Nachdenkzettel: Git

1.

Was sind Gründe ein Version Control System (VCS) wie Git zu verwenden?

Es bietet eine Versionskontrolle (Speicherung von Versionen von Code und Wiederherstellung dessen), man kann hat die Möglichkeit gleichzeitig an einem Projekt zu arbeiten (man kann diese Veränderungen auch Nachverfolgen) und es ist automatisch ein Backupsystem

2.

Welche Dateitypen gehören in ein Repo, welche nicht?

- .java, .xml, .json: Sind Code-/ Konfigdateien, welche wichtig für das Projekt sind, damit es richtig funktioniert.

- Bilddateien, Musikdateien: Sollten nur miteingebunden werden, wenn sie für das Projekt genutzt werden, da sie sonst das Repo zu stark befüllen.

- UML Modelle, Notizen, Dokumentationen: Sollten in einem Repo sein, damit man als Neueinsteiger das Projekt besser versteht.

- Vertrauliche Daten (z.B. Passwörter): Sollen nicht im Repo vorhanden sein, da es bad practice ist vertrauliche Daten öffentlich in einem Repo stehen zu haben.

- Abschlussarbeiten: Sollten nicht in einem Repo stehen, da sie nicht für die Funktion des Programms relevant sind, da es nur eine PDF ist. Dies füllt das Repo unnötig und kann verwirren.

- Log-Files: Log-Files ändern sich ständig, wodurch man es nicht mit in ein Repo packen soll, da sie auch nur für Fehleranalyse genutzt werden bei einem aktuellen Lauf.

- Konfigurationen: Konfigdateien, welche das Programm beeinflussen, sollen in dem Projekt vorhanden sein und Andere nicht, da sie unnötig sind.

3.

Bevor Änderungen committet werden muss eine Staging-Area hinzugefügt werden. (git add)
Danach kann man es per commit im Repo zwischenspeichert werden und es auch mit einer Nachricht ausstatten. Nach dem committen kann man das zwischengespeicherte mit push in das remote Repo schicken (dies sehen die anderen Repomitglieder). Durch das Pushen ist eine neue Version im Repo, dies können andere Nutzer mit einem Pull in ihre lokale Version einbinden.

4.

Wie beheben Sie einen Merge Conflict?

Git markiert den Fehler mit <<<<<<>>>>>>. Um den Fehler zu beheben muss man es manuelle machen. Dies geht mit neuschreiben oder eine ältere Version verwenden und kombinieren.
Sobald dies gemacht wurde, fängt der ganze Prozess des hinzufügen in ein Repo.

5.

Wie ist Ihr Git-Vorgehen im Team? Wann werden Änderungen gepusht? Wie oft veröffentlichen Sie Ihre Änderungen? Wie beschreiben Sie Ihre Änderungen in den commit-Messages?

In unserem Team bekomm jedes Mitglied ein Branch und pusht dort seine Änderung hinein.
Dies geschieht jedes Mal, wenn man eine neue Methode implementiert hat. Die Branch-Versionen kommen in die Main, sobald sie vollfunktionsfähig sind. Die commit-Messages sind auf Englisch und beschreiben kurz das Veränderte, neu hinzugefügte oder das Gelöste.