    Response getLoginResponse();

    String getToken();

    String getRedirectUrl();
}
```



```java
public class LibrarianAuthenticationUtility implements AuthenticationUtility {
    private static Response response;
    private String token;
    private String redirectUrl;

    @Override
    public Response getLoginResponse() {
        if (response == null) {
            String username = Environment.getProperty("librarian_email");
            String password = Environment.getProperty("librarian_password");
            password = Encoder.decrypt(password);
            response = given().
                    formParam( parameterName: "email", username).
                    formParam( parameterName: "password", password).
                    log().all().
                when().
```

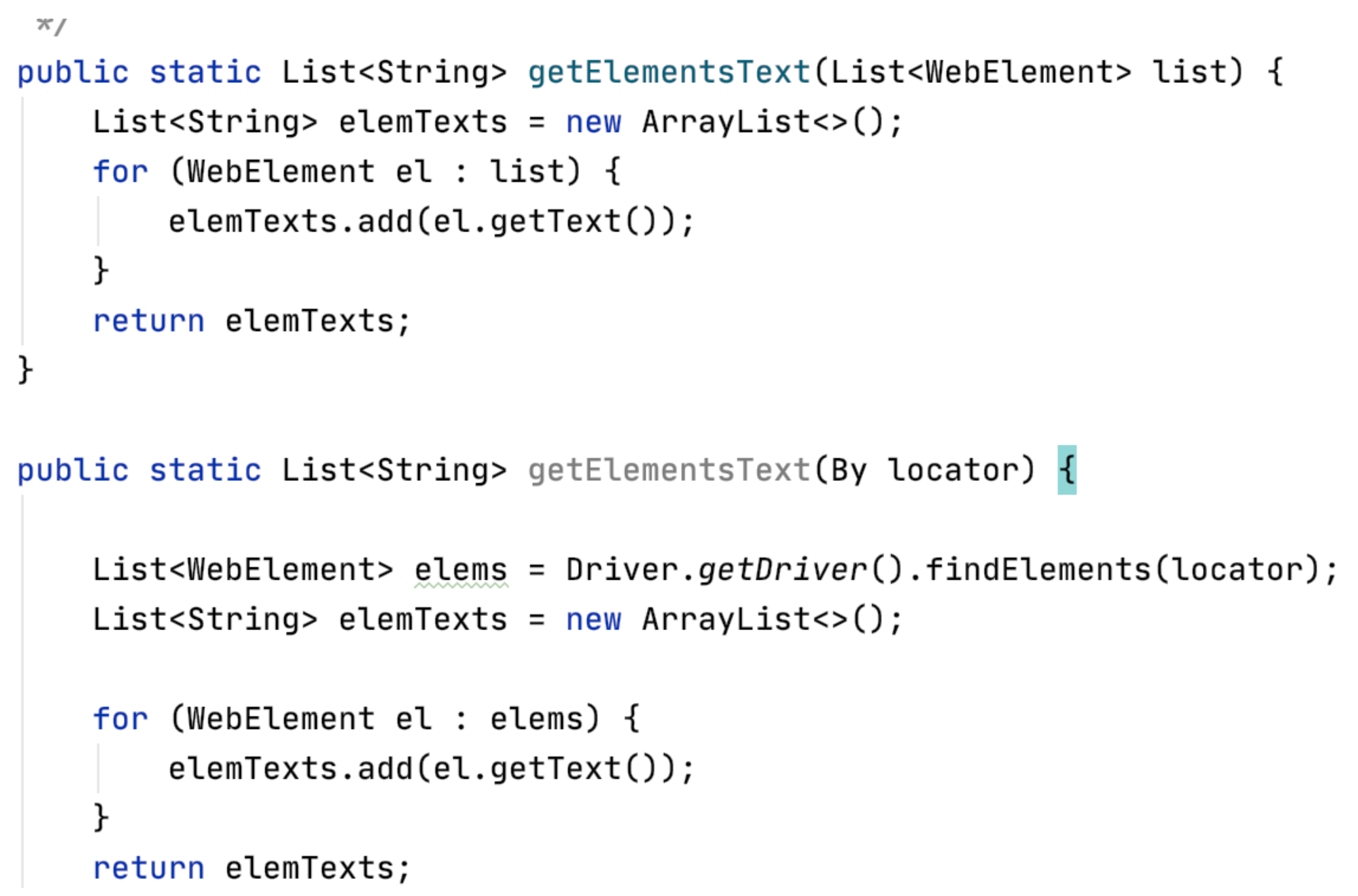
HOW DID YOU USE OOP CONCEPTS IN YOUR FRAMEWORK

- Polymorphism
- Overriding: overriding methods from the interface



```
@Override
public Response getLoginResponse() {
    if (response == null) {
        String username = Environment.
```

- Overloading: in our utilities we have methods which can take different parameters



```
 */
public static List<String> getElementsText(List<WebElement> list) {
    List<String> elemTexts = new ArrayList<>();
    for (WebElement el : list) {
        elemTexts.add(el.getText());
    }
    return elemTexts;
}

public static List<String> getElementsText(By locator) {

    List<WebElement> elems = Driver.getDriver().findElements(locator);
    List<String> elemTexts = new ArrayList<>();

    for (WebElement el : elems) {
        elemTexts.add(el.getText());
    }
    return elemTexts;
```

How to pass command line argument?

- Password in this comes from terminal. It means every time we run the test, we have to pass the password. Otherwise it will fail.

```
String password = System.getProperty("db_password");
// mvn test -Ddb_password=abc123
```

how did you handle sensitive data in your framework?

- 1. I passed the sensitive data like database username and password from terminal.
- 2. We used password encoders. We ahve the encoded version of the password saved in the properties file. In the code we decode the password before using it

Api in testing with cucumber + selenium

- 1. Have test cases that include api
- 2. To generate test data

https://cucumber.io/blog/open-source/cucumber-jvm-languages-support/