Testkonzept

Junit Tests

Wir haben als erstes BeforeEach importiert und darunter eine Methode erstellt mit dem Namen before(). Diese Methode wird vor jeder Testmethode einmal aufgerufen. In dieser Methode haben wir alle Items instanziert, da diese Items in allen drei Methoden vorkommen haben wir sie in die before() Methode eingefügt.

```
@BeforeEach
public void before() {
    item = new Item("pen", 5);
    item2 = new Item("vase", 40);
    item3 = new Item("chair", 100);
}
```

Die Tests haben wir immer gleich benannt beginnend mit «test» und anschliessend den Namen der Methode, welche wir testen.

Beispiel:

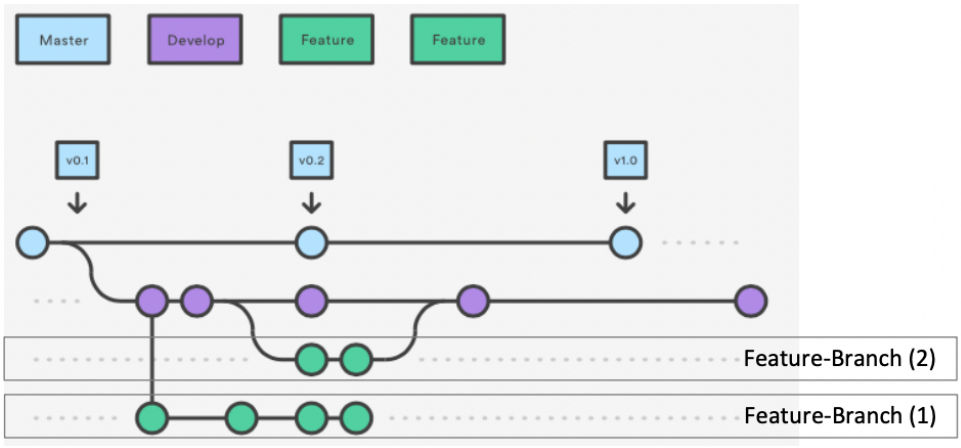
```
@Test
public void testTakeItem() {
    Command take = new Command("take", "pen");
    room.setItem(item);
    game.setCurrentRoom(room);
    game.takeItem(take);

    assertTrue(game.getInventory().getItemList().containsKey("pen"));
    assertFalse(game.getCurrentRoom().getItems().containsKey("pen"));
}
```

Hatice war dabei grundsätzlich für die einzelnen Klassen also die Räume, das Inventar und die Items aber auch für Grundstruktur verantwortlich. Luca war für die Methoden der Gameklasse zuständig.

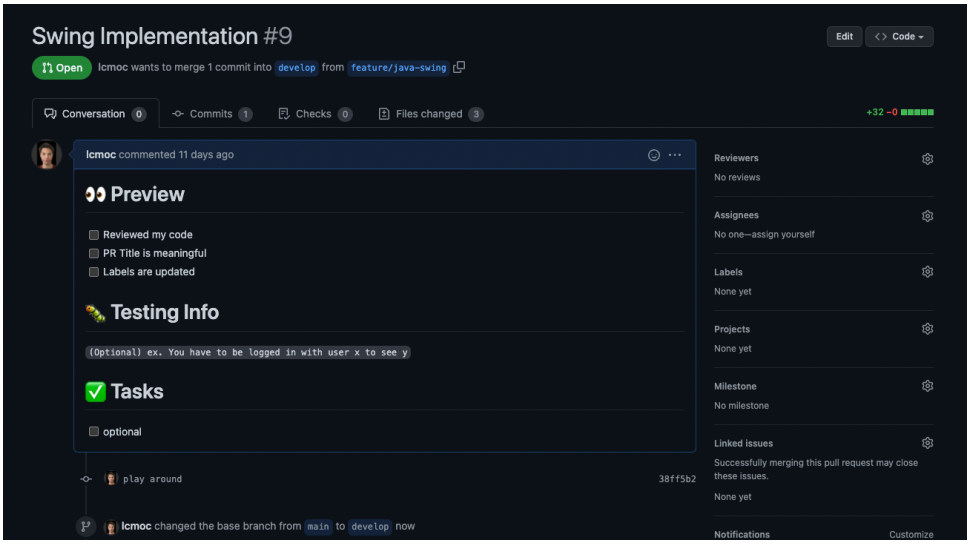
Git Vorgehensmodell

Wie während der Stunde besprochen, sind wir nach dem Vorgehensmodell unten vorgegangen.

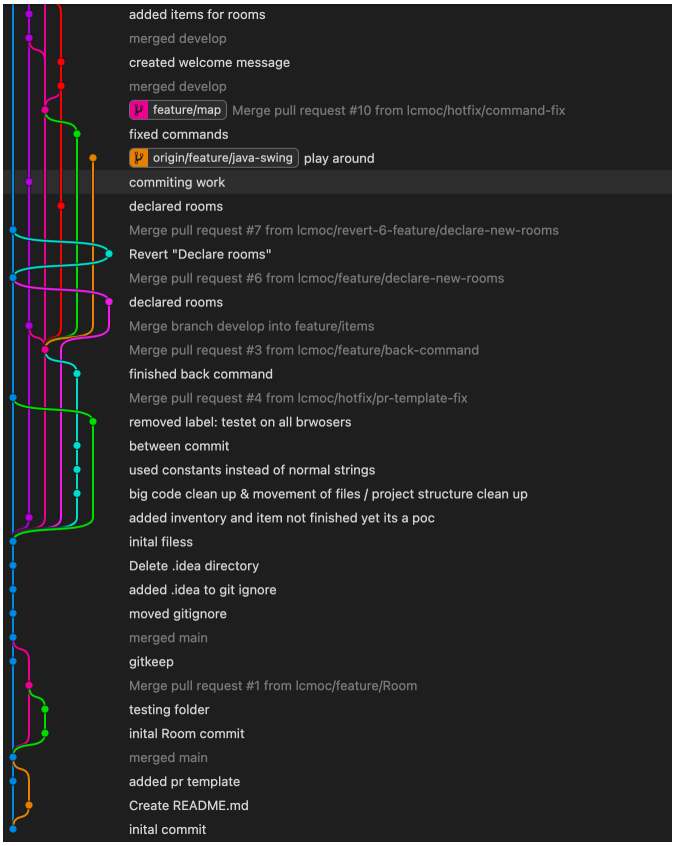


Wir haben die Branches: main, develop und individuelle feature-branches erstellt. Auf den feature-branches haben wir unsere Änderungen gemacht. Da wir diese jedes Mal, bevor wir diese in den develop branch mergen testen wollten, haben wir Pull Request's (pr's) erstellt. Durch das selbst erstellte markdown file, haben wir selbst definiert Punkte abgearbeitet, um die Reinheit des Codes zu garantieren und Bugs zu vermeiden.

Beispiel:



Ein Teil unserer git history:



Clean-Code Regeln

Unsere wichtigste Regel ist KISS – Keep it simple stupid.
In dem wir unkomplizierte, jedoch gut verständliche Namen für Methoden verwenden, wird unser Code besser verständlich und Qualitative besser.

Ausserdem wollten wir Kommentare vermeiden, da wir denken, dass Code ohne Kommentare genauso gut verständlich sein sollte.

Für die Formatierung haben wir den build-in Formatter von IntelliJ verwendet.