Debugowanie i systemy kontroli wersji

Karol Kulinowski, Agnieszka Kazimierska

10 października 2023

Debugowanie w IntelliJ[1][2]

Debugowaniem nazywamy proces znajdowania i naprawiania błędów w kodzie. **Debugger** pozwala znaleźć błędy przez wgląd w pracę programu krok po kroku w kontrolowany sposób, poprzez np. wstrzymywanie pracy programu czy podgląd wartości zmiennych w konkretnych punktach.

Pierwszym krokiem w debugowaniu jest ustalenie tzw **breakpointów**. Są to znaczniki umożliwiające zatrzymać program w miejscu, który chcemy przeanalizować i poddać testom za pomocą debuggera.

W trakcie analizy debugger pozwala także wykonywać program linia po linii w poszukiwaniu błędu, sprawdzać wartości zmiennych w trakcie działania programu oraz zmieniać ich wartość.

Debugowanie, przykłady

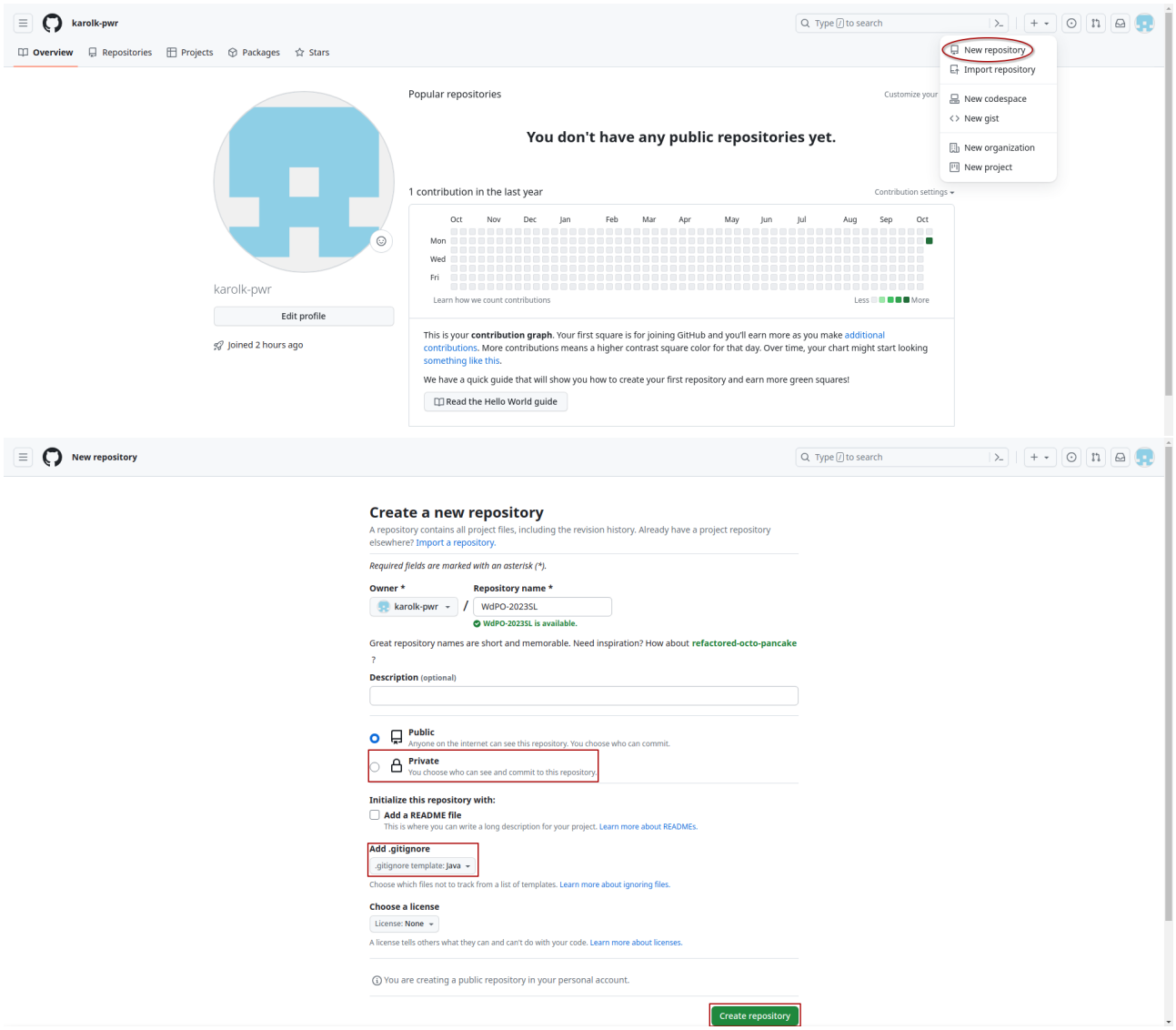
- Przykład zwierający trzy błędy.

```
1  public class debuggingExample1 {
2      public static void main(String[] args) {
3          int[] liczby = new int[] { 2, 10, 0, -25, 2, 100 };
4          System.out.println(znajdz(10, liczby));
5          System.out.println(znajdz(2, liczby));
6          System.out.println(znajdz(-33, liczby));
7      }
8      public static int znajdz(int wartosc, int[] tab) {
9          for (int i = 1; i <= tab.length; i++) {
10             if (tab[i] == wartosc) {
11                 return i;
12             }
13         }
14         return -1;
15     }
16 }
```

