

Објектно-орјентисано програмирање



Владимир Филиповић
vladaf@matf.bg.ac.rs

Александар Картељ
kartelj@matf.bg.ac.rs

Објектно-орјентисана парадигма



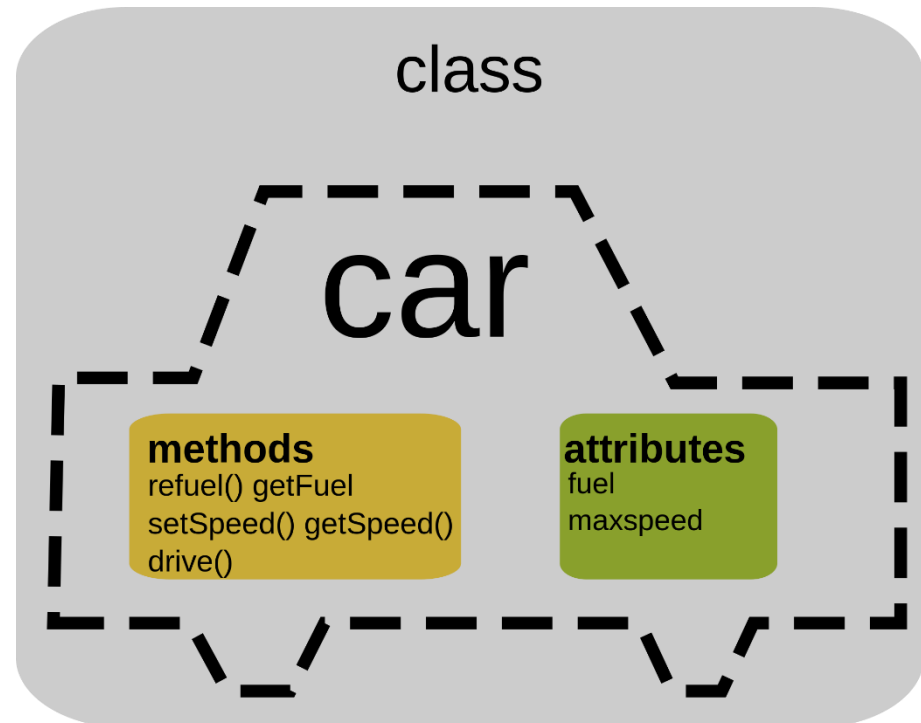
Владимир Филиповић
vladaf@matf.bg.ac.rs

Александар Картељ
kartelj@matf.bg.ac.rs



Објектно-оријентисано програмирање

- **Објектно-оријентисано програмирање (ООП)** је програмска парадигма заснована на појму објектата, који могу да садрже податке, тзв. **атрибуте**, као и кодове који описују неке активности, тзв. **методе**.
- У примеру са слике, објекат је **ауто**.
- Атрибути су:
 - Гориво
 - Максимална брзина
 - ...
- Методе су:
 - Допуни гориво
 - Подеси брзину
 - Вози
 - ...





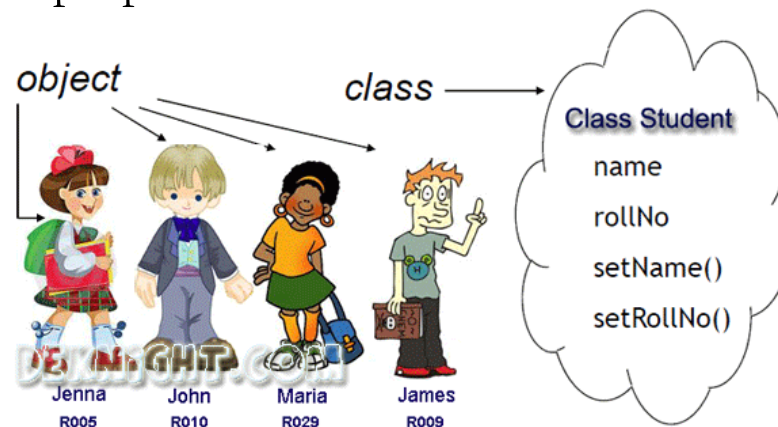
Основни појмови

- **Објекат:** интегрална целина података и процедура за рад са њима (*).
- **Атрибут** (особина) објекта су подаци унутар објекта.
- **Метод** - функција која је саставни део објекта, тј. поступак којим се реализује порука упућена објекту. Методи описују понашање објекта.
- **Порука** - скуп информација који се шаље објекту. Састоји се из адресе (објекта примаоца поруке) и саопштења (казује шта треба да се уради).



Основни појмови (2)

- **Класа** - скуп објеката са заједничким својствима, који се понашају на исти начин.
 - Класа дефинише шаблон за креирање објеката, тј описује структуру објекта.
- **Примерак** (инстанца) класе - конкретан објекат дате класе.
 - У Јави је сваки објекат примерак неке класе, а објекти постоје само током извршавања програма.



Илустрација објекта и класе



Основни појмови (3)

- Класом је дефинисан тип објекта, док свака инстанца класе има конкретну вредност атрибута, на пример:

tacka (x,y, boja)

- t1(4,5,crvena)
- t2(0,0,zuta)

automobil(broj_sasije, broj_motora, boja, broj_sedišta)

- a1(4534332, 1X467,plava,4)
- a2(4321564, B3536, bela, 5)

- Понашање објекта одређено је методима у класи који могу дејствовати на тај објекат, на пример:
Врати x-координату тачке, помери тачку за dx, за dy,...
Покрени возило, промени брзину, скрени лево, заустави...



