

Software Lifecycle Management

Final Project

Java Game TicTacToe

by Eva Lehner, Florian Scholz, Sophie Schönwälde-Rieder

February, 2026

Table of Contents

- [000 - Welcome](#)
- [001 - Create a GitHub repository_\(public\)](#)
- [002 - Create a GitHub project and link it to the created repository](#)
- [003 - Transfer the user stories to a new Kanban board in the project](#)
- [004 - Add additional issues to the kanban board](#)
- [005 - Create a new Java Maven project](#)
- [006 - Set the GitHub repository as a remote repository](#)
- [007 - Commit and push your blank Java project to the main branch](#)
- [008 - Protect the main branch](#)
- [009 - CI_CD Scripts and workflows](#)
- [010 - Working on User Stories](#)
- [011 - A final overview](#)

000 - Welcome

This document serves as documentation for the SLM-project 'Java Game TicTacToe' and provides a walkthrough illustrating the most important steps of our development process with short textual description as well as screenshots - focussing on our usage of GitHub and application of concepts studied in the seminar rather than on the game itself.

The structure of the documentation follows the chronology of our workflow - which in turn closely followed the list of tasks provided in the task description.

Regarding our collaborative process: Especially in the earlier stages of the project, the majority of tasks were done together, triple programming style. The intent was to maximise knowledge gain for all team members.

This is mentioned here to prevent faulty impressions that might be caused by the statistics showing that one member of the team authored significantly more commits than the other two members in the beginning. This is due to the fact that their account was typically used for the tasks we tackled together, mainly to simplify the workflow and keep the screenshots similar.

The GitHub repository can be accessed at <https://github.com/ww24e018/slm-tictactoe>

The project (Kanban Board) can be accessed at

<https://github.com/users/ww24e018/projects/1>

The .jar of our latest release of the game can be downloaded at

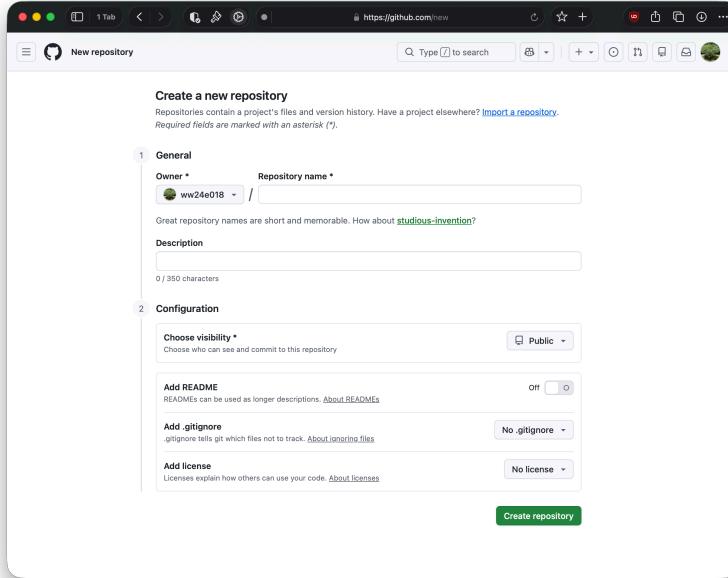
<https://github.com/ww24e018/slm-tictactoe/releases>

In the process of exploring workflows and ci/cd, a test repository was created to provide a playground for quick trials that require changes to be present on the main branch. This repo can be accessed here, should it be of any interest: <https://github.com/ww24e018/slm-tictactoe-test>

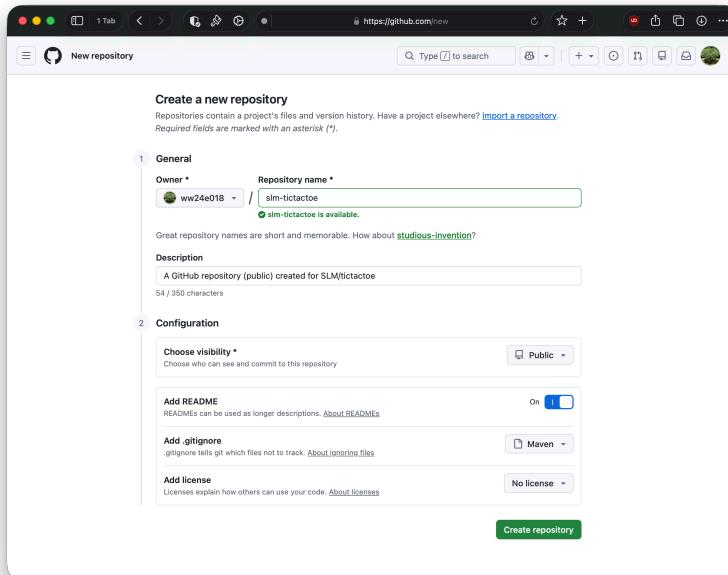
Contributors: Eva Lehner, Florian Scholz, Sophie Schönwälde-Rieder

001 - Create a GitHub repository (public)

At <https://github.com/repos> a new repository was created by clicking "new repository":



The name "slm-tictactoe" was chosen.



The repository was successfully created.

The screenshot shows a GitHub repository page for 'slm-tictactoe'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation is a search bar and a 'Code' dropdown menu. The main content area displays the repository's structure: a 'main' branch with 1 branch and 0 tags, a file .gitignore with an 'Initial commit' at 'now', and a file README.md with an 'Initial commit' at 'now'. A 'README' section follows, containing the text 'slm-tictactoe' and 'A GitHub repository (public) created for SLM/tictactoe'. To the right, there's an 'About' section with details about the repository being a GitHub repository (public) created for SLM/tictactoe, and sections for Releases (no releases published), Activity (0 stars, 0 watching, 0 forks), and Packages (no packages published). At the bottom, there's a footer with copyright information and links to GitHub's Terms, Privacy, Security, Status, Community, Docs, Contact, and Manage cookies.

In order to successfully collaborate in this new repository, the other team members were invited as collaborators:

Go to "Settings":

This screenshot is identical to the one above, but the 'Settings' tab is highlighted in the navigation bar, indicating it is the active section. The rest of the interface and repository details remain the same.

Go to "Collaborators":

The screenshot shows the 'General' tab selected in the GitHub settings interface. The left sidebar lists 'Access', 'Collaborators', and 'Moderation options'. The main content area is titled 'Code and automation' and contains a 'Code' section with a 'Copy' button and a 'Automation' section with a 'Copy' button. On the right side, there are three vertical icons: a gear for settings, a person for users, and a checkmark for automation.

