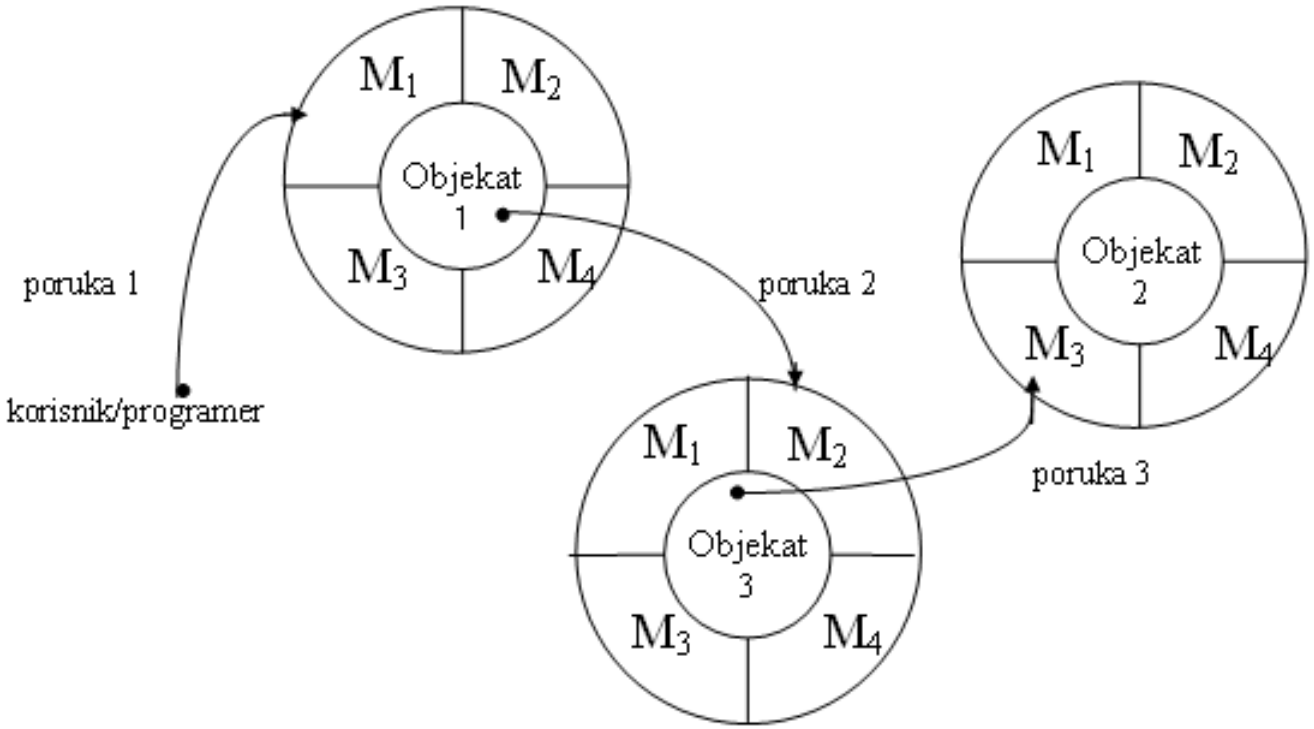
Увод у објектно програмирање:

Објектно оријентисано програмирање

Професор: **др Светлана Штрбац-Савић**

Маил / Кабинет: **svetlanas@viser.edu.rs / 501**

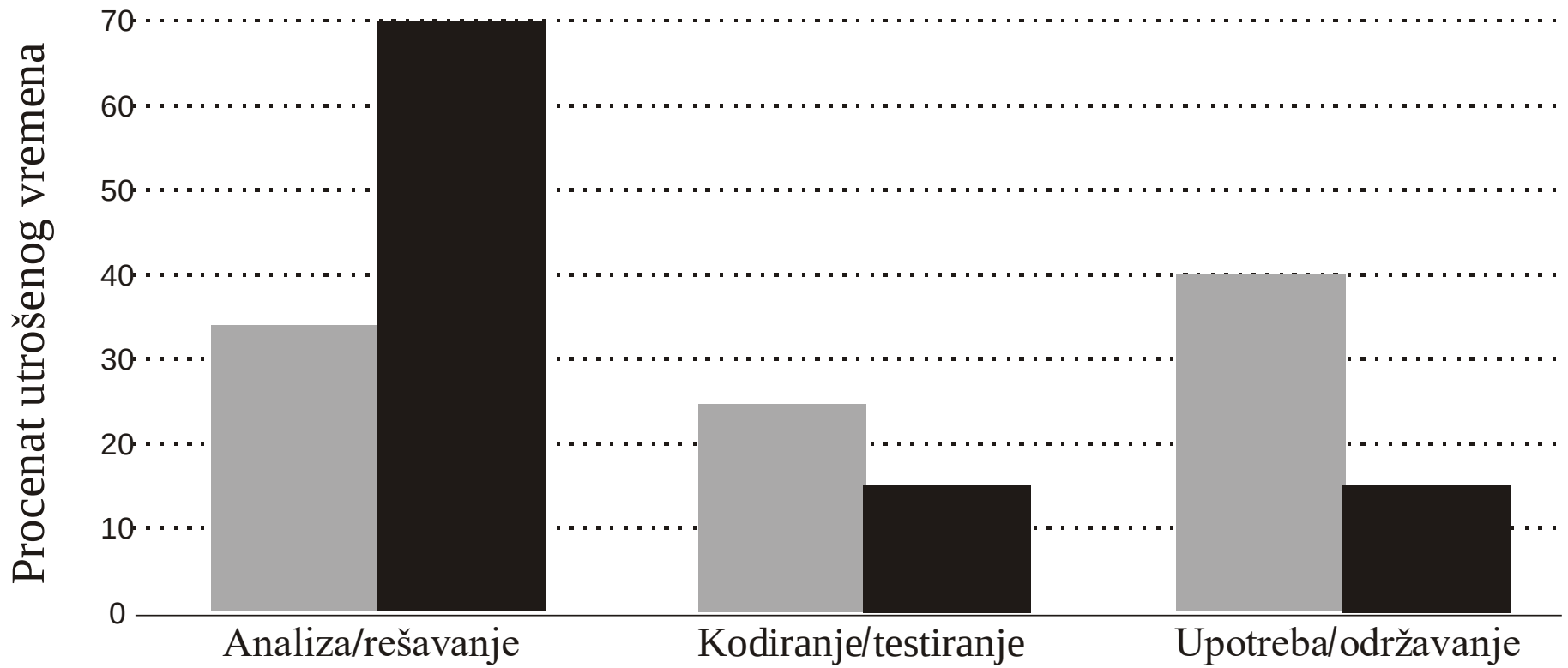
- Појам објекта - омогућити једноставан тимски рад великом броју програмера на великим и комплексним пројектима.
- За разлику од података који су непроменљиви (или врло мало променљиви), објекти су временски променљиви и имају стања која одговарају реалном свету.
- За разлику од процедура које описују како се изводи одређени поступак обраде, порука садржи спецификацију шта пошиљалац жели да се уради, а прималац одређује шта ће се тачно догодити, односно он обави посао или пренесе поруку другим објектима.
- Објекат има своје унутрашње стање чија је реализација недоступна другим објектима и операције (методе) које се над њим споља могу извршавати,



(a) Objekti šalju poruke jedan drugome

Metod 1	Metod 2	Metod 3	Metod 4	*Korisnik klase zahteva (odabira) metod pomoć poruke*
Interna implementacija metoda				*klasa određuje stvarni sadržaj svojih metoda*

b) Klase


Procenat utrošenog vremena
70
60
50
40
30
20
10
0
Analiza/rešavanje
Kodiranje/testiranje
Upotreba/održavanje