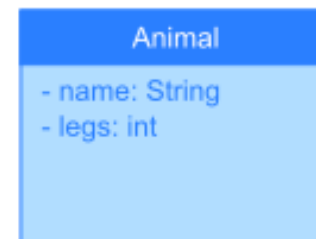
Основни појмови (4)

- Класа B је **поткласа** (класа потомак, изведена класа) класе A ако су сви примерци класе B истовремено и примерци класе A .
 - За класу A кажемо да је **наткласа** (родитељска класа) класе B .
- Поткласе представљају даљу конкретизацију наткласе.
 - Оне настају додавањем нових својстава (атрибута и/или метода) или модификовањем постојећих својстава наткласе.
- Ако је неки објекат примерак класе B , онда је он истовремено и примерак њене наткласе A .
- **Наслеђивање**- механизам за креирање нових класа из постојећих.
 - Наслеђивањем се формирају релације између класа.



Основни појмови (5)

- Наслеђивање описује односе:
 1. **«јесте конкретизација»** када се посматра са позиције поткласе
 - нпр. «Студент јесте конкретизација од Човек».
 2. **«јесте уопштење»** када се посматра са позиције надкласе.
 - Нпр. «Човек јесте уопштење од Студент».
- Поред наслеђивања, битан однос међу класама је и **садржавање** – када објекат једне класе као свој део садржи објекат друге класе:
 1. **«је део од»** ако се посматра са позиције објекта-дела
 - нпр. «Мотор је део од Ауто».
 2. **«садржи»** ако се посматра са позиције објекта-садржаоца.
 - На пример, «Ауто садржи Мотор».

