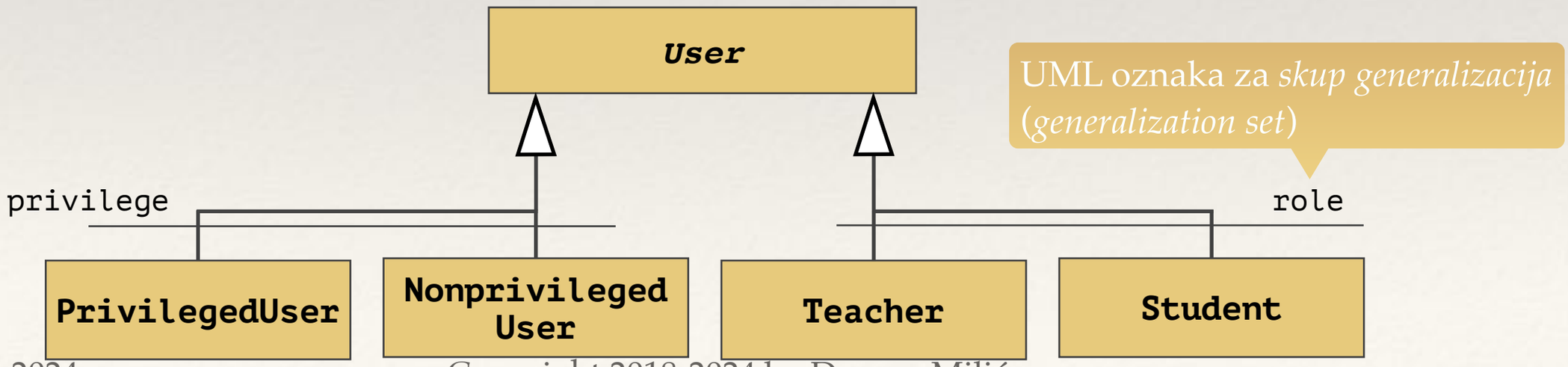
# Hijerarhijska dekompozicija

- U skupovnoj logici, osnovna klasa (generalizacija) predstavlja nadskup skupa objekata njene izvedene klase (specijalizacije). Ali šta ako je osnovna klasa apstraktna?
- Pošto ona tada nema svoje direktne instance, ne postoje elementi tog skupa koji nisu ujedno i elementi skupa predstavljenog nekom od izvedenih klasa. Prema tome, apstraktna klasa predstavlja *uniju* skupova predstavljenih izvedenim klasama
- A šta ako izvedene klase, posmatrane kao skupovi, imaju presek, odnosno zajedničke elemente?
- Ovo se obično dešava ako se generalizacije/specijalizacije prave po različitim, ortogonalnim kriterijumima
- Na primer, u nekom školskom sistemu, *korisnici* (*User*) mogu biti klasifikovani po pravima pristupa na *privilegovane* (*Privileged User*) i *neprivilegovane* (*NonprivilegedUser*); sa druge strane, mogu se klasifikovati prema svojoj ulozi (*role*) na *nastavnike* (*Teacher*) i *učenike* (*Student*)
- Na jeziku UML, ovakve različite grupe relacija generalizacija/specijalizacija nazivaju se *skupovima generalizacija* (*generalization set*)
- Ako su izvedene klase kao skupovi disjunktni, tj. nemaju presek (zajedničke instance), skup generalizacija naziva se *isključiv* (*exclusive*); ako je takva osnovna klasa apstraktna, ona predstavlja uniju disjunktnih podskupova - *particiju* (*partition*)



# Hijerarhijska dekompozicija

- Međutim, izvedene klase iz različitih skupova generalizacija često imaju zajedničke instance, odnosno presek. Na primer, jedan nastavnik može biti privilegovan korisnik
- U nekim jezicima, kao što je UML, objekat može biti instanca više klasa (koje nisu u relaciji generalizacije/specijalizacije). Štaviše, objekat se može *dinamički reklasifikovati* (*reclassify*) tokom svog životnog veka: mogu mu se dodavati i oduzimati klase kojima pripada (time se dodaju ili oduzimaju sva svojstva tih klasa)
- U tradicionalnim, statički tipiziranim OO programskim jezicima, kakav je i C++, ovo nije podržano, pa objekat uvek mora biti direktna instanca jedne i samo jedne klase. Ta klasa se uvek mora odrediti u trenutku kreiranja tog objekta i objekat se ne može reklasifikovati tokom svog životnog veka
- Kako onda rešiti ovakvu situaciju? Višestrukim izvođenjem nove klase koja predstavlja presek dve osnovne:

```
class User {…};

class PrivilegedUser : public User {…};
class NonprivilegedUser : public User {…};

class Teacher : public User {…};
class Student : public User {…};

class PrivilegedTeacher : public PrivilegedUser, public Teacher {…};
```