The screenshot shows the 'Access' section of the GitHub repository settings for 'slim-tictactoe'. The sidebar on the left lists various settings categories like General, Access, Code and automation, Security, and Integrations. Under the 'Access' heading, the 'Collaborators' tab is selected. A message states 'Public repository' and 'This repository is public and visible to anyone'. Below it, under 'Direct access', it says '0 collaborators have access to this repository. Only you can contribute to this repository.' A large central box titled 'Manage access' contains the message 'You haven't invited any collaborators yet' and a 'Add people' button.

This screenshot is identical to the one above, but with a modal window overlaid. The modal is titled 'Add people to slim-tictactoe' and contains a search bar with 'Q Find people' and a green 'Add to repository' button. The background of the main interface is dimmed.

The screenshot shows the 'Pending invite' section of the GitHub interface. At the top, there's a search bar with 'Type' and a placeholder 'Find a colla'. Below the search bar, the text 'Pending invite' is followed by a small square icon with a minus sign. This pattern repeats twice more, creating a list of pending invites.

< Previous Next >

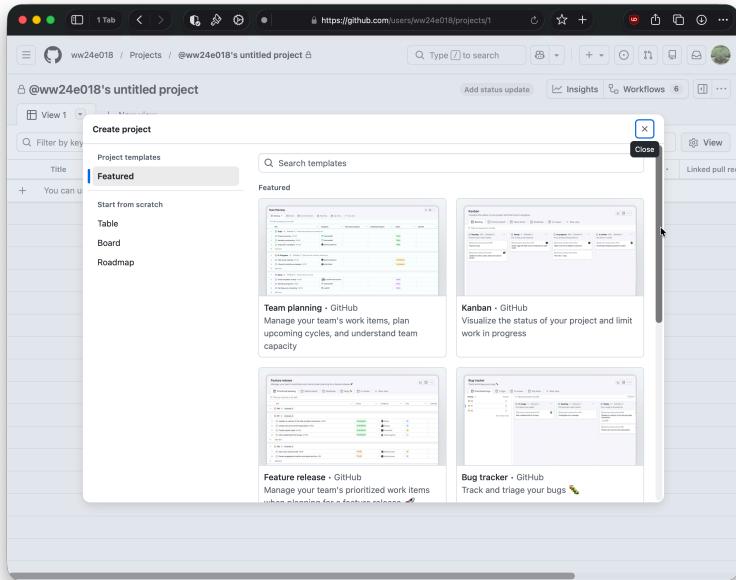
002 - Create a GitHub project and link it to the created repository

A new project was created by choosing the tab "Projects":

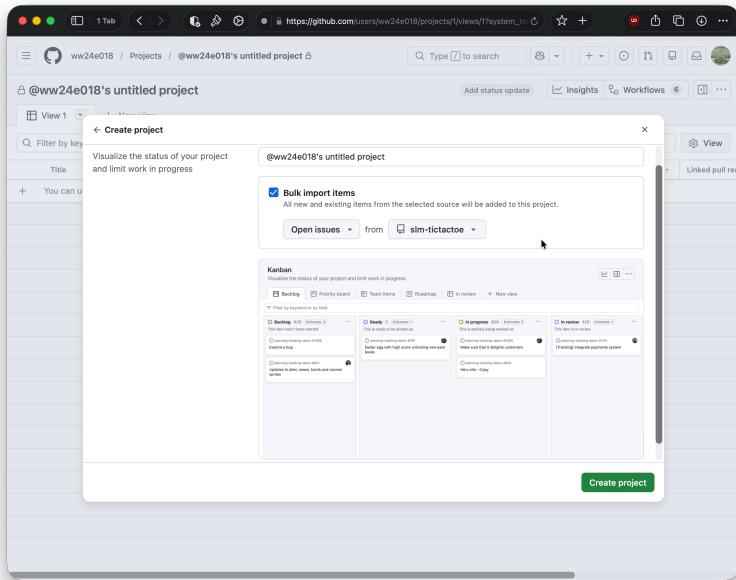
The screenshot shows the GitHub repository page for 'slm-tictactoe'. The 'Code' tab is selected, displaying the main branch ('main'), one branch, and zero tags. The repository was created by 'ww24e018' with an initial commit. The README file contains the text: 'A GitHub repository (public) created for SLM/tictactoe'.

The screenshot shows the GitHub 'Projects' tab for the 'slm-tictactoe' repository. The tab is highlighted in red. The interface is described as a 'spreadsheet' where project tables can be filtered, sorted, and grouped. It includes sections for 'Welcome to projects', 'Provide quick access to relevant projects.', and buttons for 'Link a project' and 'New project'.

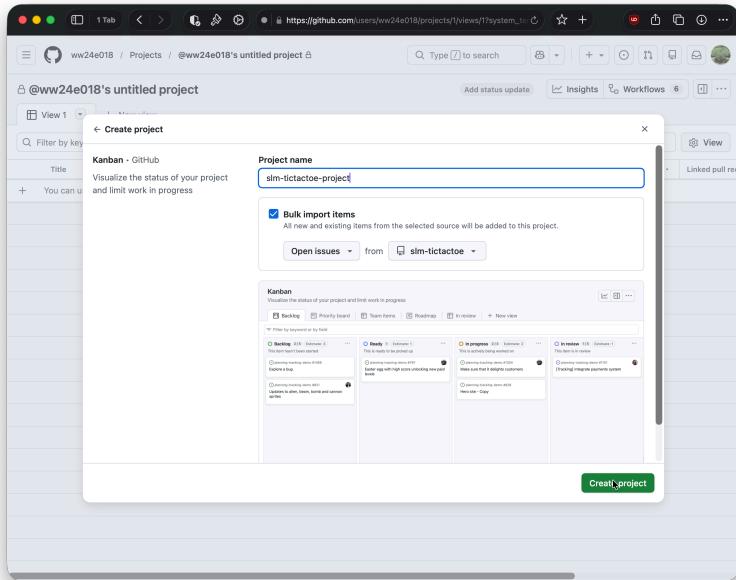
After clicking "+ New project":