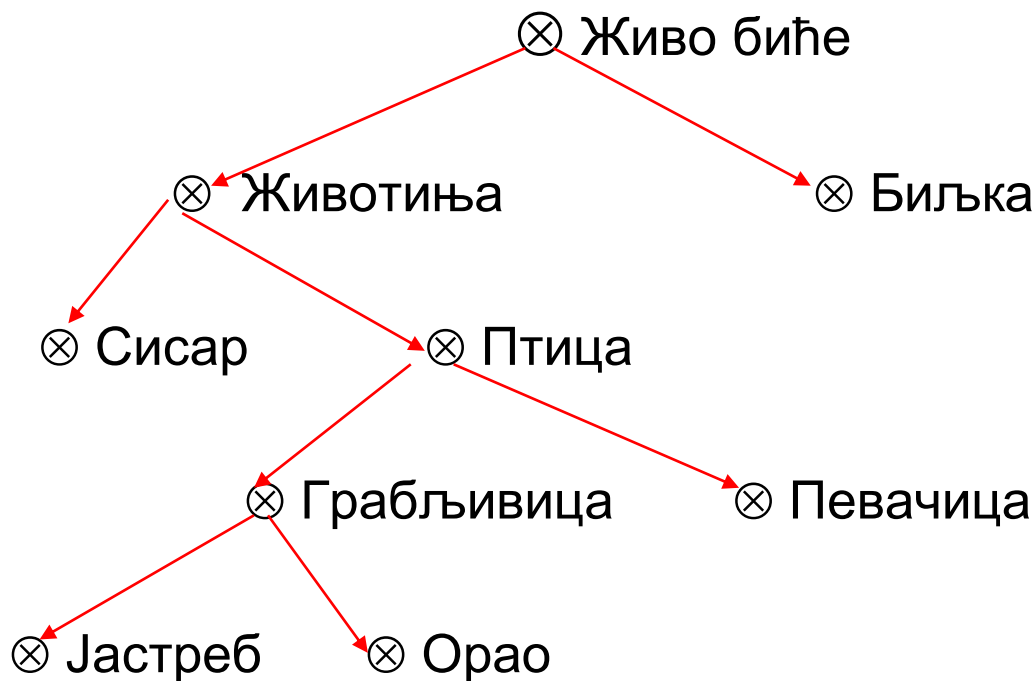


Дијаграм класе



Примери наслеђивања

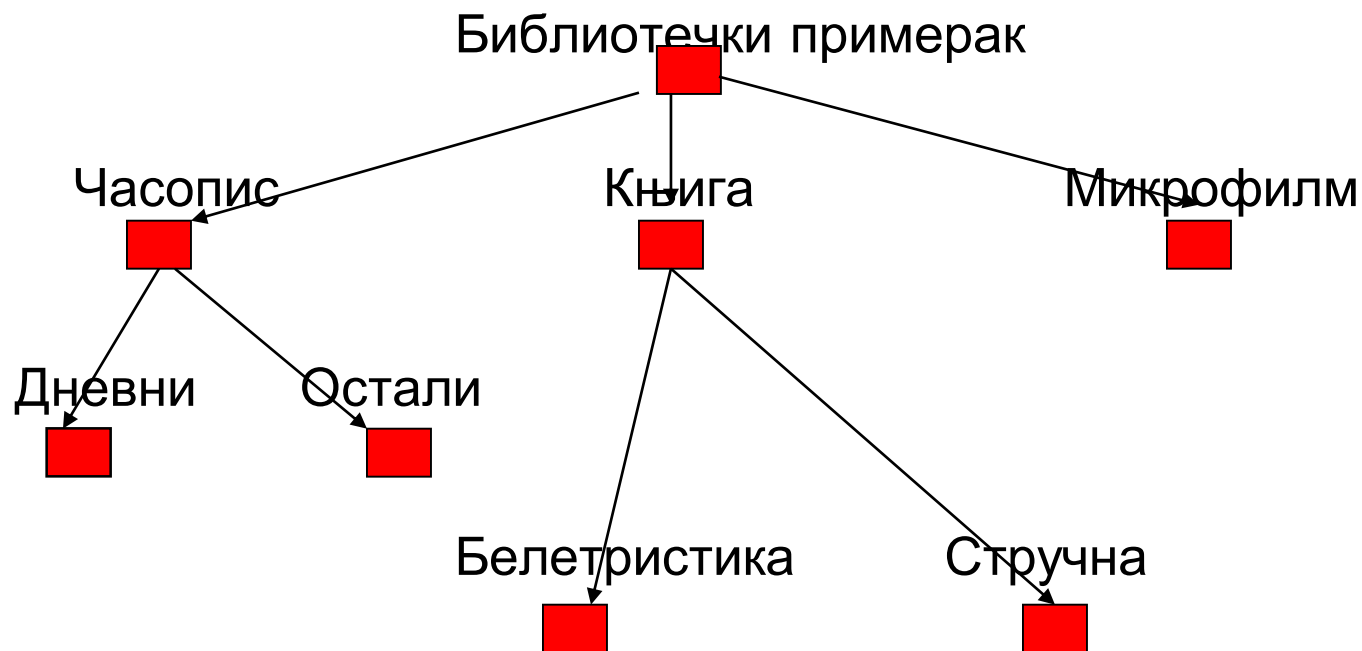
Наслеђивањем се формира хијерархија класа.





Примери наслеђивања (2)

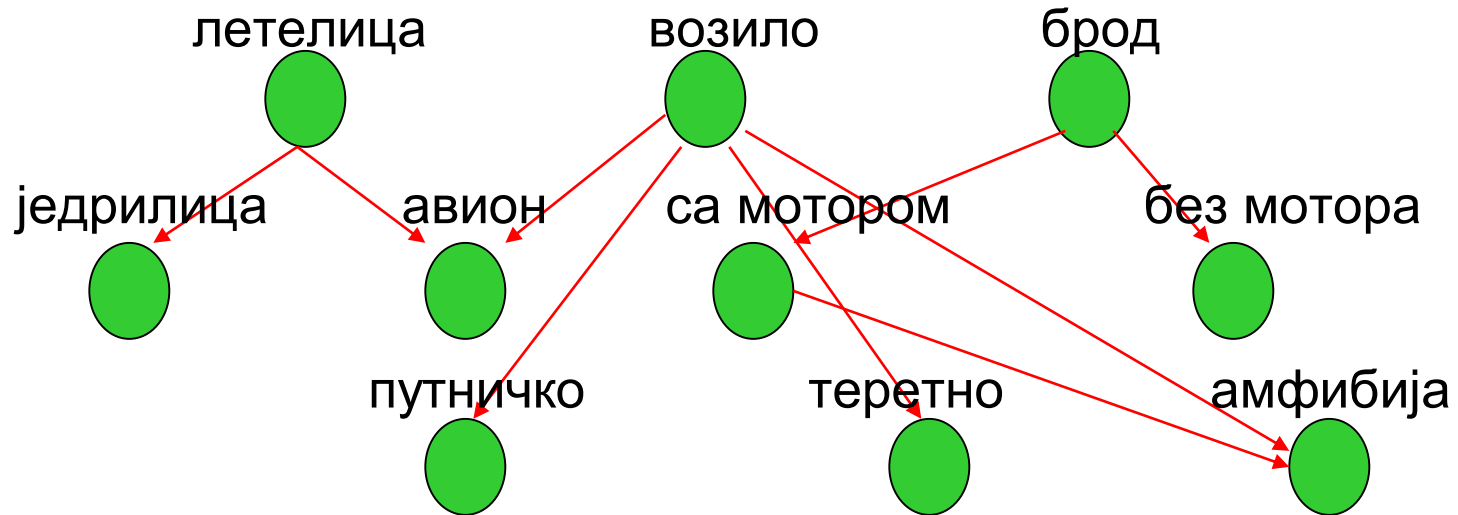
Још један пример хијерархијског наслеђивања:





Примери наслеђивања(3)

Пример вишеструког наслеђивања - класа може имати више директних наткласа.



Вишеструко наслеђивање није подржано у Јава-језику, али јесте у језику C++.



Зашто је објектно-оријентисан концепт доживео велики успех?

- Погодан за: анализу, пројектовање и програмирање.
- Олакшано одржавање софтвера
- Омогућава лако и једноставно уклапање модула.
- Поновна искористивост софтвера.
- Најпогоднији за симулирање догађаја.

