# Wie kann man git beim Programmieren verwenden?

Bislang haben wir zu Demo-Zwecken in unserem Tagebuch geschrieben - das hatte vor allem den Vorteil, dass wir die Zeitleiste vor unserem inneren Auge hatten. Nun wollen wir uns überlegen, wie man git beim Programmieren einsetzen kann.

Dazu stellen wir uns vor, dass wir mit dem Arduino ein Auto steuern wollen, geplant sind die folgenden Projektschritte:

1. Unterprogramme wie Vorwärts, Kurven und Ähnliches, um das Auto zu steuern
2. Einbau eines Ultraschallsensors, um das Auto vor einem Hindernis zu stoppen
3. Einbau eines IR-Sensors, auslesen der IR Werte
4. Verwendung des IR-Sensors, um mit dem Auto einer schwarzen Linie auf dem Boden zu folgen

## Jedes Feature ein Commit!

- Es empfiehlt sich, mindestens für jedes Feature einen eigenen Commit anzulegen.
- Wenn man ein funktionierendes Programm hat, und Änderungen vornehmen möcht, bei denen man sich unsicher fühlt, ob die geplante Programmierung klappt, kann man sich den aktuellen Stand als Commit sichern.
- Vorsicht ist bei "Backup-Commits" geboten, z.B. immer am Ende einer Unterrichtsstunde, denn in diesem Fall läuft man Gefahr, Code in einem Snapshot zu sichern, der nicht funktioniert - das ist natürlich suboptimal. Das sollte man nur mit gutem Grund tun, beispielsweise wenn man den Code auf einen externen Server "pushen" möchte, um zuhause weiter zu arbeiten - dazu später mehr.

## Beispiele

Wir erstellen ein Bluej Projekt, das wir mit git verwalten wollen. Dabei verwenden wir zunächst nicht die eingebaute git Funktionalität von Bluej, hier soll es darum gehen, wie man mit einfachen Verzweigungen (Branches) arbeiten kann. Du kannst das Beispiel einfach nachvollziehen, wir verwenden einfach die Einstiegsübungen von dieser Wiki-Seite.

- Lege ein neues Bluej-Projekt an.
- Öffne eine Kommandozeile im Projektverzeichnis
- Initialisiere ein git-Repository im Projektverzeichnis.

Lösungsvorschläge zu den Programmieraufgaben findest du auf dieser Seite, von dort kannst du Lösungen für einzelne Aufgaben übernehmen, ohne alles jetzt programmieren zu müssen.So kannst du dich auf das Arbeiten mit git konzentrieren.

---

**(A1)**

Überhemne den "langen" Lösungsvorschlag für die Modulo-Methode in dein Projekt. Teste die Funktionalität und erstelle einen ersten Commit.

Ändere die Modulol-Methode jetzt so ab, dass sie dem zweiten, sehr kurzen Lösungsvorschlag entspricht. Erstelle einen weiteren Commit mit sinnvoller Commit Message.

Im Ergebnis könnte das dann so aussehen:

```
git lg
* b664f26 - (HEAD -> main) Modulo Methode gekürzt (vor 13 Sekunden)
* 143f0be - Erster commit (vor 6 Minuten)
```

"Löse" die Aufgabe 2, verfahre dabei analog zur Modulo Methode: Übernehme zunächst den ersten Lösungsvorschlag, erstelle einen Commit, ändere die Methode dann so, dass sie dem zweiten Lösungsvorschlag entspricht und erstelle dann einen weiteren Commit.

Als Zwischenergebnis erhält man einen Verlauf wie den Folgenden:

```
git lg
* 118e70a - (HEAD -> main) Tauschen Methode wie in Vorschlag 2 (vor 3
Sekunden)
* 99ba536 - Tauschen Methode wie in Vorschlag 1 (vor 25 Minuten)
* b664f26 - Modulo Methode gekürzt (vor 30 Minuten)
* 143f0be - Erster commit (vor 35 Minuten)
```