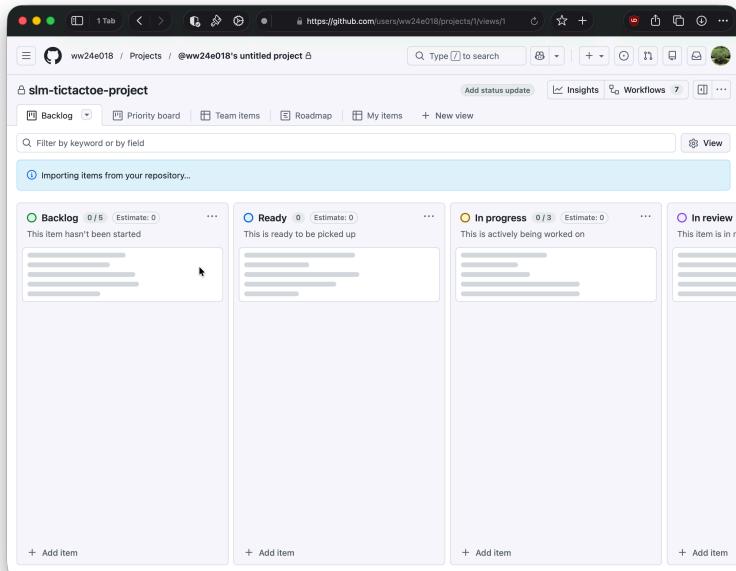
Choosing "Kanban":



Naming new project and linking it to our existing repository:



Hooray! We have a blank kanban board!



after reload:

At first we assumed that rights to the repo automatically give rights to the project linked to it. After realising that this was not the case, all team members were invited to the project as well:

clicking Settings:

Project settings

Project name: sim-tictactoe-project

Default repository: ww24e018/sim-tictactoe

Short description:

README

Save changes

Clicking "Manage access":

Who has access

Private project

Manage

Invite collaborators

Search by username

Role: Write

Manage access

1 member

Find a collaborator

ww24e018 Role: Admin Remove

Inviting via "Invite" via the process followed.

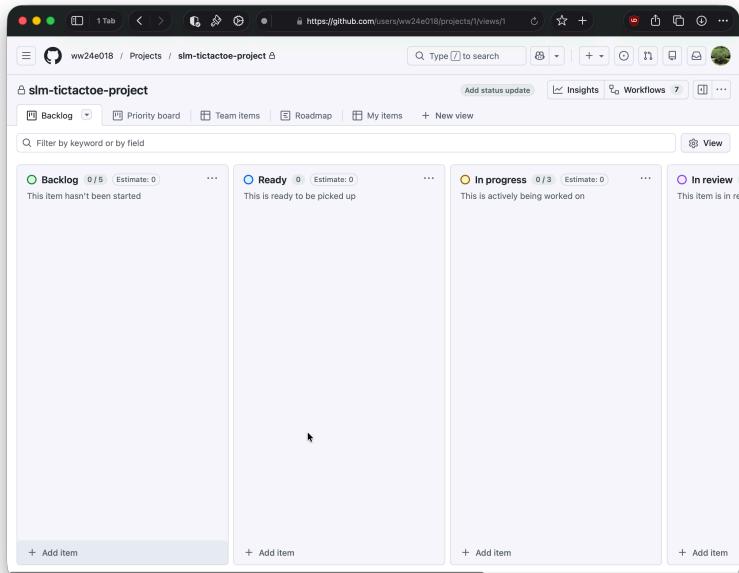
003 - Transfer the user stories to a new Kanban board in the project

The user stories in the project description are:

- As a player, I want to be able to **make a move** by choosing an empty square, so that I can place my symbol on the board. (1)
- As a player, I want to be able to **see the current state** of the game, so that I can keep track of the moves made by both myself and my opponent. (2)
- As a player, I want to **be notified when the game has ended** in a win, loss or draw, so that I can see the result of the game. (3)
- As a player, I want to be able to **start a new game** after the current game has ended, so that I can play again. (4)

The bold parts will be chosen as titles. The parts in () are the IDs given by us.

The Board:



Clicking "add item" in the backlog-column:

The screenshot shows a digital project management board with two columns: "Backlog" and "Ready". The "Backlog" column is highlighted with a blue header bar. It contains the text "This item hasn't been started". The "Ready" column has a blue header bar and contains the text "This is ready to be". Below the columns, there is a search bar and a text input field with placeholder text "Start typing to create an item, or type # to select a repository".

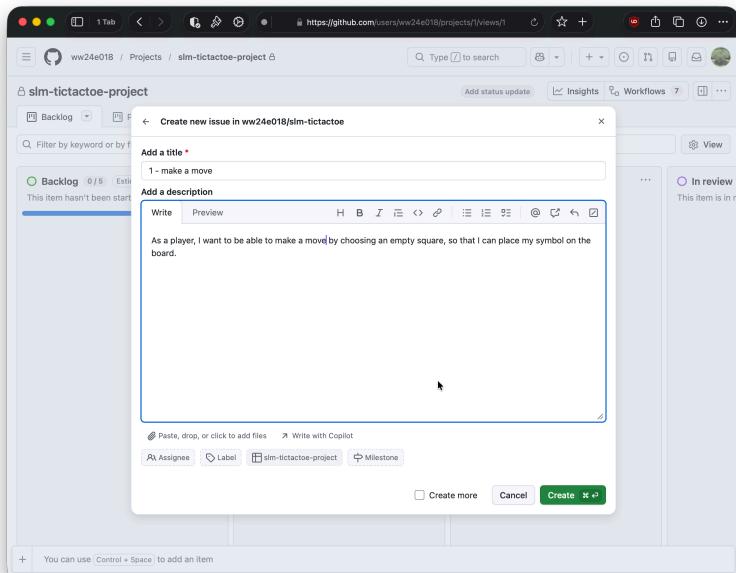
Clicking "+" is possible:

The screenshot shows a digital project management board. A large blue "+" button is prominently displayed, with the text "Create new item or add existing item" above it. Below the button, there is a text input field with placeholder text "Start typing to create an item, or type # to select a repository".

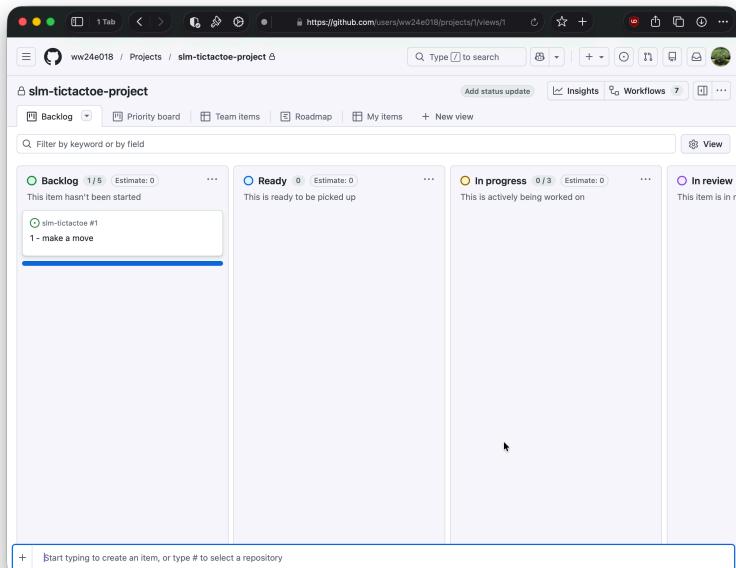
create new issue

The screenshot shows a web browser window for GitHub, displaying a project titled "slim-tictactoe-project". A modal dialog box is open, titled "Create new issue in ww24e018/slim-tictactoe". The dialog has fields for "Add a title" (with "Title" entered) and "Add a description" (with "Type your description here..."). At the bottom of the dialog are buttons for "Create more", "Cancel", and a large green "Create" button.