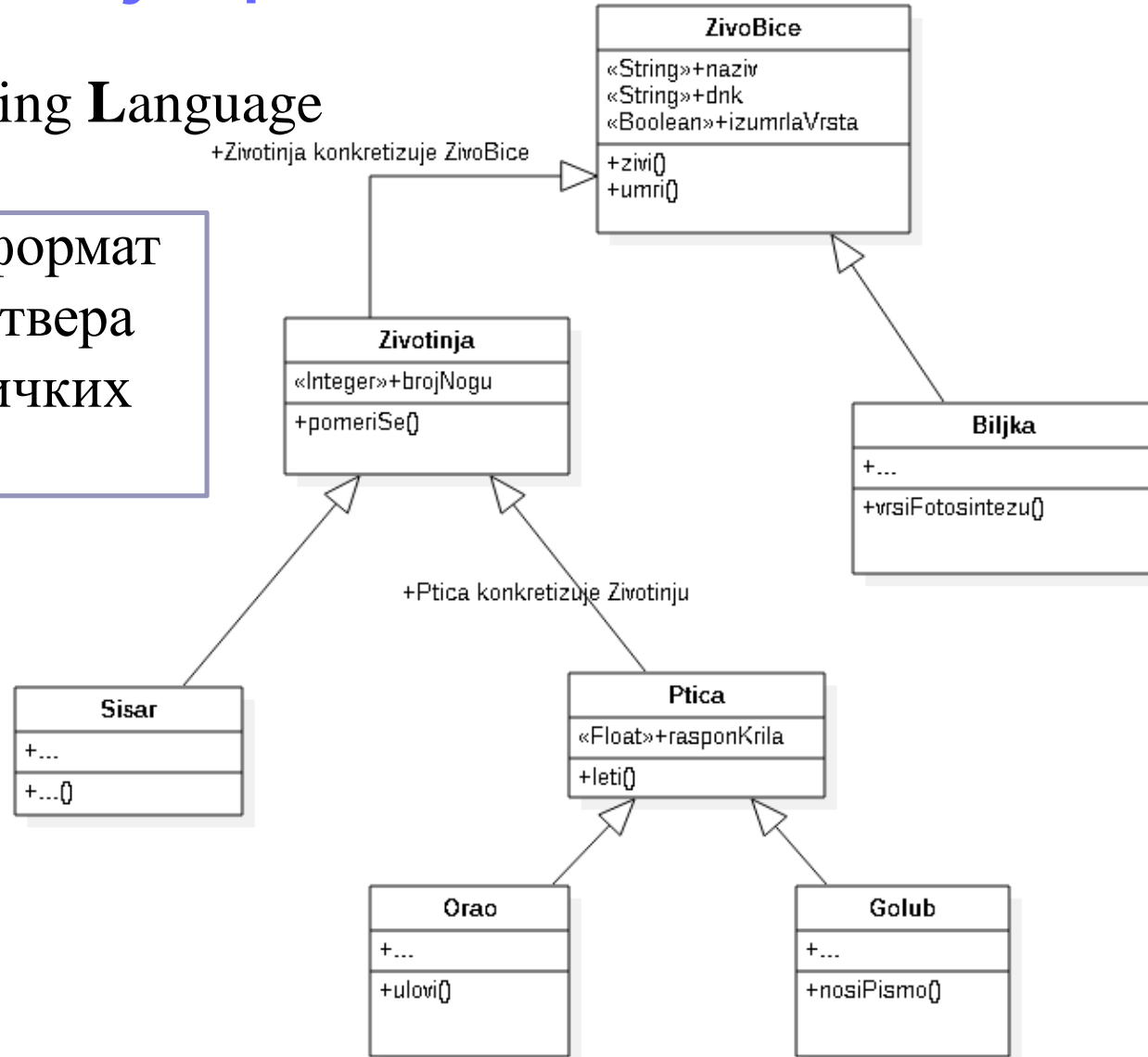


УМЛ дијаграми класа

- енг. **Unified Modelling Language**
- Укратко:

Стандардизовани формат за моделовање софтвера коришћењем графичких облика и текста.

- Постоје различити типови дијаграма.
- Тренутно нас интересују само дијаграми класа.





