PB162 Programovanie v jazyku Java I

1. Úvod

Jakub Smadiš

13. februára 2022

Organizačné pokyny

- základné informácie sú dostupné na wiki
- na riešenie nejasností používajte diskusné fórum
- cibuľová výuka – čo sa naučíme jeden týždeň použijeme v tom ďalšom
- predmet je časovo náročný

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

(b) *Čo znamená správa verzií?*

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

(b) *Čo znamená správa verzií?*

Zaznamenávanie zmien súborov v čase. Užívateľ sa tak môže vždy vrátiť k pôvodnej verzii, pozrieť si čo sa odvtedy zmenilo, atď.

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

(b) *Čo znamená správa verzií?*

Zaznamenávanie zmien súborov v čase. Užívateľ sa tak môže vždy vrátiť k pôvodnej verzii, pozrieť si čo sa odvtedy zmenilo, atď.

(c) *A čo je potom GitHub?*

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

(b) *Čo znamená správa verzií?*

Zaznamenávanie zmien súborov v čase. Užívateľ sa tak môže vždy vrátiť k pôvodnej verzii, pozrieť si čo sa odvtedy zmenilo, atď.

(c) *A čo je potom GitHub?*

Webová služba, ktorá poskytuje webhosting gitových projektov, takže sa k nim dostanete odvšadiaľ (ak máte pripojenie na internet). Podobnou službou je napríklad GitLab alebo Bitbucket.

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

(b) *Čo znamená správa verzií?*

Zaznamenávanie zmien súborov v čase. Užívateľ sa tak môže vždy vrátiť k pôvodnej verzii, pozrieť si čo sa odvtedy zmenilo, atď.

(c) *A čo je potom GitHub?*

Webová služba, ktorá poskytuje webhosting gitových projektov, takže sa k nim dostanete odvšadiaľ (ak máte pripojenie na internet). Podobnou službou je napríklad GitLab alebo Bitbucket.

(d) *Na čo nám to je?*

Pár otázok na ujasnenie

(a) *Čo je to Git?*

Distribuovaný systém **správy verzí**.

(b) *Čo znamená správa verzií?*

Zaznamenávanie zmien súborov v čase. Užívateľ sa tak môže vždy vrátiť k pôvodnej verzii, pozrieť si čo sa odvtedy zmenilo, atď.

(c) *A čo je potom GitHub?*

Webová služba, ktorá poskytuje webhosting gitových projektov, takže sa k nim dostanete odvšadiaľ (ak máte pripojenie na internet). Podobnou službou je napríklad GitLab alebo Bitbucket.

(d) *Na čo nám to je?*

Na bezpečné prepisovanie kódu, na paralelné tímové programovanie, záloha, na zistenie kto napísal tak hrozný kód vo veľkom projekte...

Základné príkazy

`git commit`

- slúži na vytvorenie kontrolného bodu/snapshotu, ku ktorému sa môžme kedykoľvek vrátiť
- bod je vytvorený (iba) na lokálnom počítači
- každý commit má svoju správu, časovú známku a unikátny hash
- commit má v sebe uložené zmeny oproti predošlému commitu

`git push`

- nahrá commity do vzdialeného repozitáru (ktorý sa nachádza napríklad na GitLabe)

`git pull`

- stiahne zmeny (commity) zo servera do lokálneho adresára
- niečo podobné ako `git clone`, ale nevytvára nový priečinok, iba ho aktualizuje so serverovým

Sledovanie súborov

Pri vytváraní nových súborov ich `git` automaticky nesleduje. Súbor zahrnieme do sledovania príkazom `git add <file>`.
Ak chceme pridať všetky súbory, môžme použiť prepínač `-A` alebo `--all`.

Poznámka (RTFM)