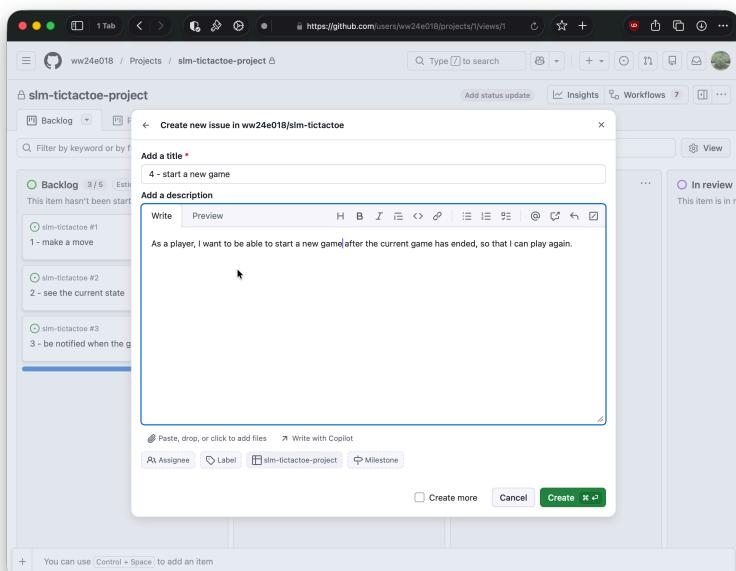
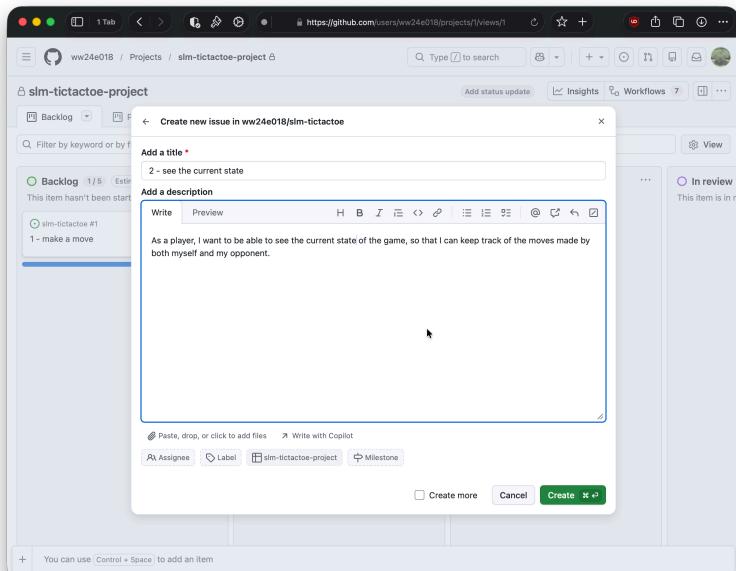
Filling in title and description:



The first issue has been created:



Screenshots of 2 and 4 and the final state:



The board:

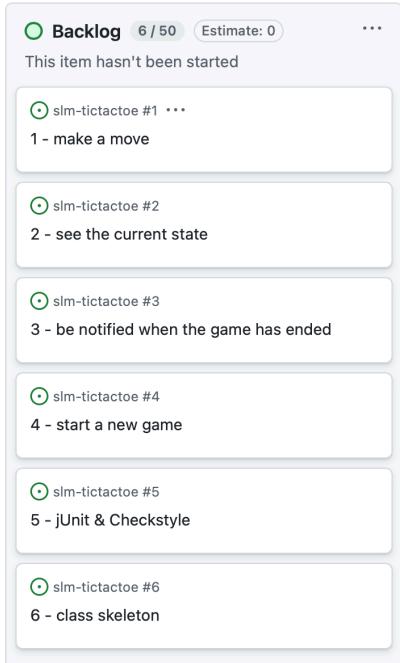
The screenshot shows a project management interface with the following details:

- Project Header:** ww24e018 / Projects / **slm-tictactoe-project**
- Navigation:** Backlog (selected), Priority board, Team items, Roadmap, My it
- Search Bar:** Filter by keyword or by field
- Backlog Section:** 4 / 5 Estimate: 0
- Ready Section:** 0 Estimate: 0
- Items in Backlog:**
 - slm-tictactoe #1: 1 - make a move
 - slm-tictactoe #2: 2 - see the current state
 - slm-tictactoe #3: 3 - be notified when the game has ended
 - slm-tictactoe #4: 4 - start a new game
- Items in Ready:** This is ready to be picked up

004 - Add additional issues to the kanban board

Additional issues were created to enable branch-naming for setup-tasks.

It was discovered there is a default WIP-Limit for the backlog column. It was at first randomly changed to 50 and later cleared altogether as we did not feel a necessity for a WIP limit in the backlog section in the context of this project.



The following items were added:

5 - jUnit & Checkstyle #5

Open

soschoen opened 6 minutes ago

As a developer I want to use jUnit for unit tests and checkstyle to check adherence to formal style conventions.

Assignees
No one - [Assign yourself](#)

Labels
None

Projects
[slm-tictactoe-project](#)

Status
Backlog

Milestone
None

Relationships
None yet

Development
Code with agent mode

Notifications
Customize
Subscribe

Add a comment

Write Preview [Create a branch](#) for this issue or link a pull request.

Use Markdown to format your comment

Paste, drop, or click to add files

[Close issue](#) [Comment](#)

6 - class skeleton #6

soschoen opened 4 minutes ago

As a developer I want the classes from the class diagram to exist before we start working on individual user stories.

Assignees: No one - [Assign yourself](#)

Labels: No labels

Projects: slm-tictactoe-project (Status: Backlog)

Milestone: No milestone

Relationships: None yet

Development: [Code with agent mode](#)

[Create a branch](#) for this issue or link a pull request.

Notifications: Subscribed

Issue #6 was further divided into 3 subissues:

soschoen opened last week

As a developer I want the classes from the class diagram to exist before we start working on individual user stories.

