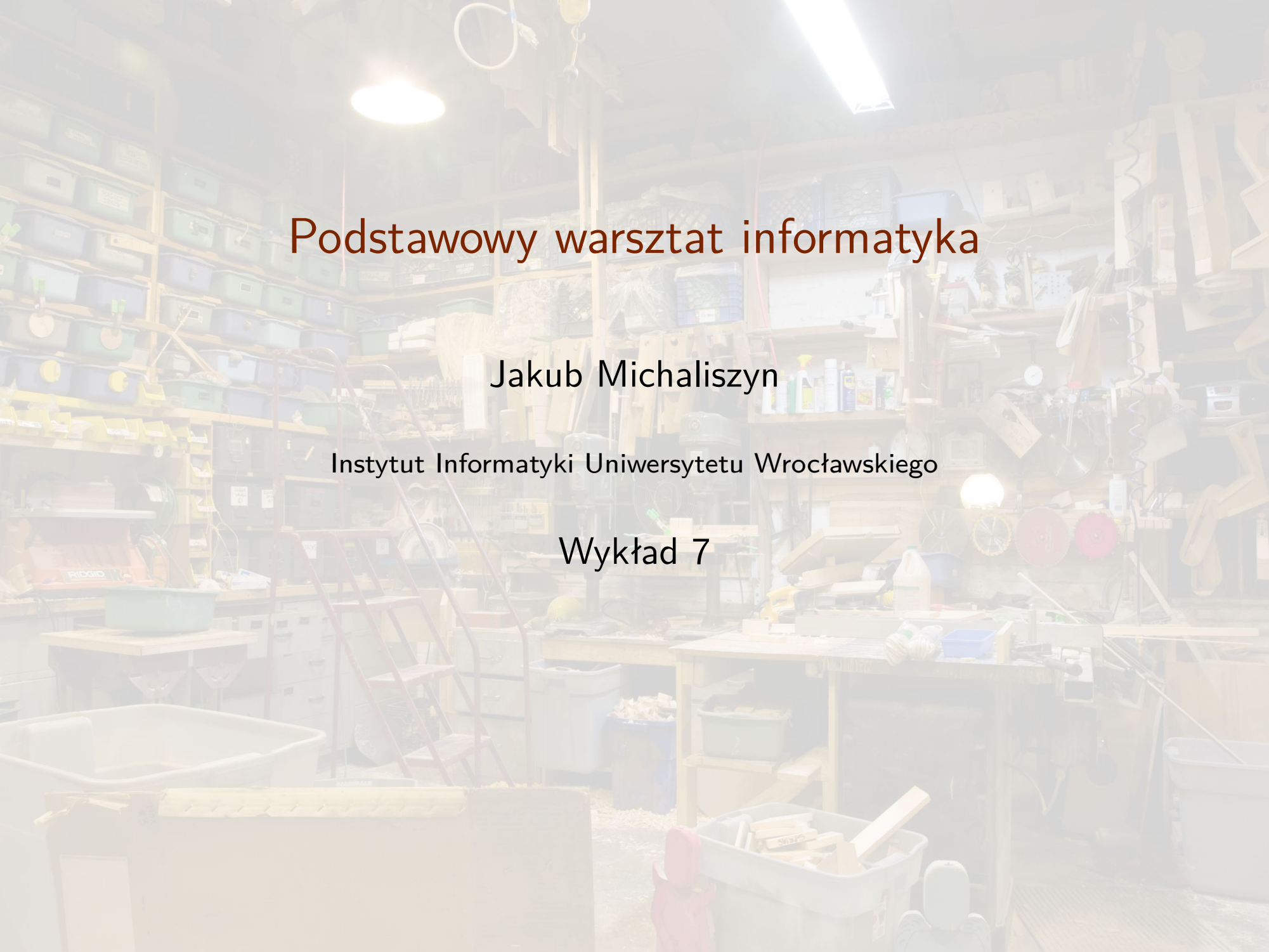

Podstawowy warsztat informatyka

Jakub Michaliszyn

Instytut Informatyki Uniwersytetu Wrocławskiego

Wykład 7

Ogłoszenia

Terminarz
12-20 stycznia: rozwiązywanie list 11-12, w czasie pracowni można przyjść na konsultacje.
16 stycznia, 15:45: Przesłanie rozwiązania zadania bonusowego 9.4 (CV).
20 stycznia: Wykład 8 - Git na poziomie plików
20 stycznia: Lista 12 - przesyłanie rozwiązań (do południa).
23 stycznia, 15:45: Przesłanie rozwiązania zadania bonusowego 9.5 (kupony).
23-27 stycznia: Lista 13 (wszystkie grupy).
27 stycznia: Wykład 9 - kompilowanie i debugowanie.
2 lutego: Lista 15/specjalna w grupach jotop i karp.
3 lutego: Kolokwium dla tych, którzy nie zdobyli 54% punktów.
3 lutego, do południa: Lista 14 - przesłanie rozwiązań.

Przypomnienie – forki i pull requesty

Problem: chcemy współpracować z różnymi ludźmi. Chcemy, aby owoce ich pracy lądowały w naszym repozytorium, ale żeby nie mogli nic popsuć.

Przypomnienie – forki i pull requesty

Problem: chcemy współpracować z różnymi ludźmi. Chcemy, aby owoce ich pracy lądowały w naszym repozytorium, ale żeby nie mogli nic popsuć.

Rozwiązanie: współpracownicy tworzą kopię repozytorium (forka), a po skończonej pracy proszą o to, byśmy wciągnęli ich kod do naszego repozytorium (pull request).

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

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 remove 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 remove plik1
- git add plik2

Przy przenoszeniu plików należy zachować szczególną ostrożność!

git reset plik – usuwa z gita, ale nie z komputera

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?
.Trashes
ehthumbs.db
Thumbs.db
```

Problem 7

Problem: Musimy wielokrotnie rozwiązywać te same konflikty (bo np.powtarzamy merge).

Problem 7

Problem: Musimy wielokrotnie rozwiązywać te same konflikty (bo np.powtarzamy merge).

REuse REcorded REsolution feature
git config –global rerere.enabled true

Problem 8

Problem: coś się popsuło, ale nie wiadomo kiedy.

Problem 8

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.
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

Google i stackoverflow ftw.