Week 8: Intro to tP - Summary and Step-by-Step Tasks

Summary

Week 8 focuses on starting the implementation of tP v1.0, emphasizing incremental development, proper Git workflows, and project tracking using GitHub tools. Key points include avoiding common mistakes like incorrect commit messages, improper PR branches, and non-compliant method comments. The week introduces project schedule tracking via GitHub issues, milestones, PRs, and releases. Teams should begin coding v1.0 in small steps, repurposing code from iP or course activities with proper credit. Add JUnit tests for code quality.

Step-by-Step Details for Week 8 Tasks

1. Review Common Mistakes and Best Practices

- **Avoid incorrect Git commit message subjects**: Follow conventions to prevent issues with merged PRs affecting contribution stats.
- **Ensure proper Java method header comments**: Use required phrasing style for the first sentence.
- **Use separate branches for PRs**: Always send PRs from a branch in your fork, not the master branch, to follow the forking workflow correctly.
- **Caution on PR merging**: Do not rebase/squash to preserve commit timestamps for grading.

2. Start Implementing v1.0

- **Repurpose code ethically**: Adopt code from team members' iP, AddressBook-Level2, or course activities (e.g., personbook), but give credit and do not claim as your own.
- **Incremental development**: Add code in small steps, working in parallel to produce a simple working version after one week, and a more functional version by the end of the iteration (two weeks).
- **Follow prescribed workflow**: Use the project schedule tracking strategy outlined below.

3. Set Up Project Schedule Tracking with GitHub Tools

- **Track WHAT (tasks)**: Post tasks as issues in GitHub issue tracker. Break down features into small, standalone tasks (e.g., "Implement parser support for adding events" instead of "Write the Developer Guide").
    - Write descriptive titles, assign type.* and priority.* labels if applicable.
    - Ensure issues align with breadth-first iterative approach for working versions.
- **Track WHO (assignment)**: Assign issues to team members.
- **Track WHEN (scheduling)**: Assign issues to GitHub milestones with deadlines.
- **Track HOW (progress)**: Create PRs for tasks in progress; merging PRs (and closing issues) indicates completion.
- **Track iteration progress**: Use milestones for iterations; releases for deliverables; close milestone on completion.
- **Optional**: Use GitHub Projects linked to issues/PRs for management.

4. Detailed Issue and PR Management

- **Creating issues**: Define tasks as issues with good titles and minimal descriptions. Use for user stories (e.g., title: "As a user I can add a deadline").
- **Assigning issues**: Assign to one person per task; split shared tasks.
- **Milestones**: Create for iterations (e.g., v1.0), set deadlines, assign issues/PRs.
- **PR creation**: Use forking workflow (separate branch per PR). Title same as issue. Use draft PRs, assign to milestones. Use "Fixes #123" for auto-closing.
- **PR reviewing**: Follow best practices; give location-specific comments; use LGTM.
- **PR merging**: Only merge passing CI; sync repos after merging; keep branches for grading.
- **Update tests**: Modify input.txt and EXPECTED.txt for regression testing.

5. Add JUnit Tests

- **Recommendation**: Each team member adds JUnit tests for their tP code.
- **Examples**:
    - Test Utils.java with UtilsTest.java (same package for access).
    - Test Parser.java with ParserTest.java (check naming conventions).
    - Test AddressBook.java with AddressBookTest.java.
- **Purpose**: Ensure code quality and practice testing.

6. Iteration Workflow

- **Start**: Create milestone (e.g., v1.0), set deadline, create/assign issues.
- **During**: Send PRs, review/merge, adjust as needed.
- **End**: Update milestone, create release if needed, close milestone.

By following these steps, ensure a structured, incremental approach to tP v1.0 development with proper tracking and quality assurance.