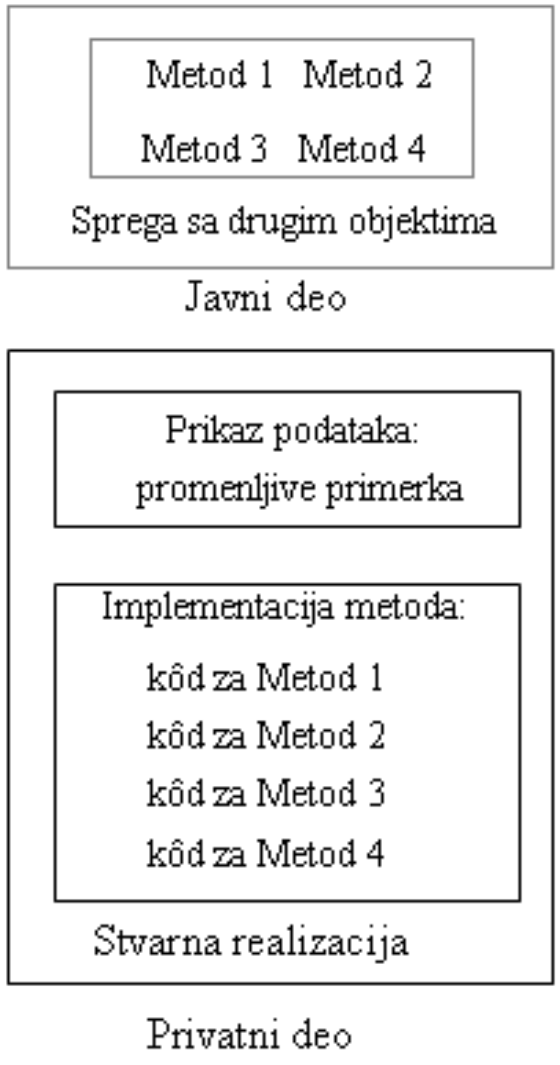
- Особине објектног програмирања су:
  - модуларност,
  - апстракција података,
  - класе,
  - наслеђивање,
  - полиморфизам,
  - енкапсулација и
  - динамичко повезивање и додела меморије.

- ***Модулира се и моделира проблем*** (стварни свет), а не решење проблема. Програм се формира око података који одговарају реалним објектима: Радник, Машина, Плата, Студент, Оцена, итд.;
- ***Модули су врло аутономни***, оперишу својим подацима по својим процедурама. На овај начин постиже се да измене у једном модулу не утичу на друге модуле;
- Омогућавају генерисање врло робустних програма у које се лако уграђују механизми за заштиту и изоловање грешака;
- Модули се могу ***више пута користити*** (у разним апликацијама).

Апстракција података

- за одређену примену, за сваки објекат дефинишу се његови ***атрибути*** (***attributes***), односно ***својства*** (***properties***) која се могу мењати само преко дозвољеног скупа операција - ***метода*** (***methods***), односно ***понашања*** (***behaviors***)[1].
- Апстрактни тип података садржи не само дозвољени скуп вредности него и спецификацију операција које су легалне за нови тип података.

[1] У разним књигама користе се различити називи за својства и понашање објеката, поред поменутих користе се још и појмови као: ресурси (resources) или приватне променљиве чланови (private member variables) За својства, и услуге (services) или операције (operations) и функције чланови (member functions).


Metod 1 Metod 2
Metod 3 Metod 4
Sprega sa drugim objektima
Javni deo
Prikaz podataka:
promenljive primerka
Implementacija metoda:
kôd za Metod 1
kôd za Metod 2
kôd za Metod 3
kôd za Metod 4
Stvarna realizacija
Privatni deo

- Класе представљају апстрактни тип података.
- Објекат је конкретни примерак класе.
- Једна класа може имати произвољно много објеката. Сви објекти имају исти скуп метода, али им се својства разликују.
- На пример класа Аутомобил може имати својства име, број_пређених_км, итд., а објекти класе Мерцедес и Пунто ће имати различите вредности својтава иако су објекти исте класе

- Објекти су повезани преко релација, основна три типа релација су:
  1. Потклассе
  2. Контејнери и сарадници
  3. Колаборатори

- Потклаcа је ***релација типа “је”***, односно објекат у једној класи је подтип неке друге класе.
- Ове релације се откривају у току пројектовања апликације или поступцима специјализације и детаљисања или поступцима генерализације и агрегације.

Генерализација и потклассе

- Генерализација је апстракција у којој се скуп различитих објеката третира као генерички објекат (класа).
- Специјализација је инверзни поступак у коме се за неку класу наводе њена могућа појављивања.

- Агрегација је апстракција у којој се скуп класа и њихових веза третира као једна класа на вишем нивоу апстракције.
- Агрегација се у хијерархији класа приказује као мешовита класа, односно као тип објекат-веза.
- Објекат ПРОИЗВОД и КУПАЦ агрегирају се у објекат ПОШИЉКА. Исто тако, агрегирани објекат ПОШИЉКА везује се са објектом КАМИОН и заједно са објектом РАДНИК формира агрегирани објекат РУКУЈЕ.