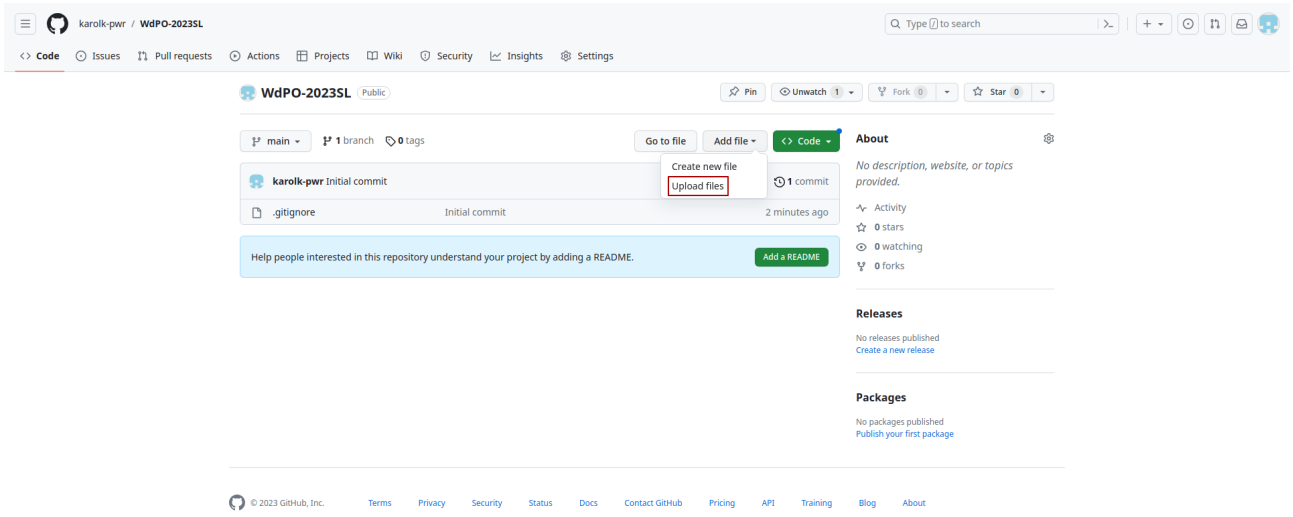
[1] https://www.jetbrains.com/help/idea/debugging-code.html
[2] https://youtu.be/59RC8gVPlvk?si=1T_jPlHl4ombtVS1

