**Testkonzept (Testprotokoll)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Klasse (Methode)** | **Testfall** | **Erwartung** | **Resultat** | **Kommentar** |
| CommandWords.isCommand | Existentes Commandword | Funktioniert | Funktioniert | OK |
| CommandWords.isCommand | Nicht existentes Commandword | Stoppt | Stoppt | OK |
| Itemtest.getType | Item Type wird verglichen | Es werden nur Item Typen akzeptiert. | Es werden nur Item Typen akzeptiert. | OK |
| Itemtest.setType | Item Type wird gesetzt | Es werden nur Item Typen akzeptiert. | Es werden nur Item Typen akzeptiert. | OK |
| Room.isLocked | Checkt, ob der Raum geschlossen ist | Er gibt true oder false zurück, ob der Raum geschlossen ist. | Er gibt true oder false zurück, ob der Raum geschlossen ist. | OK |
| Time.calculateTimer | Berechnung der Zeit | Die Zeit wird berechnet. | Die Zeit wird berechnen. | OK |

**Git Vorgehensmodell**

1. Repository erstellt
2. Projects für beide Module gemacht und Aufträge hinzugefügt
3. Branches erstellt
4. Branches nach push und merge gelöscht
5. Neue Branches erstellt und nach push und merge gelöscht
6. Zwischendurch Development in Main gemerget
7. Main getestet und gepusht

**Clean-Code Regeln**

1. Java naming Convention
2. Methoden kurzhalten
3. Alles Objektorientiert machen
4. Möglichst viel Testen
5. Javadoc

**Reflexion**

Mit dem Programm und unserem Vorgehen sind wir sehr zufrieden. Wir konnten alle Tests machen, die benötigt wurden und haben auch so viel wie möglich OOP gemacht. Die Arbeitsaufteilung war auch sehr ausgeglichen und es hat praktisch nie jemand nichts gemacht. Neu gelernt haben wir, dass man mit einer Funktion von Intellij sieht, welche Klasse wie fest getestet wurde. Nächstes mal würden wir nicht viel anders machen, ausser von Anfang an mit Clean-Code arbeiten.