Sub-issues (3 of 3)

- create Player class with unittests #12
- create Board class and unittests #13
- create TicTacToe class implementing basic game loop #14

[Create sub-issue](#)

[soschoen](#) moved this to Backlog in slm-tictactoe-project last week

[soschoen](#) added this to slm-tictactoe-project last week

[soschoen](#) moved this from Backlog to Ready in slm-tictactoe-project last week

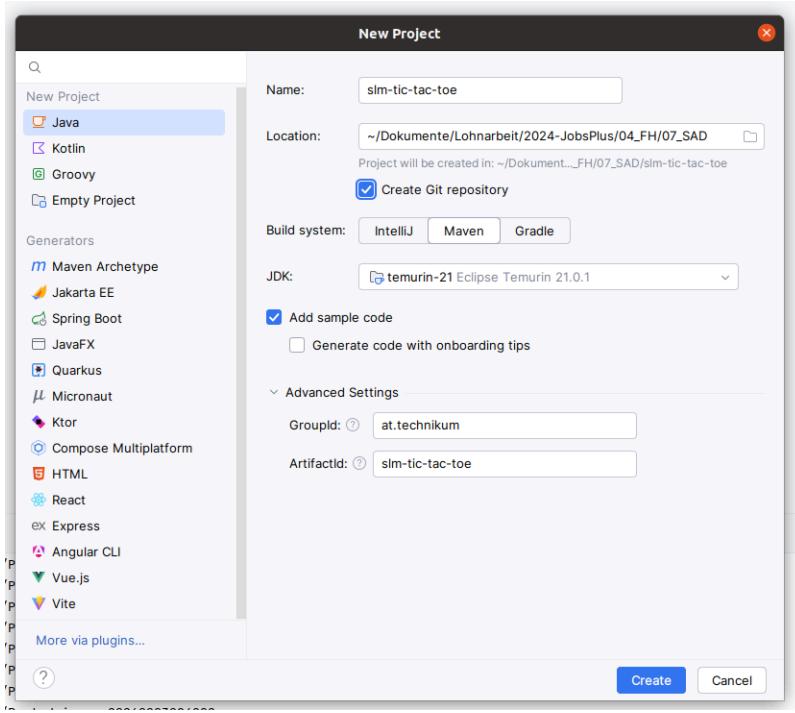
[soschoen](#) added sub-issues 5 days ago

- create Player class with unittests #12
- create Board class and unittests #13
- create TicTacToe class implementing basic game loop #14

Additionally, a couple of new issues were created regarding workflows and ci/cd, in order to set everything up correctly before starting to work on the actual user stories.

005 - Create a new Java Maven project

A new Java Maven project was created in the IDE and a new git repository was initialised in the process.



006 - Set the GitHub repository as a remote repository

The screenshot shows a Java project named "sim-tic-tac-toe" in an IDE. The project structure includes a src folder with main and test subfolders, and a resources folder containing a file named "st.technikum". The .gitignore file is open in the center pane, listing various files and folders to ignore during version control. The terminal pane at the bottom shows the command-line process of adding a GitHub remote repository named "origin" to the local project.

```

Main.java pom.xml (sim-tic-tac-toe)
Main.java pom.xml (sim-tic-tac-toe) .gitignore
1  target/
2  !.maven/wrapper/nexus-wrapper.jar
3  !*/src/main/*/*target/
4  !*/src/test/*/*target/
5
6  ## IntelliJ IDEA ##
7  .idea/modules.xml
8  .idea/jarRepositories.xml
9  .idea/compiler.xml
10 .idea/libraries/
11 *.iml
12 *.iml
13 *.ipr
14
15 ## Eclipse ##
16 .apt_generated
17 .classpath
18 .factorypath
19 .project
20 .settings
21 .springBeans
22 .sts-cache
23
24 ## NetBeans ##
25 !/nbproject/private/
26 !nbbuild/
27 !src/

```

```

eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAB/sim-tic-tac-toe$ git remote add origin git@github.com:ww24e018/sim-tictactoe.git
eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAB/sim-tic-tac-toe$ git remote -v
origin git@github.com:ww24e018/sim-tictactoe.git (fetch)
origin git@github.com:ww24e018/sim-tictactoe.git (push)
eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAB/sim-tic-tac-toe$ S

```

007 - Commit and push your blank Java project to the main branch

On our initial push to the main branch, we had a conflict between our local and remote branches.

Following differences lead to this:

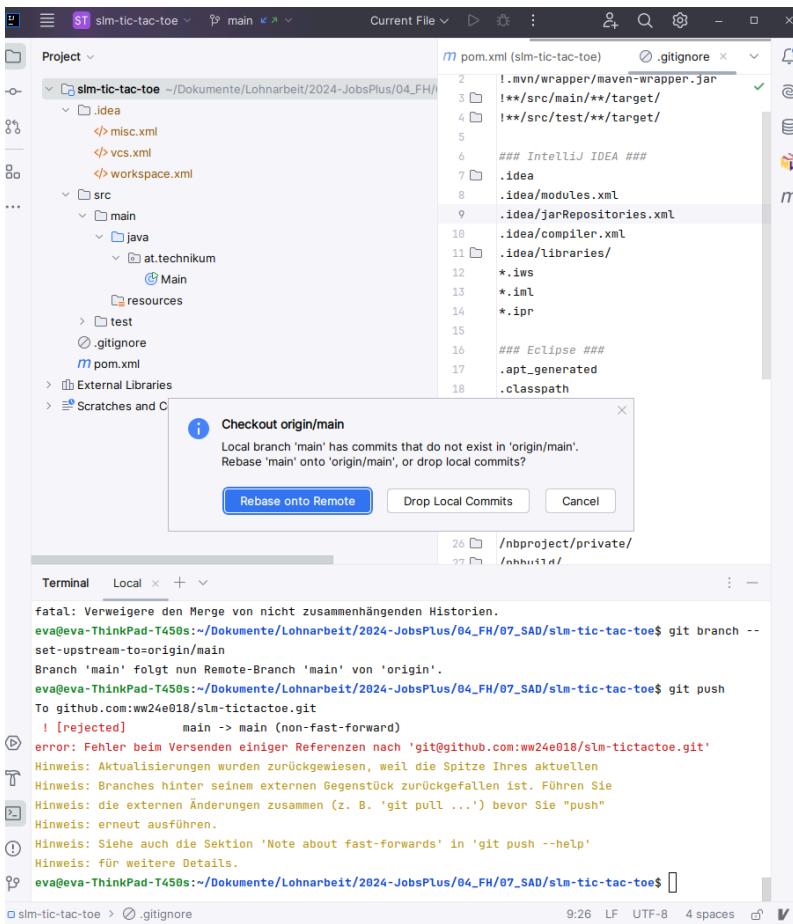
1. Remote branch was named `main` and local branch `master`.
2. On creation of the repository in github we added a `.gitignore` which was different to the `.gitignore` file that was automatically generated by IntelliJ.

