Java tečaj (2. dio)

Osnove implementacije objektnog modela u Javi
(i jezicima C++, C# i sličnima)

[nadopuna glavne prezentacije]

Marko Čupić

Implementacija objektnog modela

- Slike i stanje u memoriji odnose se na primjer iz glavne prezentacije, slide 9

Implementacija objektnog modela

- **Razred** je nacrt prema kojem se konstruiraju **primjerci razreda** (objekti)
- U memoriji postoji na jednom mjestu bajtkod svih metoda definiranih u razredu
- Sve statičke metode prevode se upravo kako su definirane
  - Ako je statička metoda **`m`** definirana u razredu **`R`**, pozivamo je s **`R.m(argumenti)`**.

Implementacija objektnog modela

- Sve nestatičke metode automatski se proširuju argumentom imena **`this`** koji je po tipu referenca na primjerak razreda u kojem je metoda definirana
  - Kada pozivamo **`obj.m(x,y,z)`** zapravo se "ispod haube" poziva **`m(obj,x,y,z)`** i ta umetnuta referenca u metodi je vidljiva kao ključna riječ **`this`**
  - Posljedica: iste nije moguće pozivati ako ne postoji objekt nad kojim bismo je pozvali

Implementacija objektnog modela

- Nestatičke članske varijable definirane u razredu konceptualno pripadaju primjerku razreda (objektu)
- Svaki puta kada operatorom **`new`** alociramo novi primjerak razreda, u memoriji se zauzima mjesto za pohranu vrijednosti svih nestatičkih članskih varijabli
  - Kažemo da svaki objekt ima vlastitu kopiju takvih varijabli

Implementacija objektnog modela

- Razlog zašto se nestatičkim metodama mora proslijediti referenca na objekt nad kojim su pozvane jest taj da one imaju pristup nestatičkim članskim varijablama definiranim u razredu
  - Sjetite se metode
    `String getIme() { return this.ime; }`
  - Koju to varijablu **`ime`** ta metoda vraća? Metoda, iako formalno ne deklarira da prima **`this`**, mora ga primiti kako bi znala od kojeg objekta treba dohvatiti i vratiti ime

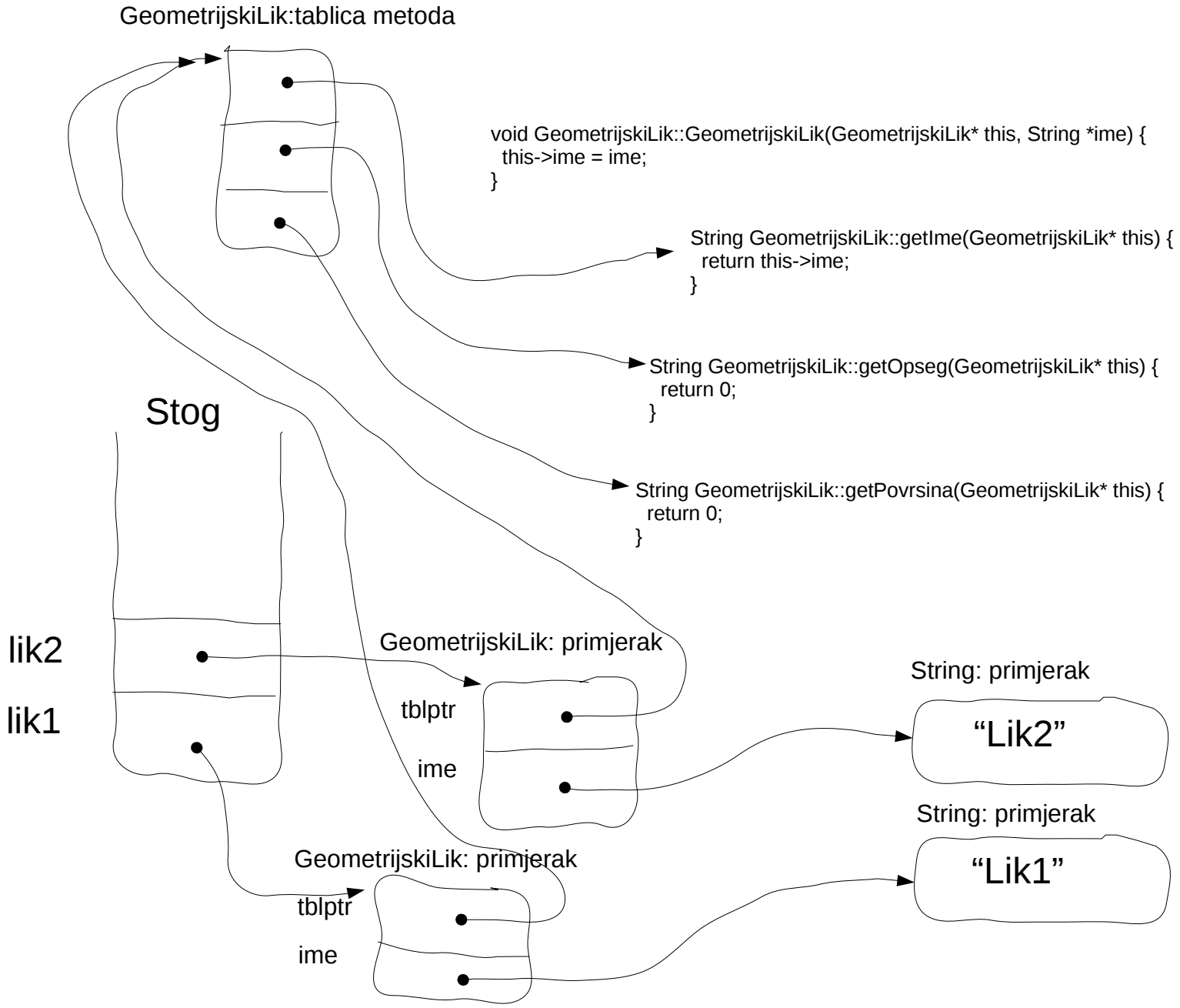
Implementacija objektnog modela

- Opisani mehanizam automatskog proširenja popisa argumenata metode (u implementaciji) i slanja reference na objekt omogućava nam da se pravimo kao da metodu pozivamo nad objektom i da ona pripada objektu.
  - To međutim može zbuniti pa oprez: nema svaki objekt kopiju strojnog koda definiranih metoda – u memoriji je strojni kod svake metode samo na jednom mjestu a nad kojim objektom treba obaviti operaciju, doznaje preko predanog argumenta **`this`**

Implementacija objektnog modela

- Dodatno, za svaki razred, u memoriji postoji po jedna tablica pokazivača na bajtkod njegovih nestatičkih metoda (dakle, memorijsko zauzeće je jedna tablica po razredu, ne objektu)
- Svaki primjerak razreda (objekt) uz mjesto za pohranu vrijednosti nestatičkih članskih varijabli još zauzima mjesto i za pohranu pokazivača na tu tablicu pokazivača na implementacije metoda
- U memoriji zauzetoj za objekt, taj je pokazivač uobičajeno na samom početku

Implementacija objektnog modela



Implementacija objektnog modela

- Kada prevodilac u kodu naiđe na poziv nestatičke metode nad objektom preko reference tipa R:
  `obj.m(x, y, z)`
  on potraži indeks u tablici pokazivača na implementacije metoda (neka je to **`k`**), i generira indirektni poziv metode:
  `(obj->tblptr[k])(obj, x, y, z)`
- Ovakav način pozivanja omogućit će dinamički polimorfizam pri nasljeđivanju: izvedeni razredi moći će ponuditi nove implementacije metoda