Git

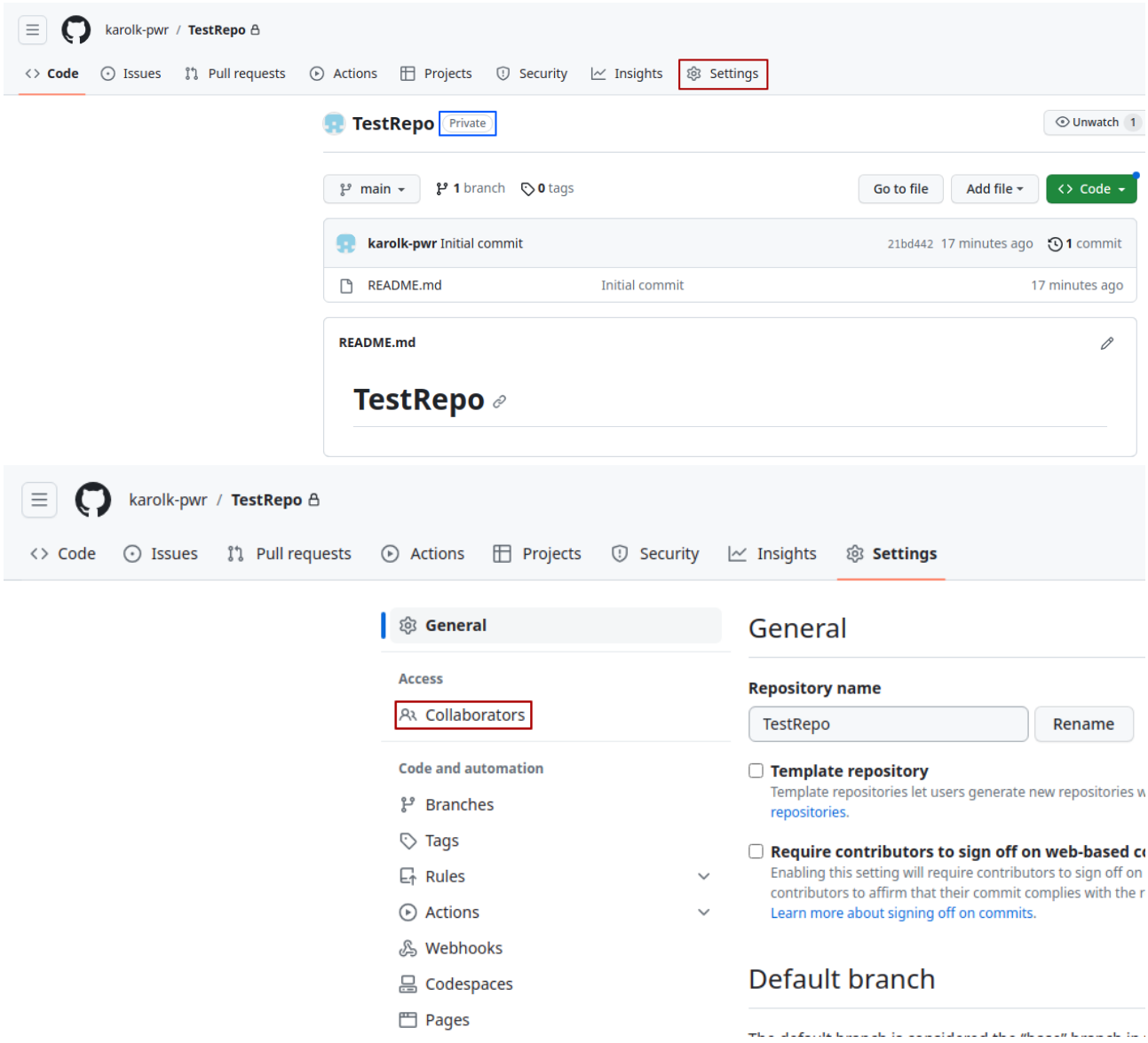
- **Git** jest narzędziem, które umożliwia stworzenie ogólnodostępnego **repozytorium** i pracę nad nim (tworzenie kolejnych wersji tego samego repozytorium o nowych własnościach jednocześnie archiwizując poprzednie wersje).
- Rozróżniamy dwa typy repozytoriów: **zdalne** i **lokalne**.
- Repozytorium zdalne to takie, które jest przechowywane na serwerze np. internetowym. W internecie istnieje wiele stron umożliwiających przechowywanie repozytoriów, są to m.in. **GitHub**, GitLab, Gerrit.
- Repozytorium zdalne możemy pobrać na swój komputer i uzyskujemy wtedy repozytorium lokalne. Przy pomocy takiego repozytorium możemy tworzyć wiele **branchy**, na których możemy dokonywać zmian w kodzie repozytorium. Takie operacje nie mają wpływu na repozytorium zdalne dopóki programista ich tam nie doda.



Rysunek 1: Tworzenie nowego repozytorium na GitHub



Rysunek 2: Dodawanie plików do repozytorium



Rysunek 3: Dodawanie użytkowników do repozytorium

Repozytorium lokalne i wybrane komendy

- `git --version`: zainstalowana wersja gita, tą komendą można sprawdzić czy jest poprawnie zainstalowany na komputerze.
- `git init`: tworzy puste repozytorium w danym folderze.
- `git status`: status aktualnego repozytorium.
  ?? - Untracked files, A - Files added to stage, M - Modified files, D - Deleted files.
- `git add plik.txt`: dodaje "plik.txt" do środowiska przejściowego.
  `git add -all` lub `git add -A`: dodaje wszystkie pliki w folderze.
- `git commit -m "First release"`: pierwszy commit, tzn. zatwierdzenie zmian i komentarz. Można też odrazu zacommitować wszystkie zmiany pomijając poprzedni krok:
  `git commit -a -m "Interestig commit"`.
- `git branch -M main`: stworzenie branchu "main".
- `git log`: historia commitów dla danego repozytroium.
- `git remote add origin <url repozytorium>`: łączenie ze zdalnym repozytorium.
- Przed pierwszym pushem należy skonfigurować dane commiter'a
  `git config --global user.name "username"`
  `git config --global user.email "user@email.com"`.
- `git push -u origin main`: przesłanie gałęzi „main" do zdalnego repozytorium.
- `git push -u origin master`: przesłanie zmian do głównej gałęzi zdalnego repozytorium.
- `git clone <url repozytorium>`: tworzy lokalną kopię zdalnego repozytorium.
- `git pull`: aktualizuje lokalne repozytorium.
- `git <command> -help`, `git help --all`: pomoc z gitem (komendy, opcje komend itd.).