We resolved this by (see screenshot of history)

1. Rename master (local) to main (`git branch -m master main`)
2. Pull (`git pull git@github.com:ww24e018/slm-tictactoe.git main`)
3. Set upstream of local main to origin/main (`git branch --set-upstream-to=origin/main`)
4. Git push
5. Rebase & resolve conflict
6. `git push`

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure for "slm-tic-tac-toe". It includes a `src` folder containing `main`, `java`, `test`, and `resources` subfolders. A `gitignore` file is also present in `src`. The `target` folder contains build artifacts like `classes.jar` and `dependency.jar`.
- Terminal:** Displays the command-line history for the current session. The commands shown are:
 - `git branch -m master main` (Renaming the local branch)
 - `git pull git@github.com:ww24e018/slm-tictactoe.git main` (Pulling from the remote repository)
 - `git branch --set-upstream-to=origin/main` (Setting the upstream for the local main branch)
 - `git push` (Pushing the changes to the remote repository)
 - `git status` (Showing the current status of the repository)
 - `git commit -m "Initial Commit of empty Maven project to remote repository"` (Committing the changes with a descriptive message)



History leading to rebase button:

```

eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ git push
fatal: Der aktuelle Branch master hat keinen Upstream-Branch.
Um den aktuellen Branch zu versenden und den Remote-Branch
als Upstream-Branch zu setzen, benutzen Sie

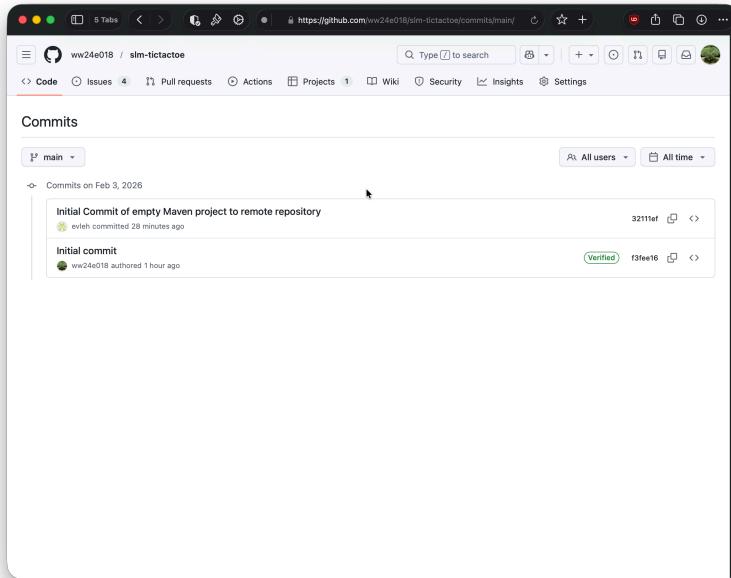
git push --set-upstream origin master

eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ git merge master
fatal: Verweigere den Merge von nicht zusammenhängenden Historien.
eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ git branch -m master main
fatal: Branch 'main' existiert bereits.
eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ git branch -m master main
eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ git push
fatal: Der aktuelle Branch main hat keinen Upstream-Branch.
Um den aktuellen Branch zu versenden und den Remote-Branch
als Upstream-Branch zu setzen, benutzen Sie

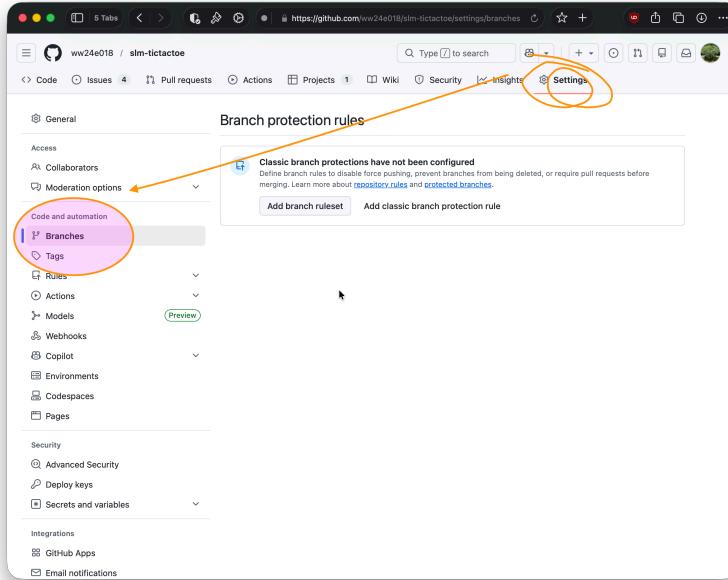
git push --set-upstream origin main

eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ git push --set-upstream origin main
To github.com:ww24e018/slm-tictactoe.git
 ! [rejected]      main -> main (non-fast-forward)
error: Fehler beim Versenden einiger Referenzen nach 'git@github.com:ww24e018/slm-tictactoe.git'
Hinweis: Aktualisierungen wurden zurückgewiesen, weil die Spitze Ihres aktuellen
Hinweis: Branches hinter seinem externen Gegenstück zurückgefallen ist. Führen Sie
Hinweis: die externen Änderungen zusammen (z. B. 'git pull ...') bevor Sie "push"
Hinweis: erneut ausführen.
Hinweis: Siehe auch die Sektion 'Note about fast-forwards' in 'git push --help'
Hinweis: für weitere Details.
eva@eva-ThinkPad-T450s:~/Dokumente/Lohnarbeit/2024-JobsPlus/04_FH/07_SAD/slm-tic-tac-toe$ 
```

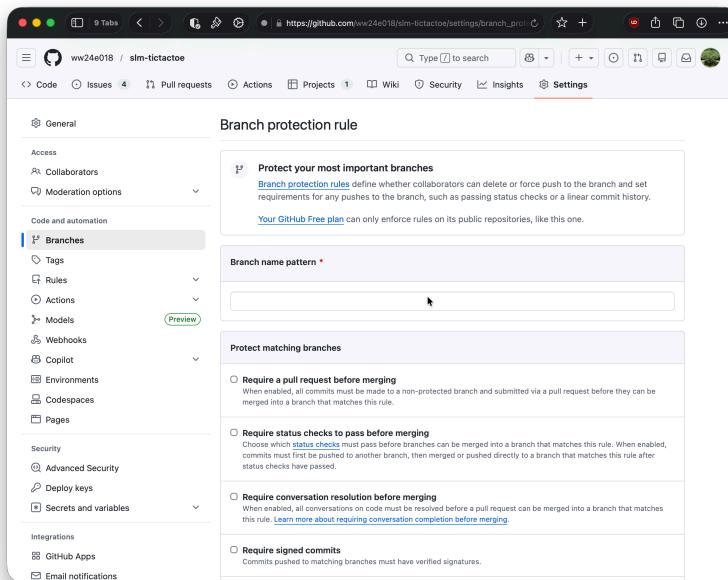
Screenshot of the commit history on GitHub:



008 - Protect the main branch



After reading documentation about branch protection rules ([repository rules](#) / [protected branches](#)), it was decided to try the "add classic button":



The screenshot shows the GitHub branch protection settings page. On the left, there's a sidebar with 'Advanced Security', 'Deploy keys', and 'Secrets and variables'. The main area lists several protection rules:

- Require conversation resolution before merging**: When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule.
- Require signed commits**: Commits pushed to matching branches must have verified signatures.
- Require linear history**: Prevent merge commits from being pushed to matching branches.
- Require deployments to succeed before merging**: Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.
- Lock branch**: Branch is read-only. Users cannot push to the branch.
- Do not allow bypassing the above settings**: The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

Below these is a section titled 'Rules applied to everyone including administrators':

- Allow force pushes**: Permit force pushes for all users with push access.
- Allow deletions**: Allow users with push access to delete matching branches.

A green 'Create' button is at the bottom.

Defaults appearing after clicking the first checkbox

The screenshot shows the 'Protect matching branches' section. Under 'Require a pull request before merging', the 'Required approvals' dropdown is set to '1'. Other options shown include:

- Require a pull request before merging**: When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.
- Require approvals**: When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged. A dropdown shows 'Required number of approvals before merging: 1 ▾'.
- Dismiss stale pull request approvals when new commits are pushed**: New reviewable commits pushed to a matching branch will dismiss pull request review approvals.
- Require review from Code Owners**: Require an approved review in pull requests including files with a designated code owner.
- Require approval of the most recent reviewable push**: Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

Other options were left at default.

After clicking 'create':

The screenshot shows the GitHub repository settings for 'wv24e018 / slim-tictactoe'. The 'Branches' section is selected. A modal window titled 'Branch protection rule created.' is open, indicating a new rule has been added. The rule applies to the 'main' branch, which currently has no protection rules applied. The modal includes a 'Level up your branch protections with Repository Rules' section with a 'Learn more' link and a 'Go to rulesets' button. The left sidebar lists other repository settings like General, Access, Collaborators, and Moderation options.

009 - CI_CD Scripts and workflows

The process of research and development

To simplify research and development - and to keep the submission repo free of commits doing trial and error on syntax and meaning - a decision was made to use a custom test-repository on GitHub to experiment more freely.

This repository is also public and is available [here](#) if a deeper view into that process is required (or desired).

It does not have the full spectrum of main-branch protections because its purpose is to serve as a playground for quick test runs that might require changes to be present on main.

The process of implementing results

was the usual of creating tickets/issues/board-items (! not userstories) on the board, creating branches, pushing branches, creating merge requests, requesting review, doing review, incorporating review and (at some point) merging.

Links to the main 3 topical MRs/PRs:

- <https://github.com/ww24e018/slm-tictactoe/pull/11>
- <https://github.com/ww24e018/slm-tictactoe/pull/19>
- <https://github.com/ww24e018/slm-tictactoe/pull/21>

And links to the issues these close:

- <https://github.com/ww24e018/slm-tictactoe/issues/10>
- <https://github.com/ww24e018/slm-tictactoe/issues/18>
- <https://github.com/ww24e018/slm-tictactoe/issues/20>

Exemplary screenshot:

The screenshot shows a sequence of GitHub events for a pull request. It starts with a commit from 'ww24e018' adding a GitHub Actions file. This is followed by a self-assigned issue, a review request from 'evleh' and 'soschoen', and an update to the heading. 'soschoen' approves the changes. A comment from 'soschoen' follows, saying 'looks good to me :)' with a smiley face emoji. The pull request is then merged by 'ww24e018' into the 'main' branch. A success message indicates the merge and provides a link to delete the branch. This pattern repeats for two more pull requests, each receiving approval, a comment, and being merged into 'main'.

The suggestion made in the comment above lead to <https://github.com/ww24e018/slm-tictactoe/issues/20> / <https://github.com/ww24e018/slm-tictactoe/pull/21>.

Technical description

The process resulted in 2 yaml files:

- `.github/workflows/maven.yml`
- `.github/workflows/maven-publish.yml`

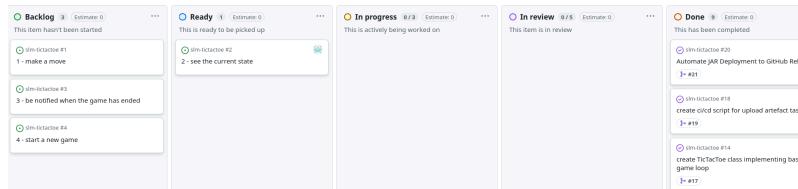
These names are the default ones given by the github-actions wizard. Later the question turned up if those two (who have an identical build-command) could be unified.

It was assumed this would be possible but would/might require research into and testing of more advanced github-actions-syntax to distinguish between the different trigger-events.

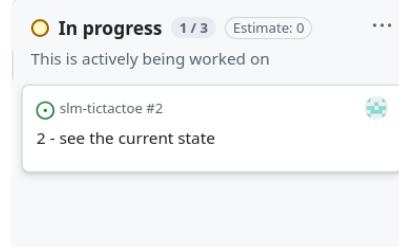
For now - while possibly slightly inefficient - they satisfy requirements.

010 - Working on User Stories

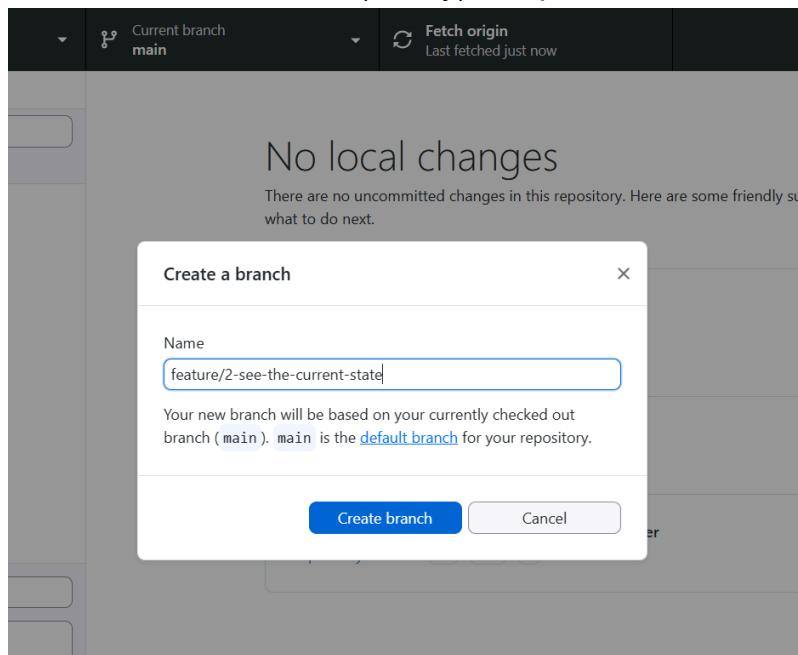
- Once a user story is moved to `ready`, it is assigned to one or more contributors after checking in with each other in person.



