Software Engineering Group
Prof. Dr. Sebastian Baltes
Brenda Chiteri
Andrei Costeniuc, Xenia Herr,
Lukas Moritz, Xel Pratscher

Universität Heidelberg
Sommersemester 2025

Software Engineering (ISE)

Übungsblatt 3

Abgabe in Zweiergruppen erlaubt

Aufgabe 3.1 (Branching, Merging und Konfliktlösung: 1,5 Punkte)

Diese Aufgabe bezieht sich auf das Repository des vorherigen Übungsblattes:
`https://github.com/se-uhd/ise25-26_assignment02.git`
This exercise refers to the repository used for the previous assignment:
`https://github.com/se-uhd/ise25-26_assignment02.git`
Zwei lokale Entwicklungszweige `feature/is-empty-method` und `feature/null-check` haben denselben Commit als direkten Vorfahren. In beiden Zweigen wurde die Methode `isEmpty()` in der Datei `DoublyLinkedList.java` geändert:
Two local branches `feature/is-empty-method` and `feature/null-check` were branched from the same commit. In both branches, a commit modified the method `isEmpty()` in the file `DoublyLinkedList.java`:

- In `feature/is-empty-method` wurde die Zeile wie folgt geändert: `return length == 0;`
  In `feature/is-empty-method`, the line was changed to: `return length == 0;`
- In `feature/null-check` wurde die Zeile wie folgt geändert: `return begin == null;`
  In `feature/null-check`, the line was changed to: `return begin == null;`

Zuerst wird der Branch `feature/is-empty-method` in `main` merged. Beim Mergen des Branches `feature/null-check` tritt ein Konflikt auf.
First, the branch `feature/is-empty-method` is merged into `main`. When merging other branch `feature/null-check`, a conflict occurs.

Ihre Aufgaben sind:
Your tasks are:

(a) **Erklären** Sie kurz, warum der Merge-Konflikt auftritt und wie Git den Konflikt in der Datei markiert.
Briefly explain why the merge conflict occurs and how Git marks the conflict in the file.
(0,5 Punkte)

(b) **Geben** Sie die in Stichworten die notwendigen Aktionen an, um den Konflikt zu aufzulösen und den Merge abzuschließen.
Provide bullet points outlining the actions required to resolve the conflict and complete the merge.
(0,5 Punkte)

(c) **Nennen** Sie eine Art von Konflikt, die mit Git alleine nicht gefunden werden kann.
Name of kind of conflict that cannot be found using Git alone.
(0,5 Punkte)

Aufgabe 3.2 (Buildautomatisierung und CI: 3,5 Punkte)

Forken Sie das auf GitHub verfügbare Repository und klonen Sie Ihren Fork in ein Verzeichnis Ihrer Wahl. Sie benötigen dafür ein kostenloses GitHub-Konto:
`https://github.com/se-uhd/ise25-26_assignment03`
Ihre Abgabe für diese Übung ist ein Link zu Ihrem Fork des Assignment-Repositories. Alternativ können Sie das Repository klonen und beispielsweise mit GitLab arbeiten. Beachten Sie dazu die Hinweise auf der letzten Seite.
Fork the repository available on GitHub and clone your fork into a directory of your choice. You will need a free GitHub account to do this:
`https://github.com/se-uhd/ise25-26_assignment03`
Your submission for this exercise is a link to your fork of the assignment repository. Alternatively, you can clone the repository and work with GitLab. See corresponding remarks on the last page.