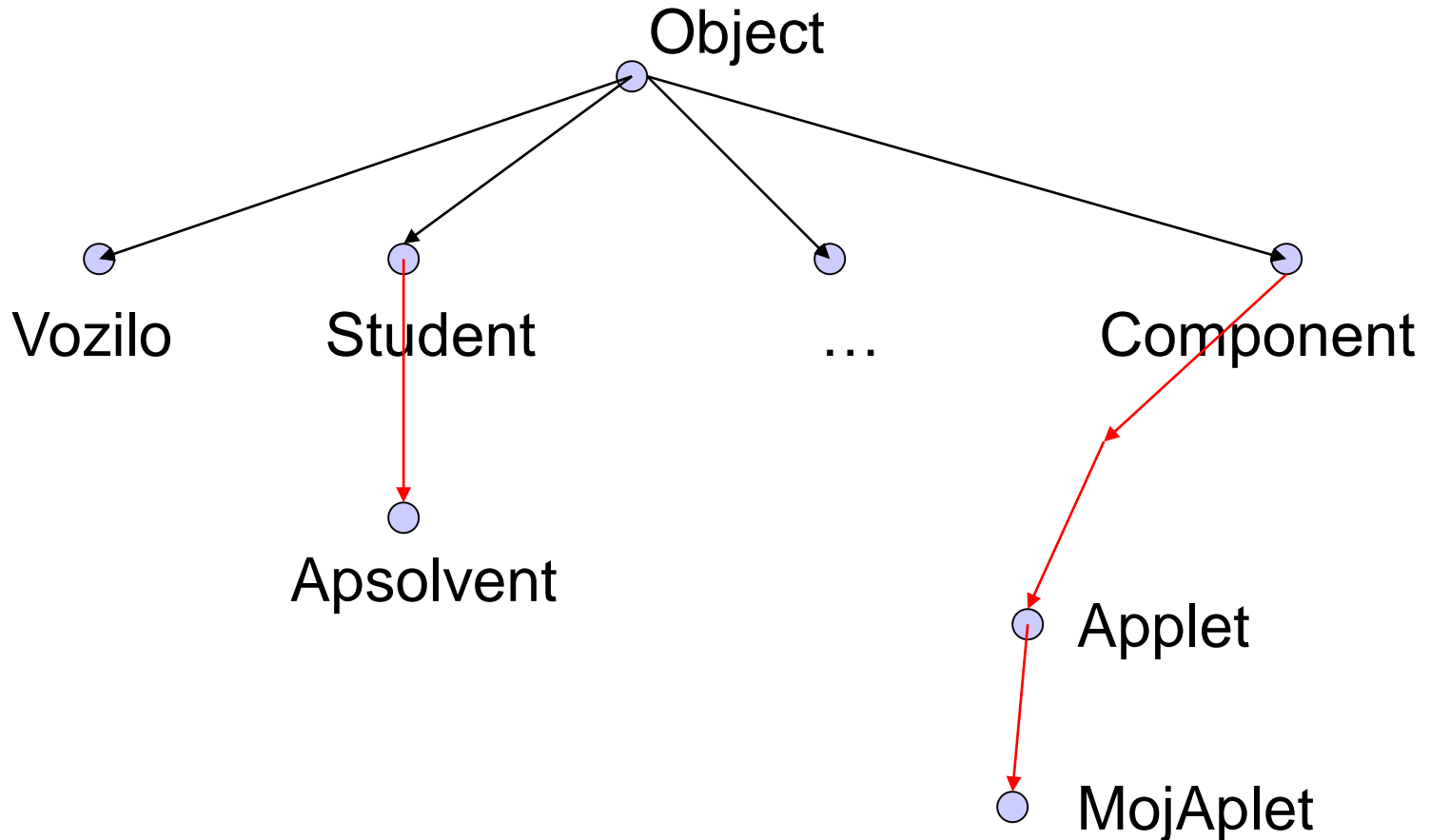
Јава као објектно-оријентисан језик

- Јава је објектно оријентисан програмски језик.
- Јава и C++ деле многе од принципа на којима се заснивају.
- Највећи број њихових разлика се односи на стил и на структуру.
- Јава је дизајнирана тако да подржава само једноструко наслеђивање.
- У циљу одржавања ефикасности Јаве, нису баш сви елементи реализовани као објекти:
 - Логичке величине (истина и лаж),
 - бројеви и други прости типови нису реализовани као објекти.



Јава као објектно-оријентисан језик (2)

Јава је објектно-оријентисан језик са **хијерархијском** структуром класа.





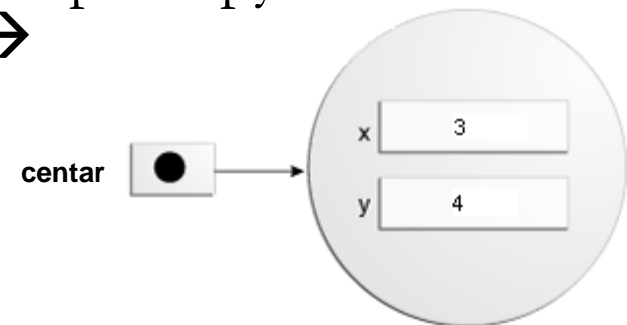
Јава као објектно-оријентисан језик (3)

- На врху класне хијерархије се налази класа `Object`.
 - Класа `Object` припада пакету `java.lang` из Јава API-ја.
- За сваку класу су битне две карактеристике:
 - **стање** се описује преко атрибута
 - **понашање** њених инстанци одређује се преко метода.
- Класе у Јави су организоване по пакетима. Основни пакети су:
`java.lang`, `java.util`, `java.io`, `java.net`, `java.awt`, `java.applet`.
- Пакет је скуп класа и интерфејса намењених једној врсти посла (које чине сродну целину).



Јава као објектно-оријентисан језик (4)

- **Важно** је запамтити да је Јава стриктно објектно оријентисана:
 - не допушта да се декларише било шта што није енкапсулирано (учаурено) у објекат, ово се односи и на атрибуте и на методе.
- На нивоу меморије променљива која представља објекат у Јави уствари садржи референцу на део меморијског простора који заузима дати објекат.
- На пример, објектна променљива **centar** која представља тачку у дводимензионалном координатном простору са координатама (3,4) изгледа овако →





Апстрактне класе

- **Апстрактна класа** је класа у којој постоји бар један метод који нема дефиницију (тело) већ само његову декларацију (заглавље).
- Такав метод се зове **апстрактан метод**.
- На пример, на претходном дијаграму класа за жива бића, метод **poмериSe** би се могао прогласити апстрактним, а разлог је:
 - Знамо да би животиња требала да може да се помера.
 - Међутим, не померају све животиње на исти начин, па не можемо на нивоу класе Животиња дефинисати конкретан начин померања.



Апстрактне класе (2)

