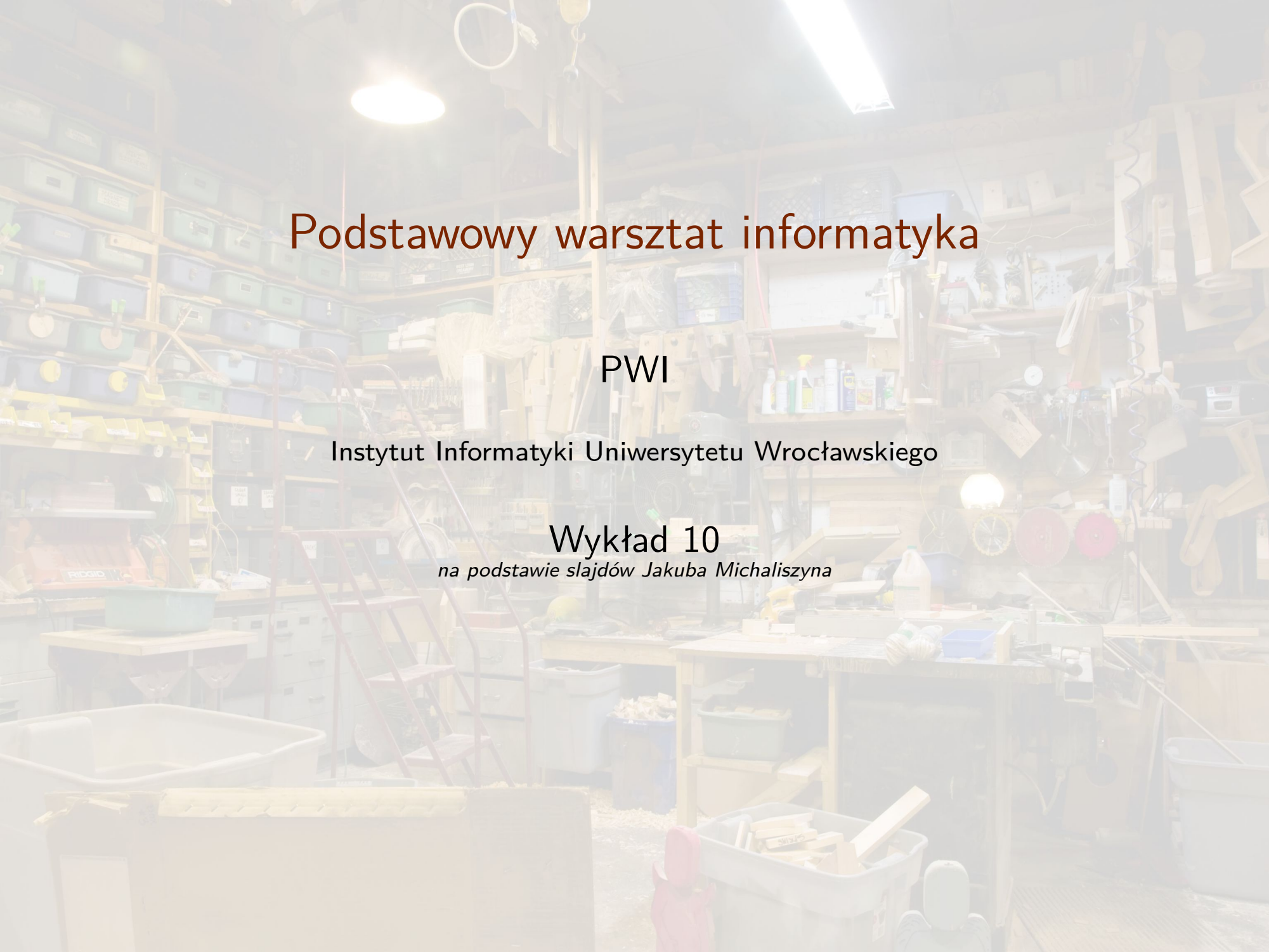

Podstawowy warsztat informatyka

PWI

Instytut Informatyki Uniwersytetu Wrocławskiego

Wykład 10

na podstawie slajdów Jakuba Michaliszyna

Jak ładnie wyświetlić historię?

```
git log --abbrev --oneline   --all --graph --decorate --color
```

Jak ładnie wyświetlić historię?

```
git log --abbrev --oneline   --all --graph --decorate --color
```

```
piotrek@piotrek-msi:~/Pulpit/pwi/git-repo/wyklad$ git log --abbrev --oneline   --all --graph --decorate --color
*   8d49110 (HEAD -> master, origin/master, origin/HEAD) Merge branch 'master' of https://github.com/iiuwr18/wyklad
|\
| * d484251 rpzwijamy główną gałąź
* | 2c3fcd0 rozwijamy główną gałąź
|/
| * 17fa820 (origin/boczna_gałąź, boczna_gałąź) boczna gałąź
|/
* 9ff08ad demonstracja jak zrobić kolejnego commita :)
* da6587d dopisek zrobiony na wykładzie
* 9270f38 Initial commit
```

Problem 1

Problem: Chcemy odrzucić lokalne, nieskomitowane zmiany.

Problem 1

Problem: Chcemy odrzucić lokalne, nieskomitowane zmiany.

Popularne rozwiązanie (niezbyt dobre?) `git stash`

Problem 1

Problem: Chcemy odrzucić lokalne, nieskomitowane zmiany.

Popularne rozwiązanie (niezbyt dobre?) `git stash`

git stash odkłada zmiany na później, nie kasuje ich.