Entwicklungsumgebung einrichten: Installieren Sie die benötigten Abhängigkeiten. Sie können die im Repository bereitgestellte Datei `mise.toml` verwenden, um die Installation zu automatisieren (siehe mise Dokumentation für Details). Alternativ können Sie die in der Datei deklarierten Abhängigkeiten manuell installieren. Sie benötigen auch Docker (oder eine kompatible Open-Source-Alternative), um die Anwendung zu bauen und auszuführen. Installieren Sie eine Java IDE. Wir empfehlen IntelliJ IDEA, Sie können aber auch andere Editoren wie Visual Studio Code mit geeigneten Erweiterungen verwenden. Beachten Sie jedoch, dass unsere Beispiele und Anweisungen auf IntelliJ IDEA basieren. Es gibt verschiedene Plugins, um `mise` innerhalb von IDEs zu verwenden. Importieren Sie Ihren Fork über `Datei → Neu → Projekt aus bestehenden Quellen`. Wählen Sie die `pom.xml`-Datei in Ihrem geklonten Repository aus und öffnen Sie sie, um das Projekt einzurichten. Stellen Sie sicher, dass die korrekte Maven-Version in den IDE-Einstellungen konfiguriert ist. Öffnen Sie schließlich das integrierte Terminal in Ihrer IDE und führen Sie den folgenden Befehl aus, um das Projekt zu bauen: `mvn clean install`. Alternativ können Sie die IntelliJ Maven-Integration verwenden. Sie werden feststellen, dass der Build fehlschlägt.
Setting up your development environment: You can use the `mise.toml` file provided in the repository to automate the installation (see mise documentation for details). Alternatively, you can manually install the declared dependencies. You also need Docker (or a compatible open-source solution) to build and run the application. Install a Java IDE. We recommend IntelliJ IDEA, but you can also use other editors such as Visual Studio Code with suitable extensions. Note, however, that our examples and instructions are based on IntelliJ IDEA. There are different plugins to use `mise` within IDEs. Import your fork via `File → New → Project from Existing Sources`. Select and open the `pom.xml` file in your cloned repository to set up the project. Ensure that the correct Maven version is configured in the IDE settings. Finally, open the integrated terminal in your IDE and run the following command to build the project: `mvn clean install`. Alternatively, you can use the IntelliJ Maven integration. You will notice that the build is failing.

Ihre Aufgaben sind:
Your tasks are:

(a) **Erstellen Sie einen neuen Branch `feature/fix-test` und checken Sie ihn aus.**
Create a new branch `feature/fix-test` and check it out.

(0,5 Punkte)

(b) **Beheben Sie den fehlgeschlagenen Test.**
Fix the failing test.
Die Maven-Ausgabe gibt Hinweise darauf, welcher Test fehlschlägt. Committen Sie Ihre Änderungen in den Branch, den Sie zuvor erstellt haben.
The Maven output provides hints about which test is failing. Commit your changes to the branch you created earlier.

(1 Punkt)

(c) **Testen Sie die REST-API mit `curl`.**
Test the REST-API using `curl`.
Ihre Aufgabe ist es, einen neuen Point-of-Sales (POS) mit einem `POST`-Request an die REST API zu senden, und dann das Ergebnis mit einer `GET`-Anforderung zu überprüfen.
Your task is to create a new point of sales (POS) by sending a `POST` request to the REST API and then verify the result with a `GET` request.

- Starten Sie die Anwendung wie in der `README`-Datei beschrieben.
  Start the application as described in the `README` file.
- Besuchen Sie den folgenden Eintrag auf Google Maps, um die erforderlichen Details der `Bäckerei Kohlmann` zu extrahieren.
  Visit the following entry on Google Maps to extract the required details for the `Kohlmann bakery`.
- Verwenden Sie diese Informationen, um eine gültige JSON-Nutzlast zu erstellen und senden Sie sie über `curl` mit einem `POST`-Request an den entsprechenden Endpunkt.
  Use that information to create a valid JSON payload and send it via `curl` using a `POST` request to.
- Verwenden Sie dann erneut `curl`, um einen `GET`-Request zu senden und zu überprüfen, ob Ihr Eintrag erfolgreich gespeichert wurde.
  Then, use `curl` send a `GET` request and confirm that your entry was successfully saved.
- Dokumentieren Sie die von Ihnen verwendeten Befehle in der `README`-Datei und committen Sie Ihre Änderungen in den Branch, den Sie zuvor erstellt haben.
  Document the command you have used in the `README` file and commit your changes to the branch you created earlier.

(1 Punkt)

(d) **Ändern Sie den GitHub Actions Workflow wie folgt ab.**
Modify the GitHub Actions workflow as follows. Öffnen Sie die GitHub Actions Workflow-Datei (`build.yml`) und nehmen Sie die folgenden Änderungen vor:
Open the GitHub Actions workflow file (`build.yml`) and make the following changes:

- Aktualisieren Sie den Trigger, sodass der Workflow bei jedem `push` auf Branches ausgeführt wird, die mit `feature/` beginnen.
  Update the trigger so the workflow runs on any `push` to branches that start with `feature/`
- Committen Sie Ihre Änderungen in den Branch, den Sie zuvor erstellt haben.
  Commit your changes to the branch that you created earlier.

Ändern Sie keinen anderen Teil der Datei.
Do not modify any other part of the file.