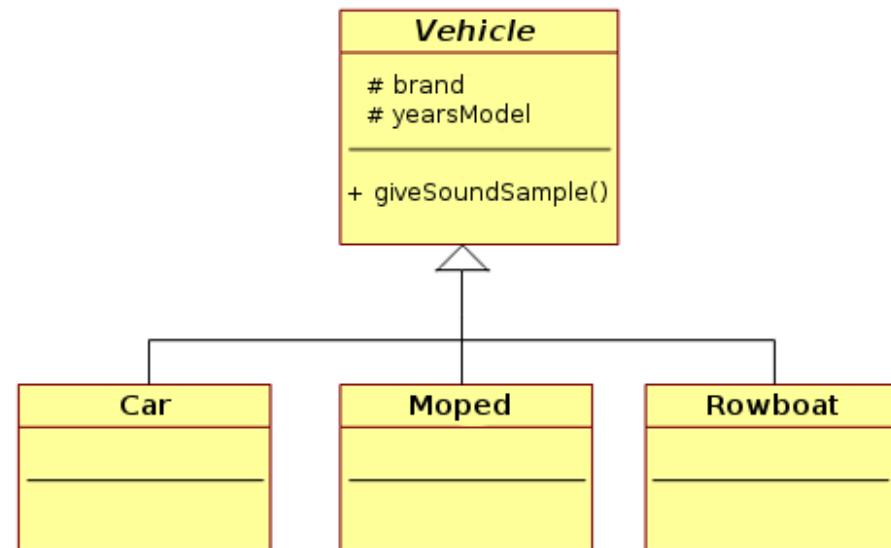
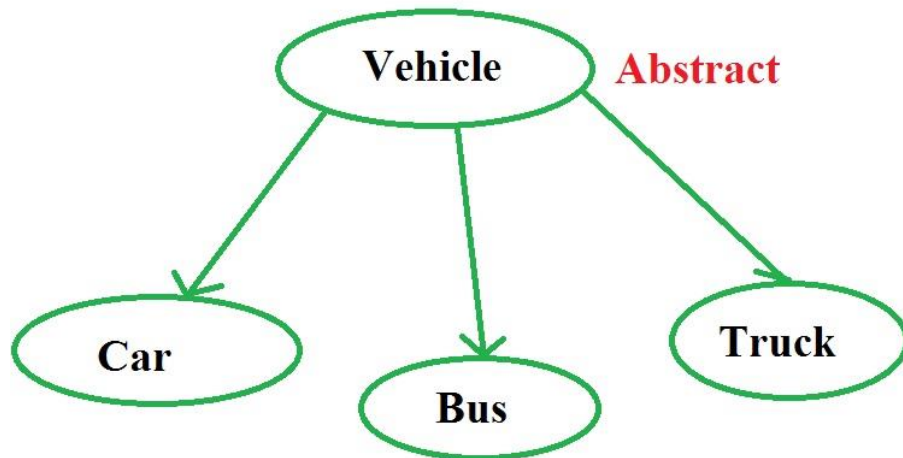
- Апстрактна класа **се не може инстанцирати**, тј. не може се направити објекат апстрактне класе!

- Поткласа апстрактне класе:
 1. Може реализовати све апстрактне методе и у том случају она постаје регуларна класа.
 2. Не мора реализовати апстрактне методе и у том случају и поткласа остаје апстрактна.



Апстрактне класе (3)

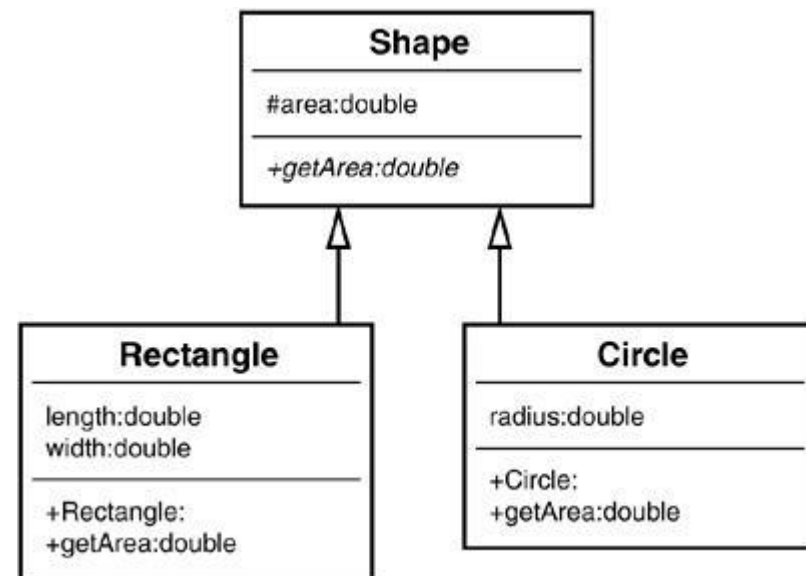
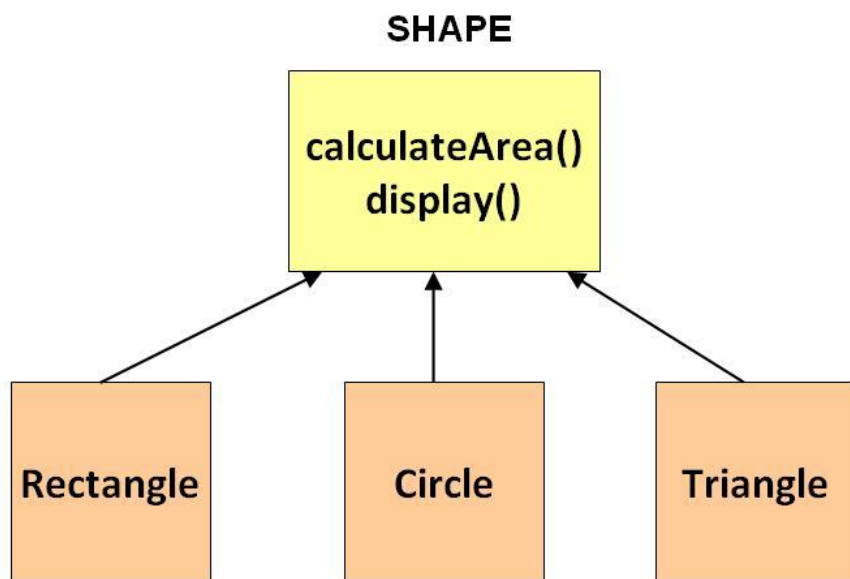
- Често, програмер користи апстрактне класе ако жели да обавеже да све поткласе дате класе морају да реализују неко понашање.
- Инстанца конкретне класе је истовремено и примерак апстрактне наткласе (исто као код наслеђивања регуларних класа)
- Пример наслеђивања са апстрактом класом:





