CPS 420: Software Engineering – Project 2

Fall 2024
Deadline: October 25, 2024; 11:59 PM

1 Background and Objectives

1.1 Overview

This project focuses on enhancing your skills in writing tests for Java applications and setting up Continuous Integration (CI) pipelines using Travis CI. You will be provided with a Java application that requires testing. Your task is to identify and write appropriate tests, and set up a CI pipeline that automates building, testing, and deploying the application. The final deliverable will be a GitHub repository containing your code and CI configurations.

1.2 Deliverables

- **GitHub Repository:**
  - Public repository containing your Java application, tests, and CI configuration.
  - Include a `README.md` explaining how to build and run the application.
- **Documentation:**
  - Report detailing the parts of the code you identified for testing.
  - Explanation of the tests you wrote and why they are important.
  - Overview of your CI pipeline setup.
  - Screenshots demonstrating successful builds and deployments.

1.3 Project Background

The town of **Oakwood** relies heavily on its public library, a cornerstone of the community that offers access to knowledge and resources for people of all ages. Due to increasing demands and outdated management practices, the library is in need of a modern solution to manage its operations efficiently.

As part of a collaborative initiative, you have been tasked with developing a new **Library Management System (LMS)** that will streamline the library's functions, improve accuracy in tracking books and memberships, and enhance the overall user experience for both librarians and patrons.

The LMS will cover key functionalities such as:

- **Book Management:** Cataloging new books, updating existing records, and managing inventory levels.

- **Member Management:** Registering new members, maintaining member information, and enforcing borrowing limits.
- **Loan Management:** Recording the borrowing and returning of books, calculating due dates, and handling overdue notifications.
- **Search Capabilities:** Enabling efficient search of the library's catalog by various criteria such as title, author, or ISBN.

1.4 Code Explanation

1. Book.java: The Book class represents a book in the library with attributes like ISBN, title, author, and the number of copies available. It includes methods to borrow and return books, along with input validation.
   - The constructor and setters include input validation to ensure data integrity.
   - The borrowBook() method decrements copiesAvailable and throws an exception if no copies are left.
   - The returnBook() method increments copiesAvailable.
   - The toString() method provides a readable representation of the book.
2. Member.java: The Member class represents a library member with attributes like member ID, name, and a list of borrowed books. It includes methods to borrow and return books, enforcing a loan limit.
   - The member has a loan limit defined by MAX_LOAN_LIMIT.
   - The borrowBook() method allows a member to borrow a book if they haven't reached the limit.
   - The returnBook() method processes the return and updates the member's borrowed books list.
   - The member maintains a list of Loan objects representing their borrowed books.
3. Loan.java: The Loan class represents a loan transaction between a member and a book. It includes the loan date, due date, and whether the book has been returned.
   - Each loan has a loanDate and a dueDate calculated based on the loan period.
   - The isReturned flag indicates whether the book has been returned.
   - The isOverdue() method checks if the current date is past the due date.
4. LibrarySystem.java: The LibrarySystem class manages the overall operations of the library, including adding books and members, searching, and handling loans.
   - The library system maintains lists of books, members, and loans.
   - Methods are provided to add and remove books and members.
   - Search functions are implemented for books and members.
   - Loan issuance and return are handled with appropriate validations.
   - The system can retrieve overdue loans for notifications.
5. Main.java: The Main class demonstrates how the Library Management System can be used.

- The Main class creates an instance of LibrarySystem and performs various operations.
- It adds books and registers members.
- Loans are issued to members for specific books.
- The current state of books, members, and loans is displayed.
- A book is returned, and overdue loans are checked.

1.5 Project Directory Structure

The project should be organized in the following directory structure:

```
LibraryManagementSystem/
├── .travis.yml ........................ CI configuration file
├── build.xml .......................... Apache Ant build script
└── src/ ............................... Source code directory
    └── com/
        └── example/
            └── library/ ............... Package containing library classes
                ├── Book.java
                ├── Loan.java
                ├── LibrarySystem.java
                ├── Main.java
                ├── Member.java
                └── tests/ ............. Directory for test classes
                    ├── BookTest.java
                    ├── LoanTest.java
                    ├── LibrarySystemTest.java
                    └── MemberTest.java
```

- **Root Directory (`LibraryManagementSystem/`):**
  - Contains the CI configuration file `.travis.yml` and the Ant build script `build.xml`.
- **Source Directory (`src/`):**
  - Holds all your Java source files organized into packages.
  - The package structure follows the standard Java naming convention.
- **Testing Directory (`tests/`):**
  - Located inside the `library` package.
  - Contains all your test classes, each corresponding to a class in your main application.
- **Continuous Integration File (`.travis.yml`):**
  - Placed at the root directory.
  - Configures Travis CI to build, test, and deploy your application.
- **Apache Ant Build Script (`build.xml`):**
  - Also placed at the root directory.
  - Automates the build process, including compiling source code and running tests.

2 Instructions

1. **Setup Java Application and Create Tests [Points: 20]**
   (a) **Set Up the Java Application**
   - The Java application source code is provided as an attachment. The application simulates a library management system with various functionalities.
   - Download the code and get familiarized with the code base.

   (b) **Identify Parts to Test**
   - **Analyze Critical Components:**
     - Identify classes and methods that are critical to the application's functionality.
     - Focus on areas involving data manipulation, business logic, and user interactions.
   - **Determine Testing Strategies:**
     - Decide which testing techniques are appropriate (e.g., unit testing, integration testing).
     - Consider edge cases and input validation scenarios.

   (c) **Write Tests Using Asserts:**
   - Implement your tests using `assert` statements in Java.
   - Use the sample values provided to test different scenarios.
   - Make sure you follow the naming conventions for writing tests.

   (d) **What tests to write? How many to write?**
   - You have the freedom to write any tests.
   - You need to write 10 tests, covering a wide range of test cases.

2. **Set Up Apache Ant Build Script [Points: 10]**
   - **Create a Build Script:**
     - Write a `build.xml` file to automate the compilation and packaging of your application.
     - Define targets for cleaning, compiling, testing, and packaging your code.

3. **Configure Travis CI [Points: 10]**
   - **Sign Up and Integrate with GitHub:**
     - Create an account on Travis CI and sync your GitHub repository.
     - Enable Travis CI for your repository.
   - **Create a CI Configuration File:**
     - Add a `.travis.yml` file to your repository with the necessary configurations.
     - Configure the pipeline to build, test, and deploy your application.
     - You can also run this with Github Actions if you prefer. In that case, you do not need to run the CI pipeline using Travis CI.

4. **Automate Deployment [Points: 10]**
   - **Set Up Deployment Steps:**
     - Configure the pipeline to build your application.
     - Automate the release process by publishing builds to GitHub Releases.

3 Submission

- **GitHub Link:**
  - Ensure your repository is public and put the link on the report.
- **Report:**
  - Explain the testing strategies you employed.
  - Provide an overview of the build process and how it works.
  - Provide an overview of your CI pipeline and how it works.
  - Include any challenges faced and how you overcame them.
  - Don't forget to include you github repository on the report.
  - Submit the report as a doc file on Canvas.