(1 Punkt)

Nachdem Sie die Aufgaben bearbeitet haben:
After completing the tasks:

- Pushen Sie Ihre Änderungen in einen neuen Remote-Branch `feature/fix-tests`.
  Push your changes to a new remote branch `feature/fix-tests`.
- Öffnen Sie die Registerkarte **Actions** in Ihrem geforkten Repository und bestätigen Sie, dass der Workflow erfolgreich läuft.
  Open the **Actions** tab in your forked repository and confirm that the workflow runs successfully.

Einreichungsfrist:

- Freitag, 07.11.2025, 12:00 Uhr

Nächste Übungsgruppen:

- Dienstag, 11.11.2025
- Mittwoch, 12.11.2025
- Donnerstag, 13.11.2025

Hinweise zur Abgabe:

- Wir verwenden das Format dd.mm.yyyy für **Datumsangaben** und das 24-Stunden-Format für **Uhrzeiten**.
- Fassen Sie alle Texte, Diagramme und andere Bestandteile Ihrer Abgabe in **einer PDF-Datei** zusammen.
- Die erste Seite der PDF-Datei muss Ihren **Namen** und Ihre **Matrikelnummer** enthalten; bei Gruppenabgaben die Namen und Matrikelnummern beider Gruppenmitglieder. Ob eine **Einzel- oder Gruppenabgabe** gefordert ist, ist oben auf dem Übungsblatt vermerkt.
- Die Abgabe von **Plagiaten** führt zur Bewertung mit **0 Punkten** für das gesamte Übungsblatt. Bei **Verdacht** der Verwendung von **GenAI-Tools** ohne Quellenangabe wird die Abgabe ebenfalls mit **0 Punkten** bewertet.
- Bei **UML-Aufgaben**, reichen Sie Ihre **Diagramme** als Teil der **PDF-Datei** ein. Optional können Sie Ihren **PlantUML-Quellcode** reinreichen.
- Bei **Programmieraufgaben**, reichen Sie Ihren Quellcode oder Ihre Konfigurationsdateien als Textdateien mit entsprechender Dateiendung ein (z.B. `.java` für Java-Quellcode oder `.yaml` für YAML-Konfigurationsdateien). Reichen Sie keine kompilierten Dateien ein (z.B. `.class` in Java). Für Aufgaben, die das Forken eines GitHub-Repositories erfordern, ist der Standardweg zur Einreichung Ihrer Lösung, dass Sie einen **Link zu Ihrem Fork** in die oben genannte PDF-Datei einfügen.
- Falls Sie **Windows** verwenden, empfehlen wir dringend, für die Bearbeitung der technischen Aufgaben **WSL** zu verwenden. Wir können unseren Code nicht unter Windows testen. Wenn Sie Probleme feststellen, informieren Sie uns bitte.
- Falls Sie kein **GitHub-Konto** anlegen möchten, können Sie Aufgaben, für die ein GitHub-Fork erforderlich ist, auch einreichen, indem Sie einen Link zu einer Kopie des Git-Repositories mit Ihren Lösungen in einem anderen **öffentlich oder universitätsintern gehosteten Git-Repository** angeben (Aufgaben-Repository klonen, Änderungen implementieren, in das Repository pushen, Link teilen). Beachten Sie jedoch, dass Sie damit die CI-bezogenen Aufgaben nicht bearbeiten können. Es steht Ihnen frei, GitLab als Alternative zu verwenden und die GitHub-Actions-Aufgaben an GitLab CI anzupassen. Für diese alternativen Vorgehensweisen können wir jedoch **keine Unterstützung** anbieten.
- Setzen Sie sich frühzeitig mit den benötigten **Abhängigkeiten und Tools** und deren Installation auseinander. Wir stellen für die Übungsblätter `mise.toml`-Dateien bereit, die die Installation von Abhängigkeiten vereinfachen (siehe Dokumentation von **mise**). Sie können die Abhängigkeiten auch manuell installieren, was wir aber nicht empfehlen. Sie werden auch eine lokale **Docker**-Installation benötigen. Alternativ können Sie Rancher Desktop installieren. Als Entwicklungsumgebung (IDE) empfehlen wir **IntelliJ IDEA**. Sie können auch andere Editoren wie z.B. Visual Studio Code mit entsprechenden Erweiterungen verwenden, wofür wir aber keine Unterstützung anbieten können. Alle Beispiele und Anleitungen basieren auf IntelliJ. Für mise gibt es verschiedene **IDE-plugins**.