Апстрактне класе (3)

Још један пример наслеђивања са апстрактном класом:





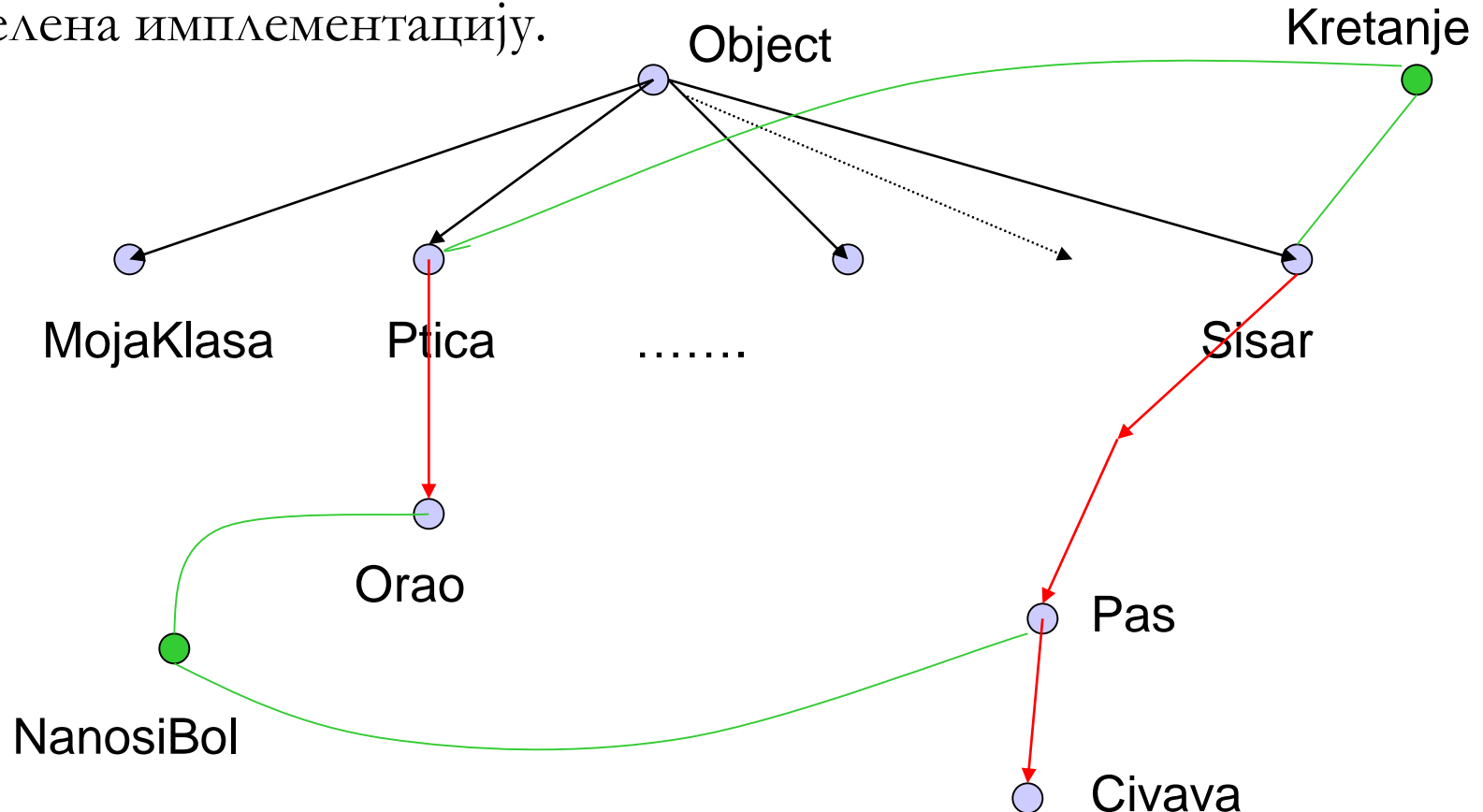
Интерфејси

- Јава допушта креирање тотално апстрактних класа, које су познате као **интерфејси**.
- У оквиру интерфејса се могу наћи само заглавља метода и дефиниције константи.
- Интерфејси описују понашање, а класа која их имплементира одређује како се то понашање реализује.
- Јава помоћу интерфејса делимично компензује непостојање вишеструког наслеђивања.
- Ако инстанца класе имплементира неколико интерфејса, може се посматрати као примерак сваког од тих интерфејса.



