Labor 2

Matișan Rareș-Ștefan

- Was meine Anwendung betrifft, möchte ich verschiedene CRUD-Operationen auf der Liste aller Zutaten durchführen. Demnach kann man, zum Beispiel, Update-Operationen auf dieser Liste verwenden, um den Preis und/ oder die Menge (wie viele Zutaten dieses Typs es gibt) einer Zutat zu wechseln. Diese Funktion hat 2 Parameter: einen String für den Namen der Zutat und eine neue Zutat.

```java
@Override
public void update(String name, Zutat newEntity) {
    if(newEntity.getPreis() > 0 && newEntity.getPreis() <= 100
            && newEntity.getMenge() >= 1 && newEntity.getMenge() <= 100) {
        for (Zutat zutat : alleZutaten) {
            if (zutat.getName().equals(name)) {
                zutat.setMenge(newEntity.getMenge());
                zutat.setPreis(newEntity.getPreis());
                break;
            }
        }
    }
    else {
        System.out.println("Your update was not possible due to not respecting the constraints");
    }
}
```

- Folgende **Bedingungen** (Constraints) wurden identifiziert:
  → **newEntity** (neue Zutat) muss nicht einen **Preis** außerhalb des Intervalls (0, 100] haben (der Intervall ist offen rechts, weil der Preis eine nichtnulle Wert haben muss)
  → **newEntity** muss nicht eine **Menge** außerhalb des Intervalls [1, 100] haben (das heißt es muss nicht unten 1 Zutat oder mehr als 100 Zutaten derselben Typ geben)

- In einfacheren Worten:
  → newEntity.getPreis() > 0 && newEntity.getPreis() <= 100
  → newEntity.getMenge() >= 1 && newEntity.getMenge() <= 100

- Für Vereinfachung werden wir “p” für newEntity.getPreis() und “m” für newEntity.getMenge() benutzen.

- Anhand dieser Bedingungen wurden folgende Testfälle für **ECP** identifiziert:
  - 1 EC gültig:
    - EC1: p = (0, 100], wobei p eine rationale Zahl ist &&
      m = [1, 100], wobei m eine natürliche Zahl ist.

  - 9 ECs ungültig:
    - EC2: p = (0, 100] && m = (-MAXINT, 0]
    - EC3: p = (0, 100] && m = [101, MAXINT)
    - EC4: p = (-MAXINT, 0] && m = [1, 100]

        - EC5: p = (-MAXINT, 0] && m = (-MAXINT, 0]
        - EC6: p = (-MAXINT, 0] && m = [101, MAXINT)
        - EC7: p = (100, MAXINT) && m = [1, 100]
        - EC8: p = (100, MAXINT) && m = (-MAXINT, 0]
        - EC9: p = (100, MAXINT) && m = [101, MAXINT)

- Anhand dieser Bedingungen wurden folgende Testfälle für **BVA** identifiziert:
    - 12 Cs gültig:
        - C1: p > 0 && m > 1
        - C2: p > 0 && m < 100
        - C3: p > 0 && m = 1
        - C4: p > 0 && m = 100
        - C5: p < 100 && m > 1
        - C6: p < 100 && m < 100
        - C7: p < 100 && m = 1
        - C8: p < 100 && m = 100
        - C9: p = 100 && m > 1
        - C10: p = 100 && m < 100
        - C11: p = 100 && m = 1
        - C12: p = 100 && m = 100

    - 18 Cs ungültig:
        - C13: p < 0 && m > 1
        - C14: p < 0 && m < 100
        - C15: p < 0 && m = 1
        - C16: p < 0 && m = 100
        - C17: p < 0 && m < 1
        - C18: p < 0 && m > 100
        - C19: p = 0 && m > 1
        - C20: p = 0 && m < 100
        - C21: p = 0 && m = 1
        - C22: p = 0 && m = 100
        - C23: p = 0 && m < 1
        - C24: p = 0 && m > 100
        - C25: p > 100 && m > 1
        - C26: p > 100 && m < 100
        - C27: p > 100 && m = 1
        - C28: p > 100 && m = 100
        - C29: p > 100 && m < 1
        - C30: p > 100 && m > 100