Strácate sa v príkazoch? Na zistenie informácii o príkaze použite

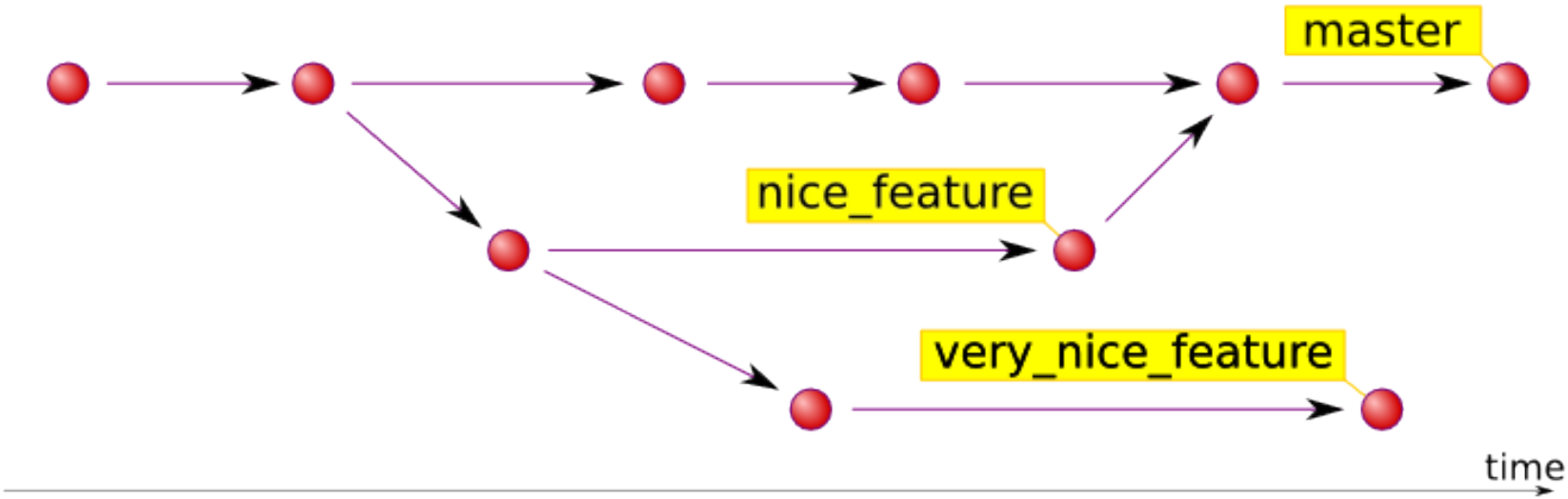
```
man git-<prikaz>
```

napr.

```
man git-commit
```

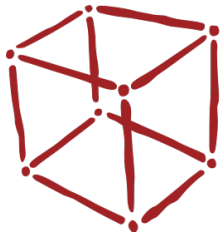
Vetvy (branches)

Chceme paralelne vyvíjať rôzne časti na jednom projekte.



Každá vetva má vlastné commity, medzi vetvami sa dá ľubovoľne prepínať.
Vetvy môžme rozdeliť (vytvoriť novú vetvu) alebo spojiť (merge).
Hlavná vetva sa volá **master**.
Vetvu vytvoríme príkazom `git branch <názov>` a prepíname sa medzi nimi pomocou `git checkout <názov>`.

Integrated development environment – IDE



NetBeans

IntelliJIDEA

eclipse

Úvod do OOP

Motivácia: chceme rozumne ukladať štruktúry, aby sa s nimi dalo lepšie pracovať.

Chceme vytvoriť/popísať objekt pes. Čo všetko by mal obsahovať?

Úvod do OOP

Motivácia: chceme rozumne ukladať štruktúry, aby sa s nimi dalo lepšie pracovať.

Chceme vytvoriť/popísať objekt pes. Čo všetko by mal obsahovať?

to, čo pes má: hlavu, oči, 4 nohy, meno, ...

to, čo pes robí: štekanie, behanie, vrtenie chvostom, ...

Úvod do OOP

Motivácia: chceme rozumne ukladať štruktúry, aby sa s nimi dalo lepšie pracovať.

Chceme vytvoriť/popísať objekt pes. Čo všetko by mal obsahovať?

to, čo pes má: hlavu, oči, 4 nohy, meno, . . . **atribúty**

to, čo pes robí: štekanie, behanie, vrtenie chvostom, . . . **metódy**

Pseudopes – intuícia

```
class Dog {

 String name;
 int age;

 String welcomeMaster() {
   return "Woof I am " + name;
 }

 String barkOthers() {
   return "Wrr HAF HAF";
 }
}
```

Pseudopes – intuícia

```
class Dog {

 String name;
 int age;

 String welcomeMaster() {
   return "Woof I am " + name;
 }

 String barkOthers() {
   return "Wrr HAF HAF";
 }
}
```

Čo ak

- chceme overiť, aby sa vek vždy nastavil na nezáporné číslo?
- nechceme zverejniť meno nášho psa?
- chceme obmedziť viditeľnosť metód?

Pseudopes – pokračovanie

```
public class Dog {

  private String name;
  private int age;

  public String welcomeMaster() {
    return "Woof I am " + name;
  }

  public String barkOthers() {
    return "Wrr HAF HAF";
  }

  public void setAge(int newAge) {
    if newAge > 0
      age = newAge;
  }

  public int getAge() { return age; }
}
```

Trieda je verejná – každý môže vyrábať psov.

Atribúty sú privátne, iba metódy v triede môžu s nimi pracovať.

Metódy na prácu s atribútmi nazývame gettery a settery.

Ako však vytvoríme konkrétneho psa?

```
<viditeľnosť> <návratový typ> <názov>(<parametre>)
```

Konštruktory

1. Konštruktor bez parametrov (vytvorí sa sám, ak neexistuje žiadny)

```
public Dog() { } // v triede Dog


Dog staryDunco = new Dog(); // v inej triede
staryDunco.setName("Dunco");
staryDunco.setAge(13);
```

2. Konštruktor s parametrami

```
public Dog(String ourName, int ourAge) {
  name = ourName;
  age = ourAge; // privatne atributy su name a age
}

Dog staryDunco = new Dog("Dunco", 13);
```

Premenná `staryDunco` sa nazýva **inštanciou triedy** Dog.

Vytváranie primitívnych typov vs. vytváranie objektov

Primitívne typy

- *int, boolean, char, double, . . .*
- premenná môže byť vytvorená "ihneď"
- každý typ má defaultnú hodnotu: int má 0, double 0.0, atď.

Objekty

- zvyčajne si vytvárame vlastné – *Dog, Car, Wheel, Main, Math, . . .*
- premenná musí byť najprv **skonštruovaná**, napr. auto si musí najprv vytvoriť kolesá
- objektová premenná má defaultne hodnotu `null`
- premenné sú prakticky ukazatele, `null` hovorí "ukazuj na nič"

Poznámka

Objekt začína na rozdiel od primitívnych typov veľkým písmenom.

Dog vs. int

Komentáre sú užitočné

Javadoc

```java
/**
 *
 * Method used to return reference to dog with given id.
 *
 * @param   id    an id of the dog we want to get
 * @return        the dog we were looking for
 */
public Dog findDog(int id) {
        return dogs[id];
}
```