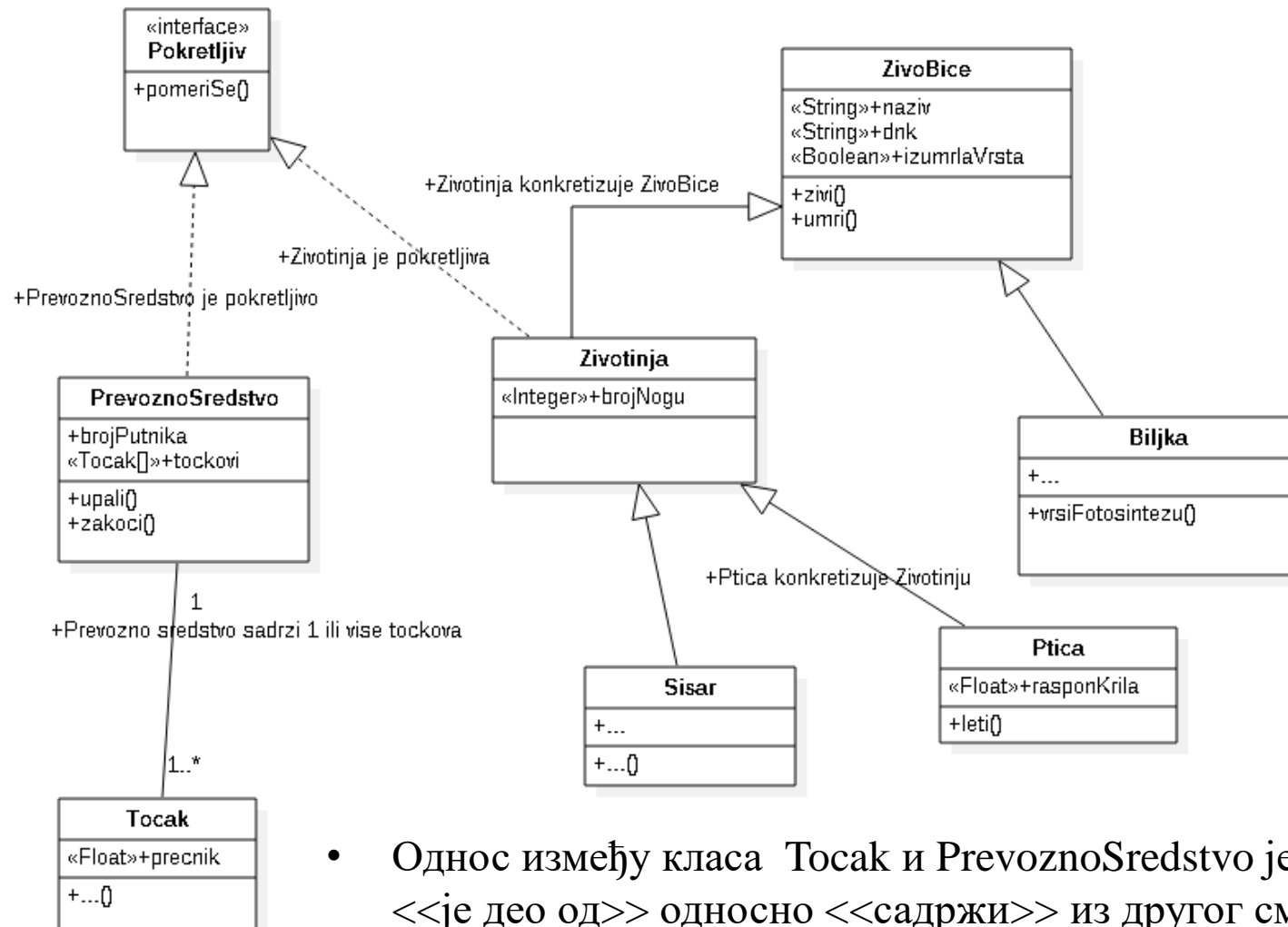
Интерфејси (2)

- Класа имплементира интерфејс ако саржи реализацију сваког од метода тог интерфејса.
- На следећем дијаграму црвена линија представља наслеђивање, а зелена имплементацију.





Интерфејси (2)



- Однос између класа `Tocak` и `PrevoznoSredstvo` је типа `<<је део од>>` односно `<<садржи>>` из другог смера.
- Овај тип односа се још зове и асоцијација.



Задачи за вежбање

1. Описати једну хијерархијску структуру са класама и поткласама.
2. Описати једну структуру са вишеструким наслеђивањем.
3. Из решења за тачку 2. вишеструко наслеђивање заменити употребом интерфејса.



Дијаграм на основу сценарија

- На основу следећег сценарија направити УМЛ дијаграм класа:

Авио-превозник је фирма која може да поседује неколико аеродрома широм света као и одређени број авиона, а најмање 1. Авио-превозник има своје име, шифру и државу у којој је регистрован.

Аеродром се описује трословном ознаком, државом у којој се налази и максималним бројем дневних летова. Постоје домаћи и међународни аеродроми, а разлика је у контроли пасоша и начину сервисирања авиона.

Авио-линија се описује својом почетном и завршном локацију односно аеродромом полетања и слетања. Поред тога, она има и своју ознаку.

За једну авио-линију постоје летови који се дефинишу ознаком излазне капије и очекиваним временом полетања и слетања.



УМЛ алати

- За прављење УМЛ дијаграма може се користити неки од следећих алата:
 - StarUML <http://staruml.io/download>
 - ArgoUML <http://argouml.tigris.org/>
 - Microsoft Visio – може се добити пуна верзија уз msdnaa matf лиценцу: <http://alas.matf.bg.ac.rs/msdnaa.html>
 - И још многи други који се могу наћи путем Интернета...



Захвалница

Велики део материјала који је укључен у ову презентацију је преузет из презентације коју је раније (у време када је он држао курс Објектно оријентисано програмирање) направио проф. др Душан Тошић.

Хвала проф. Тошићу што се сагласио са укључивањем тог материјала у садашњу презентацију, као и на помоћи коју ми је пружио током конципирања и реализације курса.