Problem 1

Problem: Chcemy odrzucić lokalne, nieskomitowane zmiany.

Popularne rozwiązanie (niezbyt dobre?) `git stash`

git stash odkłada zmiany na później, nie kasuje ich.

Rozwiązanie: git checkout – pliki

reset/checkout/revert

Command	Scope	Common use cases
`git reset`	Commit-level	Discard commits in a private branch or throw away uncommited changes
`git reset`	File-level	Unstage a file
`git checkout`	Commit-level	Switch between branches or inspect old snapshots
`git checkout`	File-level	Discard changes in the working directory
`git revert`	Commit-level	Undo commits in a public branch
`git revert`	File-level	(N/A)

https://pl.atlassian.com/git/tutorials/resetting-checking-out-and-reverting

Problem 2

Chcemy obejrzeć, co zmieniliśmy.

Problem 2

Chcemy obejrzeć, co zmieniliśmy.

Częściowe rozwiązanie: git status

Problem 2

Chcemy obejrzeć, co zmieniliśmy.

Częściowe rozwiązanie: git status

Bardzo złe rozwiązanie: git commit i git push

Problem 2

Chcemy obejrzeć, co zmieniliśmy.

Częściowe rozwiązanie: git status

Bardzo złe rozwiązanie: git commit i git push

Nienajlepsze rozwiązanie: git checkout -b, git commit, git push (w forku).

Problem 2

Chcemy obejrzeć, co zmieniliśmy.

Częściowe rozwiązanie: git status

Bardzo złe rozwiązanie: git commit i git push

Nienajlepsze rozwiązanie: git checkout -b, git commit, git push (w forku).

Dobre rozwiązanie: git diff plik

Problem 3

Problem: chcemy usunąć/przenieść pliki tak, żeby git wiedział o tej operacji.

Problem 3

Problem: chcemy usunąć/przenieść pliki tak, żeby git wiedział o tej operacji.

git rm/mv
git mv plik1 plik 2 wykonuje:

- mv plik1 plik2
- git rm plik1
- git add plik2

Przy przenoszeniu plików należy zachować szczególną ostrożność!

Problem 3

Problem: chcemy usunąć/przenieść pliki tak, żeby git wiedział o tej operacji.

git rm/mv
git mv plik1 plik 2 wykonuje:

- mv plik1 plik2
- git rm plik1
- git add plik2

Przy przenoszeniu plików należy zachować szczególną ostrożność!

`git rm --cached plik` usuwa `plik` z git-a, ale nie z katalogu roboczego

Problem 4

Problem: chcemy usunąć lokalny commit.

Problem 4

Problem: chcemy usunąć lokalny commit.

`git reset HEAD~3` zachowuje lokalne zmiany
`git reset --hard HEAD~3` usuwa lokalne zmiany (wszystkie!).

Problem 5

Problem: chcemy usunąć zdalny commit.

To jest prawie zawsze zły pomysł. Chyba, że pracujemy sami.

Problem 5

Problem: chcemy usunąć zdalny commit.
To jest prawie zawsze zły pomysł. Chyba, że pracujemy sami.

git revert HEAD
git revert -n HEAD – bez dodatkowego commita

Problem 6

Problem: chcemy ustrzec się przed przypadkowym dodaniem niektórych plików, np. baza.db, config.php itd.
Rozwiązanie: dodać pliki go .gitignore. Przykładowy plik:

```
*.dll
*.exe
*.o
*.so
*.7z
*.log
*.sql
*.sqlite
.DS_Store?
```

.gitignore nie musi być komitowany ale jest to dobrą praktyką, np. pozwala współpracownikom ignorować pliki tworzone automatycznie w czasie rozwijania/uruchamiania projektu

Problem 7

Problem: coś się popsuło, ale nie wiadomo kiedy.

Problem 7

Problem: coś się popsuło, ale nie wiadomo kiedy.

Rozwiązanie: przeszukiwanie binarne.

```
git bisect start          # rozpoczęcie procesu
git bisect bad            # oznaczamy obecną wersję jako złą
git bisect good revision # oraz oznaczamy ostatnią
                            znaną dobrą wersję
```

Potem powtarzac do rozwiązania:

```
git bisect good lub git bisec bad
```

Git hooks

Problem: chcemy zautomatyzować pewne rzeczy.

Git hooks

Problem: chcemy zautomatyzować pewne rzeczy.

Pliki wykonywalne dodajemy do .git/hooks
Nazwa skryptu powinna odpowiadać zdefiniowanej, np. skrypt
`commit-msg` przetwarza komentarze commitów
Przykład: każdy opis commita zawiera numer biletu (ticketa), np. “#123“.

```
#!/usr/bin/env ruby
message = File.read(ARGV[0])

unless message =~ /\\s*\#\\d+/
  puts "[POLICY] Nie podałeś numeru biletu."
  exit 1
end
```

Usuwanie danych wrażliwych

Problem: dodaliśmy do repozytorium jakiś wrażliwy plik, np. bazę danych albo plik z hasłem.

Usuwanie danych wrażliwych

Problem: dodaliśmy do repozytorium jakiś wrażliwy plik, np. bazę danych albo plik z hasłem.

```
git filter-branch --force --index-filter
'git rm --cached --ignore-unmatch secrets.txt'
--prune-empty --tag-name-filter cat -- --all
```

Mam inny problem

`https://github.com/k88hudson/git-flight-rules`

Mam inny problem

Google i stackoverflow ftw.