- ***Контејнер*** је релација типа “има” или “садржи”, односно један објекат може у себи да има или садржи друге објекте, или може чак бити састављен од других објеката.
- На пример у Visual Basic-у образац (form). Сваки образац садржи разне контроле (лабеле, текстуална поља, командну дугмад, оквире, итд.).

- Колаборатори, сарадници су два објекта који нису директно повезани већ један објекат користи други објекат ради постизања неког циља. Објекти су у релацији типа “користи”.
- Људи користе своје аутомобиле, у листи за плате се користи календар и штампач.
- Објекти сарадници морају бити дефинисани у истој апликацији.

- ***Релација наслеђивања***: изведена класа Б (подкласа) је једна специјална врста основне класе А (надкласа); Б има све што и А, с тим што може имати допунска својства и методе.
- Наслеђивање смањује време потребно за развој програма, јер програмер стално и намерно позајмљује методе и особине објеката који већ постоје у систему.

Пројектовање објектног програма:

- Уочавање, избор, односно идентификовање објеката;
- Дефинисање метода, односно понашања тих објеката;
- Избор података који су од значаја, односно дефинисање својстава објеката.

Избор објеката

- За сваку апликацију постоји одређени скуп објеката који су од интереса. Одређивање тог скупа објеката није јасно одређен задатак нити је једнозначан.
- Сама апликација нема јединствен опис, већ сваки појединац, било да се ради о корисницима или пројектантима, има сопствену апстракцију и саме апликације и објеката у њој.
- Да би апликација била одређена довољно је наћи бар један скуп објеката који је јасно и потпуно дефинишу.
- Да би нешто представљало објекат оно мора имати своју улогу у апликацији, односно мора имати ствари које може да уради (методи, понашања) и ствари које зна (својства).

- У захтеву да се у информационом систему Школе мора водити евиденција о релацији типа “предмети које је студент изабрао, пријавио и положио” уочавамо две именице: “студент” и “предмет” и оне представљају објекте - кандидате за апликацију.
- У овом случају предмет ће бити објекат у апликацији за евидентирање положених испита, али ће бити само својство објекту особа у апликацији за евиденцију дипломираних студената и за издавање потврде о положеним предметима

Дефинисање понашања - избор метода

- Методе је могуће пронаћи анализом глагола који дефинишу захтеве или понашање објеката у апликацији.
- Студент “бира“ предмете које ће “слушати“ и “полагати“, “пријављује испите“ и “полаже“ их.
- Наравно објекти у информационом систему могу радити више ствари него њихови оригинали из стварног живота, односно могу бити интелигентни

Дефинисање својстава

- Својства одређују шта ће објекат знати, а чине их подаци који су придружени објекту.
- При одређивању података треба узети у обзир не само основне податке, који су потребни за опис тог објекта (на пример име, број индекса, итд.), већ све податке који су потребни у апликацији, односно треба формирати списак свих својстава.

Пример полиморфизма:

- Ако је правоугаоник потклаcа класе фигура, тада ће се површина објекта који је инстанца класе правоугаоник рачунати по формули дефинисаној у методу површина класе правоугаоник, без обзира на то да ли класа фигура има свој метод који рачуна површину фигуре.

- Енкапсулација ("скривање информација") је принцип према којем су детаљи из програмске структуре "невидљиви".
- Комуникација између различитих структура може да се врши само на тачно дефинисане начине, коришћењем унапред одређених процедура.
- На овај начин се смањује међузависност модула

- Додела меморије која се обавља приликом извршавања програма назива се ***динамичком доделом меморије***.
- Могуће је да у време писања програма програмер не зна тачно који ће тип објекта бити формиран.
- То је омогућено декларацијом показивача или референце на надкласу.
- Током извршавања програма корисник бира тип објекта који ће бити формиран.

- Крајњи корисник при употреби програма комуницира директно са рачунаром. Зато начин рада са програмом постаје основно питање, а начин писања програма помоћу прототипова постаје природан.
- Језик треба да омогући стварање прототипова корисничког интерфејса:
  - Прво се направи прототип, тј. програм који садржи начин рада са програмом. Прототип корисничког интерфејса требало би да садржи: мени, палету са алаткама и примарне екране апликација, а од програмског кôда довољно је да се обезбеди кретање између менија, палете са алаткама и екрана.
  - Затим се тај начин рада тестира, на пример, код наручиоца посла, тј. корисника. Тако се постиже да сви који користе апликацију прођу кроз њу довољно рано да би се мане могле отклонити