# LU02a - Konsoleneingabe

## Main-Methode

Die Main-Methode ist der Ausgangspunkt einer Java-Applikation.

```
    public static void main(String[] args) {
        ...
    }
```

Das String-Array `args`, welches der Methode übergeben wird, entsprechen den Argumenten, wenn das kompilierte Programm aufgerufen wird.

Erstellen Sie ein Fork-Repository mit **allen** Branches von https://github.com/bzz-templates-java/ffit-lu02-library-app Klonen Sie anschliessend ihr eigenes Repository, um die Basisstruktur für die LibraryApp zu erhalten, welche Sie in den kommenden Kapiteln implementieren werden.

## Scanner

Bei vielen Backend-Applikationen erfolgt die Benutzerinteraktionen via API-Aufrufen oder anderen Schnittstellen. Für den Anfang können wir die Klasse `Scanner` nutzen. Diese erlaubt es, Benutzereingaben in der Konsole zu lesen.

Nutzen Sie die Klasse `Scanner`, um einen String einzulesen.

- https://javabeginners.de/Ein-_und_Ausgabe/Scanner.php
- https://www.w3schools.com/java/java_user_input.asp

Diesen String verwenden wir als Befehl, um verschiedene Implementierte Funktionen auszuführen.

**Anforderung 1**: Die Eingabe `quit` soll das Programm beenden, die Eingabe `help` soll die Liste aller implementierten Befehle auflisten und bei einer anderen Eingabe soll eine Nachricht ausgegeben werden, dass die Eingabe nicht als Befehl erkannt wurde. Nach dem Ausführen eines Befehls (ausser `quit`) soll der nächste eingegeben werden können usw.

Überlegen Sie sich vor der Implementation, wie der Code so gestaltet werden kann, dass zukünftige Befehle einfach hinzugefügt werden können. Vielleicht haben Sie bereits eine Idee, wie man die Befehle erfassen könnte, so dass die Liste der Befehle (`help`) nicht separat erweitern muss. Sie können aber solche Verbesserungen auch später noch erledigen.

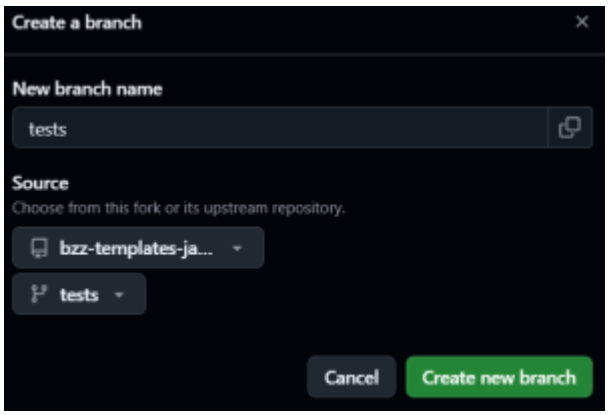
Machen Sie einen Cherry-Pick des Commits `d5cca5d`, um die Tests für diesen Implementationsschritt zu erhalten. Führen Sie die Tests aus und korrigieren Sie ihre Implementation falls nötig.

Falls der Cherry-Pick fehlschlägt, könnte es allenfalls daran liegen, dass Sie den Fork nicht mit allen Branches erstellt haben. In diesem Fall müssen Sie den Branch „tests“

manuell mit dem „tests“-Branch als Origin erstellen. Das geht z.B. via:

1. https://github.com/<USERNAME>/<REPOSITORY>/branches
2. „new Branch“

- branch name: „tests“
- Source: „bzz-templates-java/<REPOSITORY>“
- (Source Branch) „tests“




Sollte die Fehlermeldung `bad revision` auftauchen, checken Sie den anderen Branch aus, überprüfen Sie, ob der Commit in den Logs vorhanden ist und wechseln Sie wieder zurück auf den Main-Branch.

```
git checkout tests
git log
git checkout main
```

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
**https://wiki.bzz.ch/de/modul/ffit/3-jahr/java/learningunits/lu02/main?rev=1788248351**

Last update: **2026/09/01 09:39**