- When contributor starts actively working on the story, they move it to `In progress`.



- A new branch is created (locally) and published to remote.



- After the first commit is ready, we push to remote and create a pull request which we mark as draft. This is to further visualize our work in progress to the rest of the team.

5. In our description we link the pull request to the issue.

See the current state #22

Draft: soschoen wants to merge 1 commit into `main` from `feature/2-see-the-current-state`

Conversation 0 | **Commits** 1 | **Checks** 0 | **Files changed** 2

Reviewers: `ww24e018` (Request)
At least 1 approving review is required to merge this pull request.

Assignees: `soschoen`

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.
None yet

Notifications: Unsubscribe
You're receiving notifications because you authored the thread.

Add a comment

Merge pull request | You can also merge this with the command line. [View command line instructions](#).

6. When the developer is done with the ticket and all checks have passed, they set the pull request to "ready for review", request a review and move the ticket to `In review`.

Review required: At least 1 approving review is required by reviewers with write access.

All checks have passed: 1 successful check

This pull request is still a work in progress: Draft pull requests cannot be merged.

Ready for review

Merge pull request | You can also merge this with the command line. [View command line instructions](#).

Reviewers: Request up to 15 reviewers
Type or choose a user
`ww24e018` (checked)
`evleh` (selected)

Labels: None yet

In review 1 / 5 Estimate: 0

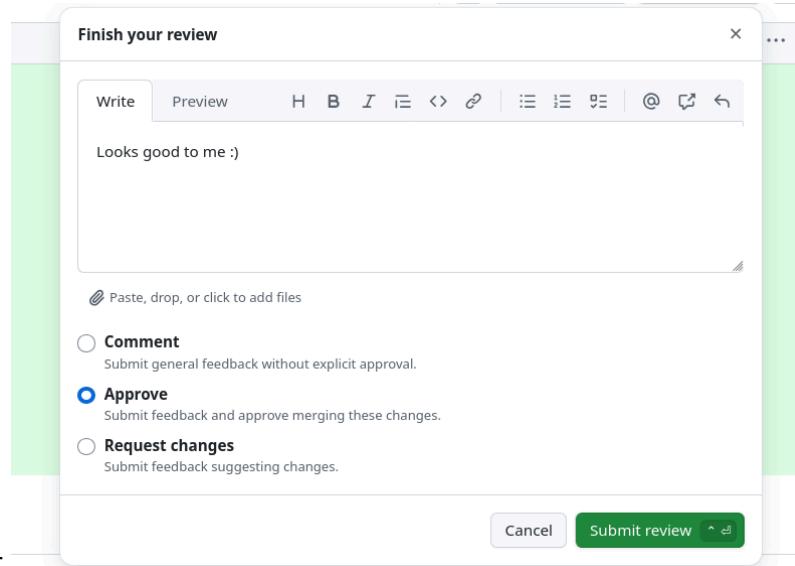
This item is in review

slm-tictactoe #2
2 - see the current state

Automate #21

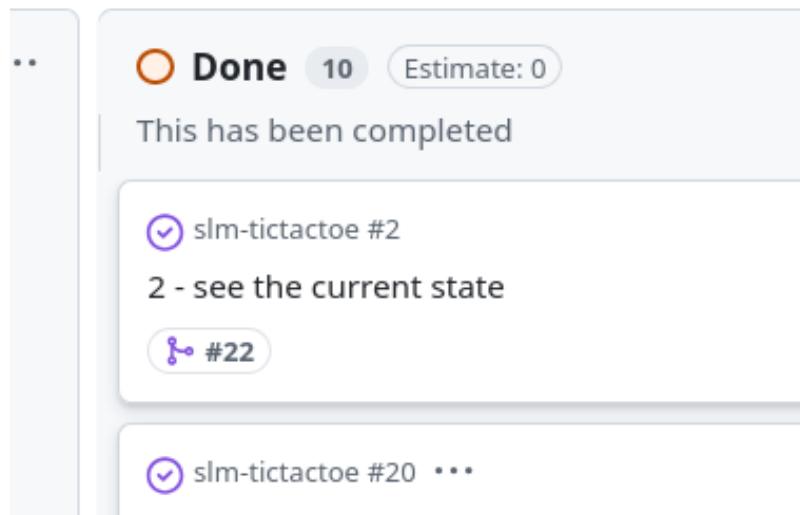
slm-create

7. The reviewer writes a review. If there are no necessary changes, they can directly



approve the request.

8. After the pull request is approved, the developer merges to main and moves the ticket to Done .



9. Sometimes the reviewer has questions, remarks, recommendations or corrections.

These are posted as comments. If necessary or useful, they are also discussed in person. The review is then submitted as requesting changes.

10. The author makes amends to the code and answers to the comments left by the reviewer. After changes have been made, they re-request a review.

11. The reviewer who requested the changes reviews the author's answers and changes to the code and ultimately resolves the conversation in the comments.

The screenshot shows a GitHub pull request interface. At the top, it says "soschoen requested changes 4 days ago". Below this is a code diff for a file named "src/main/java/at/technikum/tictactoe/Board.java". The code is as follows:

```

20 +
21 +     public boolean isFull() {
22 +         boolean nonEmptyCellFoundYet = false;
23 +         for (var i = 0; i < 2; i++) {

```

There is a comment from "soschoen" asking if the loop condition should be "i <= 2" instead of "i < 2". A reply from "ww24e018" says "fixed in commit (2 was error)". At the bottom, there is a button to "Unresolve conversation" and a note that "soschoen marked this conversation as resolved".

12. Once they are satisfied with the result, they approve the pull request. Subsequently the feature branch is merged by the author (see steps 7-8).

011 - A final overview

A clean Kanban board - all issues have been moved to done :

Exemplary game run - everything works as it should:

Our big release: