

```
1
2
3 Programiranje 'JavaScript' {
4
5     [Objektno orijentirano
6     za početnike]
7
8     < Saša Mladenović >
9     < Divna Krpan >
10    < Ante Šurlin >
11    < Dino Nejašmić >
12
13 }
14
```

Očekivani ishodi učenja

- * Razviti jednostavan objektno orijentirani (OO) projekt koristeći OO paradigmu i pripadajuće pomoćne alate.
- * Implementirati OO model u OO jeziku visoke razine korištenjem objekata, klasa, nasljeđivanja, nizova, uvjetovanih izraza i iteracije.
- * Upoznati sa načinom dokumentiranja, rasporedom, testiranjem i pronalaženjem grešaka kod OO programiranja.

Očekivani ishodi učenja

- * Objasniti prednosti korištenja OO razvojnog pristupa i u kojim slučajevima je to prikladna metodologija.
- * Primijeniti ispravnu programersku paradigmu ovisno o zadanom problemu, te biti upoznat sa utjecajem odabrane paradigme na razvoj i održavanje aplikacija.
- * Dizajnirati i implementirati prikladno grafičko korisničko sučelje za pristupni (front-end) dio objektno orijentirane aplikacije.

Microsoft Teams



Teams



Join or create team



Join a team with a code

Enter code **h278v57**

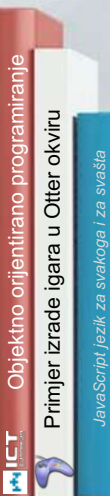
Got a code to join a team? Enter it above.

Za komunikaciju ćemo koristiti
MS Teams

Sve ostale informacije vezane uz mjesto
na kojem ćete dobivati materijale dobit
ćete na Teamsu



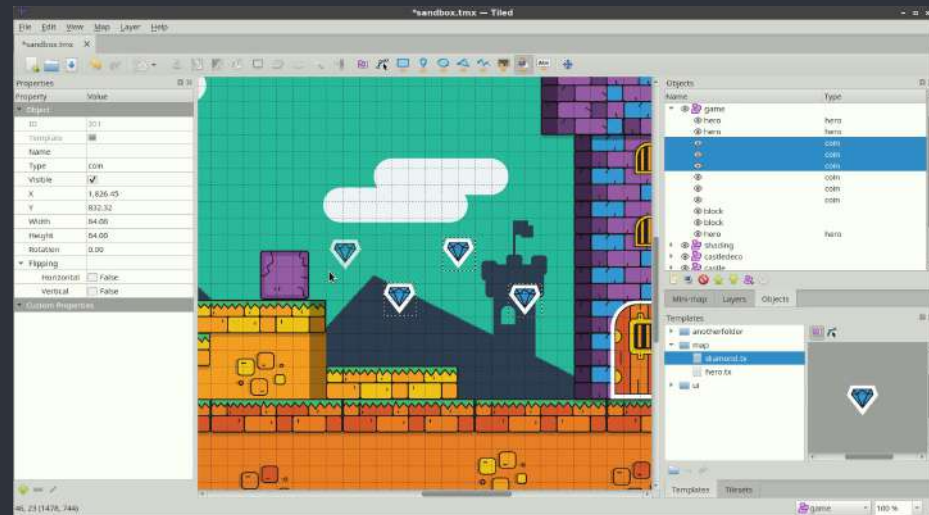
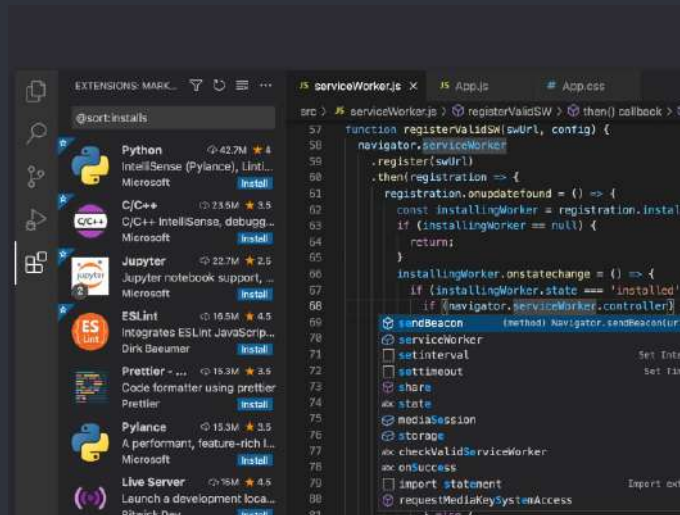
KOJE ĆEMO TEME OBRADITI I KAKO ĆEMO UČITI?



Pitanja na koja ćemo pokušati odgovoriti

- * Što je objektno orijentirano programiranje?
- * Kako koristiti tehnike objektno orijentirano programiranje pri rješavanju problema?
- * Kakva je primjena objektno orijentiranog programiranja u stvarnom životu?
- * Što je **objekt**?
- * Što je **klasa**?
- * Što je **metoda**?

Što je s vježbama?




Visual Studio Code



Kako ćemo do ocjene?

5⁺



Visual Studio Code

Održavanje vježbi



Divna Krpan

dkrpan@pmfst.hr

Grupa 1: ponedjeljak 10-12 h
Grupa 2: petak 12-14 h



Ante Šurlin

asurlin1@pmfst.hr

Grupa 1: ponedjeljak 16-18 h
Grupa 2: ponedjeljak 18-20 h



Dino Nejašmić

dnejasmic@pmfst.hr

Grupa 2: četvrtak 16 – 18 h
Grupa 2: četvrtak 18 – 20 h





Visual Studio Code

Osnovne informacije

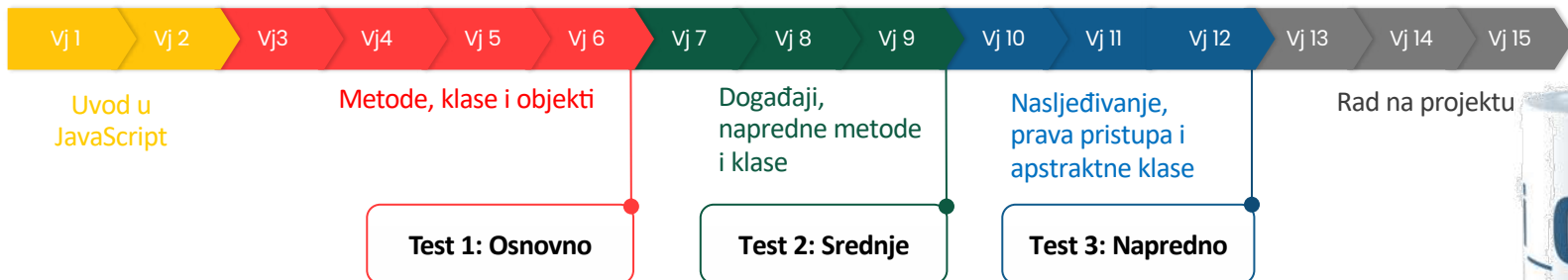


Materijali i obavijesti:

- e-kolnij na <https://edit.dalmacija.hr/>
- MS Teams



Okvirni plan vježbi i testova kontinuiranog praćenja



Uvjet za potpis i pristupanje polaganju kolegija

Vježbe

Kontinuirano vrednovanje: nema klasičnih kolokvija/ispita

Postotak
(0-100%)

Test 1:

p_1

Test 2:

p_2

Test 3:

p_3

Vježbe položene ako je:

$$p_1, p_2, p_3 \geq 25\% \quad \&$$

$$\frac{p_1 + p_2 + p_3}{3} \geq 50\%$$

Projekt

ROK ZA PREDAJU

do prvog ispitnog roka



Bonus test

- Bit će organiziran **bonus test** na kojem će se moći popravljati rezultati testova kontinuiranog praćenja:
 - Na bonus testu može se popravljati jedan, dva ili sva tri testa.
 - Pristupanjem bonus testu “**briše**” se **prethodno ostvareni rezultat** (čak i kada je novi rezultat lošiji).



Uvjeti za 'potpis' {

01 Praktična znanja

< Položeni testovi kontinuiranog praćenja >

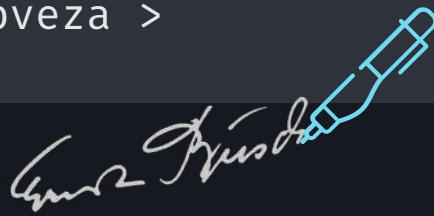
02 Koncepti u praksi

< Projekt "retro igre" predan na vrijeme >

03 Ostalo

< Samostalno i pravovremeno izvršavanje svih obveza >

}



Vrednovanje 'Ispitivanje' {

Kontinuirano vrednovanje:

- * Test (teorija + vježbe)
- * Bonus test (za popravak)

Projekt:

- * Tema (temu birate u dogovoru s nositeljem)
- * Obrana (usmeni ispit i pojašnjenje koda)

}

```
1 < “Uspješno kontinuirano vrednovanje  
2 dokazuje kako ste ovladali sintaksom i  
3 osnovnim konceptima OOP-a.  
4 Projekt ukida potrebu za složenim  
5 pismenim ispitima.” >
```

```
8 – Nema stresa ‘Uspješno položen kolegij’
```

50%

```
11 – Nimate 50% na testovima ili  
12 niste predali projekt  
13 ‘Niste ostvarili pravo na potpis’  
14
```



Lista projekata

{ Retro je in;

< [https://en.wikipedia.org/wiki/
List_of_platform_game_series](https://en.wikipedia.org/wiki/List_of_platform_game_series) >

}



Donkey Kong – 1981–1984

Nintendo R&D1



Donkey Kong – 1981–1984

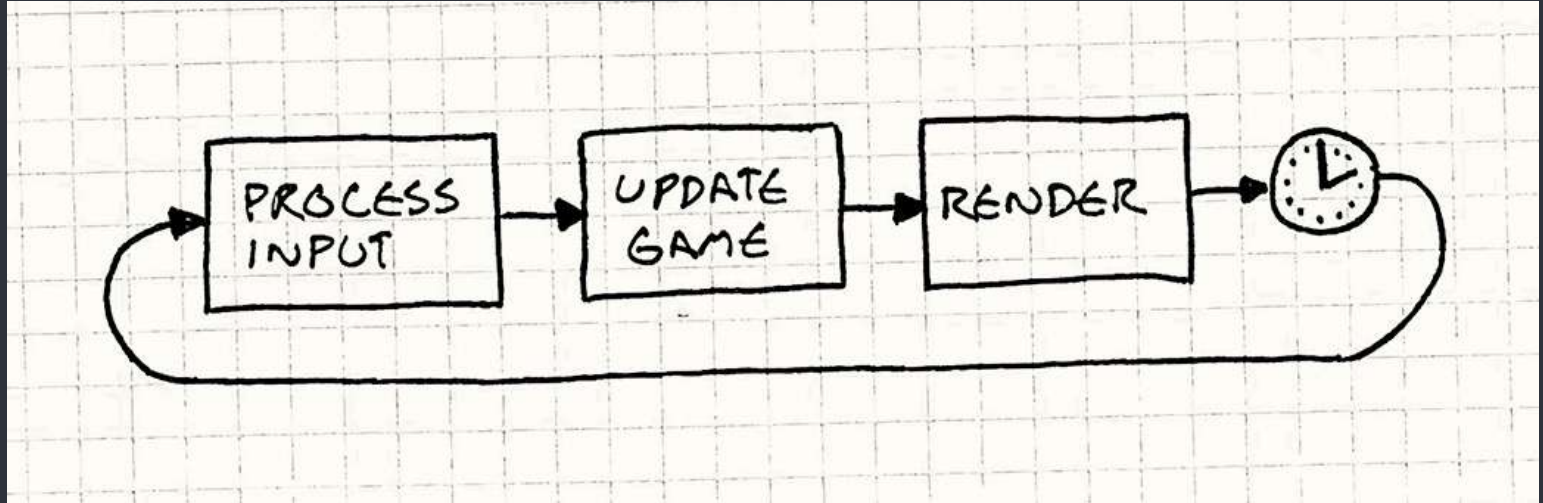
Nintendo R&D1



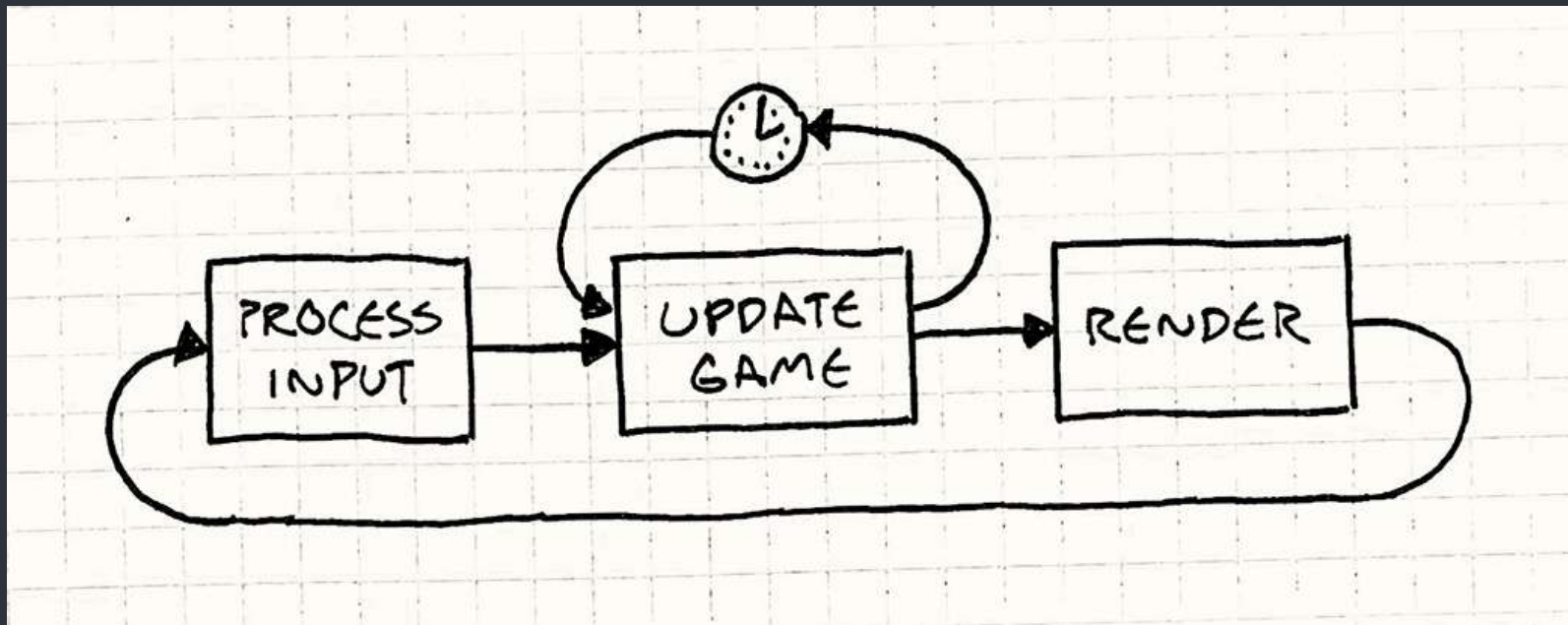
< <https://www.sprites-resource.com/fullview/117900/> >



Petlja igre



Petlja igre, malo drukčije



Sadržaj 'cjeline' {

01 Teorijski uvod

< 0 paradigama, složenosti
i programskim jezicima >

02 Osnovni koncepti

< Objekt, klasa, metoda,
modifikatori pristupa >

03 Praktična primjena

< Jednostavna klasa i
konstruktor C# i JS >

}

01 {

[Teorijski uvod]

< U čemu je problem? >

}

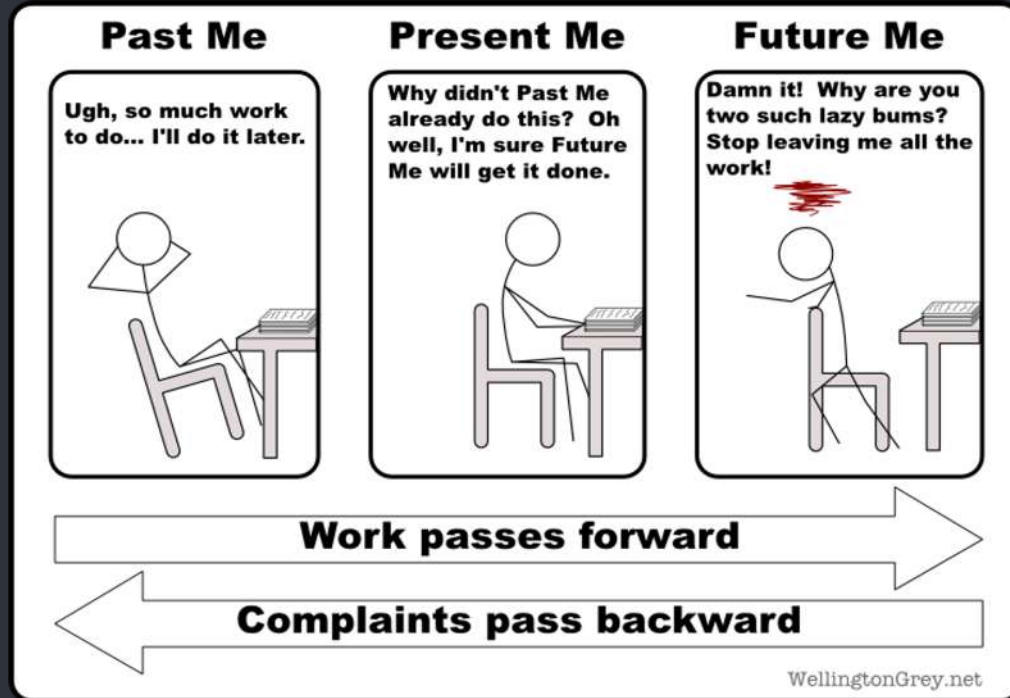
Što je u programiranju 'teško' ?



Definirati zahtjeve korisnika !
prije izrade sustava . . .



Što je u Programiranju 2 'teško' ?

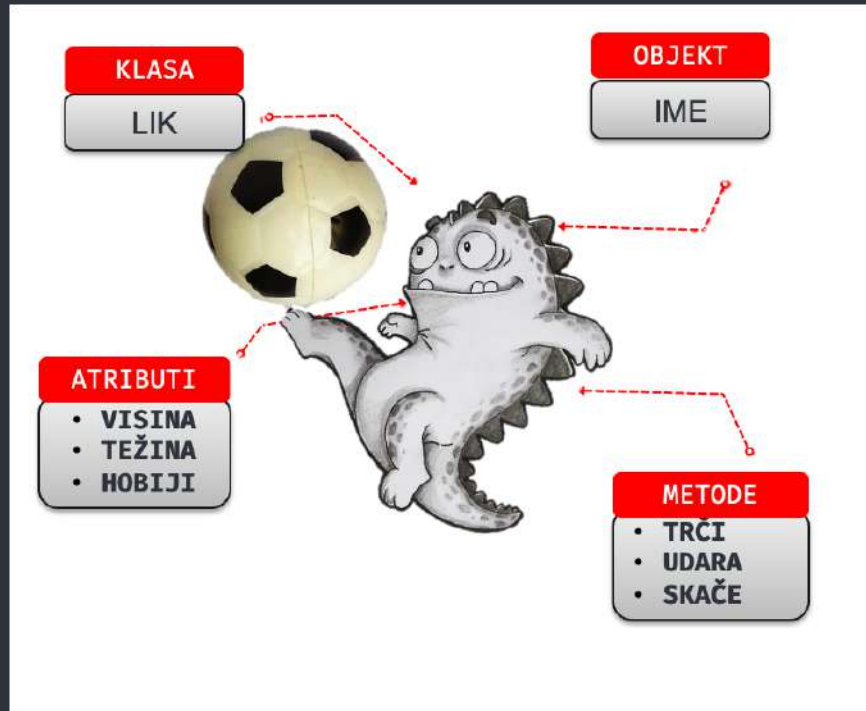




MOJ PRVI PROGRAM !

```
9  public class PrvaKlasa
10 {
11     public static void Main()
12     {
13         System.Console.WriteLine("Ovo je moj prvi C# program");
14         System.Console.ReadKey();
15     }
16 }
```

Što susrećemo u OOP-u?



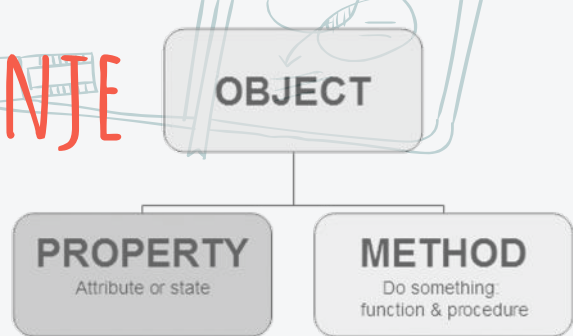


ODAKLE SVI TI KONCEPTI ?

Objektno – orijentirano programiranje koristi **objekte** i **njihove interakcije** za izgradnju računalnih programa.

Na taj se način **oblikuje** razumljiv **računalni model** nekog područja ili problema.

OBJEKTNO ORIJENTIRANO PROGRAMIRANJE



Predstavlja proširenje proceduralnog programiranja

Objekti

Slični konkretnim objektima u stvarnom svijetu, a **sadrže vlastite varijable** (polja ili članove) i **metode**

- **ATRIBUTI** objekta predstavljaju **njegova svojstva**
- **STANJE** objekta je **skupna vrijednost svih** njegovih **atributa** u bilo kojem trenutku
- **METODA** definira ponašanje objekta (što objekt “čini”).

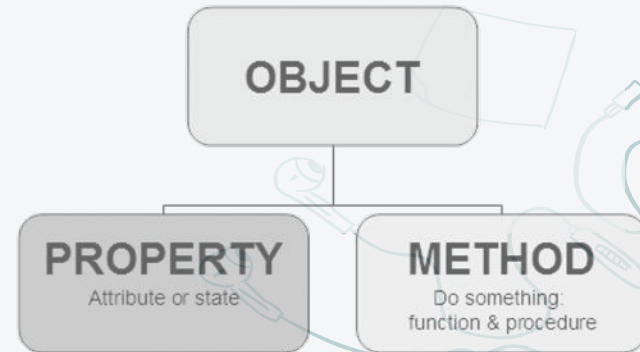
OBJEKTNO ORIJENTIRANO PROGRAMIRANJE

Gdje se uopće koristilo ?

Originalno korišteno za dvije vrste aplikacija

Računalne simulacije

Korisnička sučelja (Graphical user interfaces GUI)





Sad znam sve!



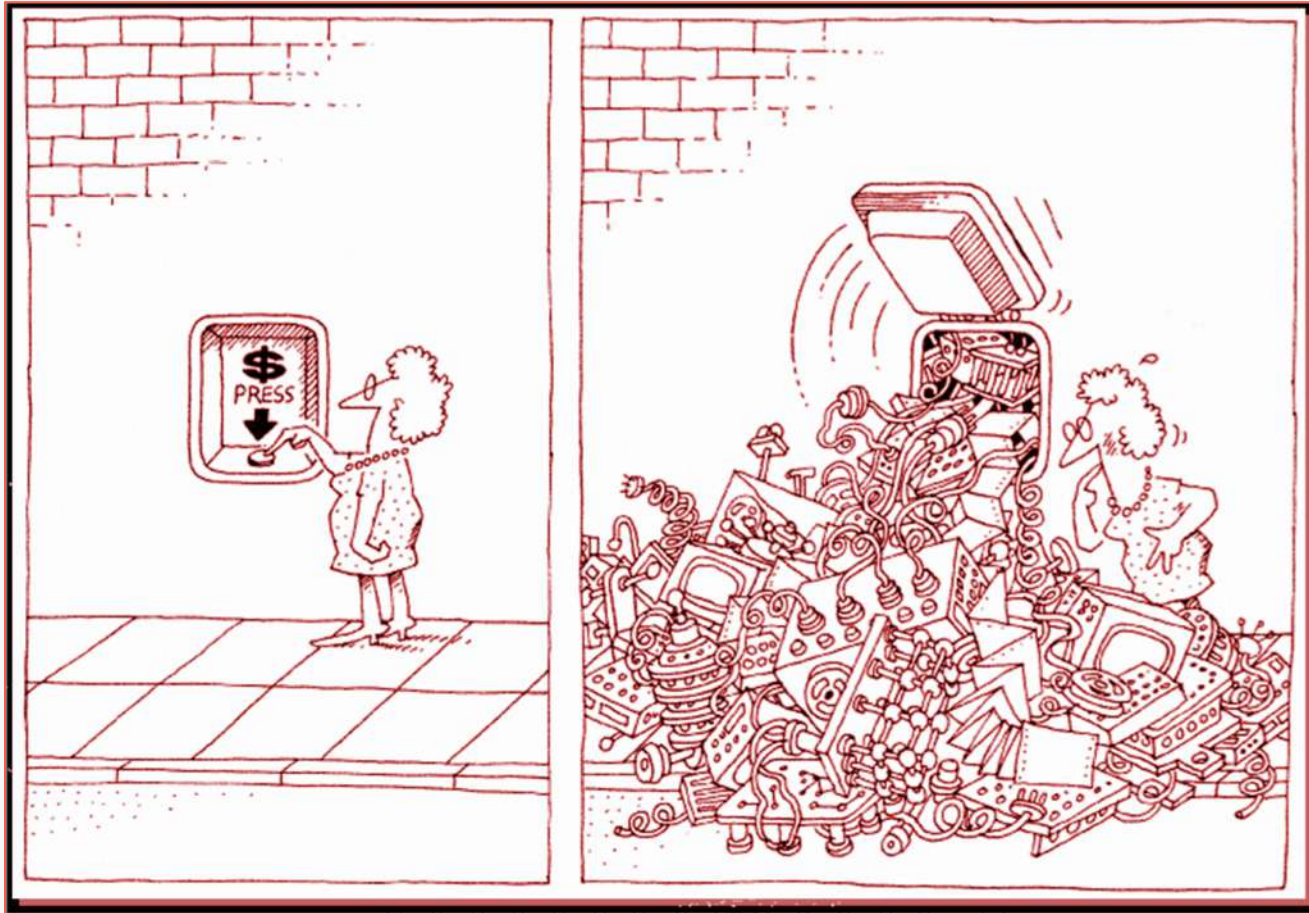
Što je u programiranju teško?!



```
1
2
3 Složenost 'Programske podrške' {
4
5     [Za korisnike i Programere]
```

```
6
7
8
9     < Što je složeno u programskoj podršci ? >
10
11
12 }
```


SLOŽENOST PROGRAMSKE PODRŠKE





SLOŽENOST PROGRAMSKE PODRŠKE

■ Razlozi složenosti programske podrške

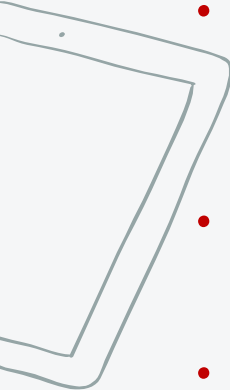
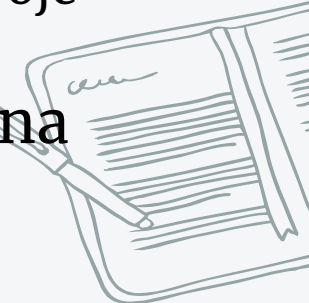


- složenost i često nejasna definicija problema
 - ✓ iterativni razvoj, interakcija s korisnikom
- teškoće u upravljanju razvojnim procesom
 - ✓ dekompozicija u module, koordinacija
- softver omogućuje ostvarenje svih elementarnih gradivnih elementa što se često koristi zbog nedostatka standarda
- problem opisa ponašanja diskretnih sustava
 - ✓ ne mogu se modelirati kontinuiranom funkcijom,
 - ✓ mogu imati ogroman broj diskretnih stanja
- nerazumni zahtjevi korisnika



SLOŽENOST PROGRAMSKE PODRŠKE



Pet atributa složenog sustava


- imaju hijerarhijsku formu
 - ✓ sastoje se od povezanih podsustava koji mogu imati svoje podsustave
 - definicija osnovnih komponenti nije jednoznačna
 - ✓ ovisi o subjektivnom gledištu promatrača
 - veze unutar komponenti su jače od veza među komponentama
 - ✓ sustavi su razloživi jer se mogu razdijeliti na komponente
 - ✓ sustavi su skoro razloživi jer komponente nisu nezavisne
- 
- 
- 
- 



SLOŽENOST PROGRAMSKE PODRŠKE



Složeni sustavi

- često se sastoje od samo nekoliko različitih podsustava, ali u mnogim kombinacijama i pojavnim oblicima
 - ✓ osnovne komponente s dna hijerarhije često su zajedničke
 - složeni sustavi evoluiraju iz jednostavnijih
- 

Razumijevanje složenih sustava olakšava prepoznavanje svojstava koja su zajednička s drugim sustavima

- ✓ zrakoplov, automobil
- 
- 
- 
- 



SLOŽENOST PROGRAMSKE PODRŠKE

- “industrial - strength software” [Booch94]
 - nijedan član razvojnog tima ne može biti upoznat sa svim detaljima projekta
 - vrlo dugi životni vijek programske podrške
 - mnogi korisnici zavise od korektnog funkcioniranja programske podrške
 - veliki broj ljudi je uključen u održavanje i poboljšanje programa
- “softverska kriza”
 - strojna osnovica je sve moćnija
 - programska podrška je sve obimnija i zahtjevnija
 - troškovi održavanja su u dramatičnom porastu
 - **ne postoji dovoljan broj ljudi osposobljenih za razvoj kvalitetne programske podrške**



SLOŽENOST PROGRAMSKE PODRŠKE

- Evolucija programskih jezika
(i tipičan slijed učenja programera)
 - Nestrukturirano programiranje
 - Proceduralno programiranje
 - Modularno programiranje
 - Objektno orijentirano programiranje

```
1
2
3  Učenje 'Programskih jezika' {
4
5      [Paradigme i Jezici]
6
7
8
9      < Što je pogrešno, a što ispravno ? >
10
11
12  }
13
14
```

NESTRUKTURIRANO PROGRAMIRANJE

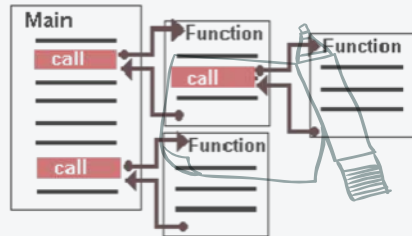
Učenje programiranja počinje pisanjem **malih i jednostavnih programa** koji se sastoje od slijeda naredbi i djeluju nad zajedničkim skupom podataka



Ponavljanje nekog posla znači i **kopiranje** naredbi

PROCEDURALNO PROGRAMIRANJE

Izdvajanjem naredbi u procedure, **program** postaje **slijed poziva** procedura



Glavni program

Podaci

Procedura 1

Lokalni podaci

Procedura 2

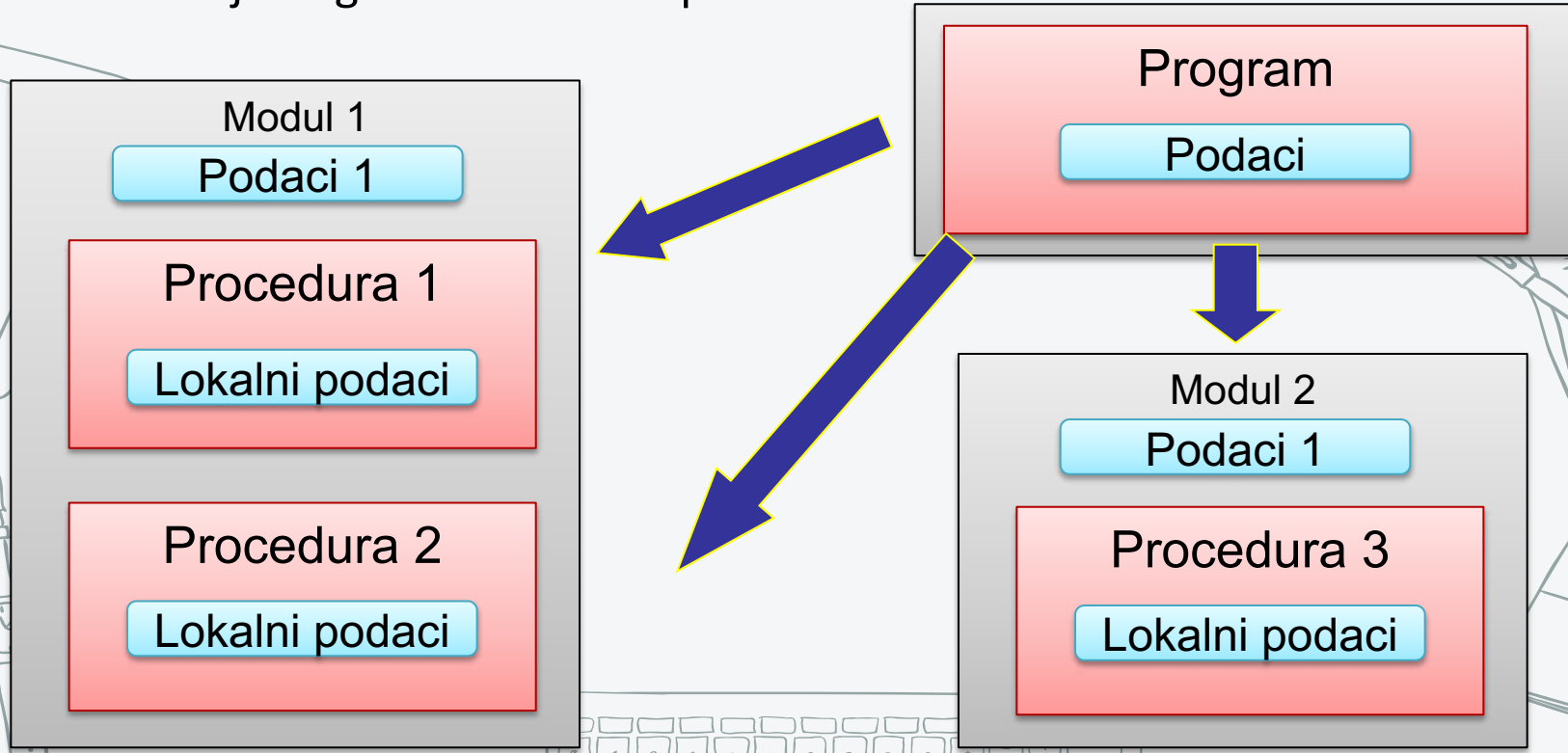
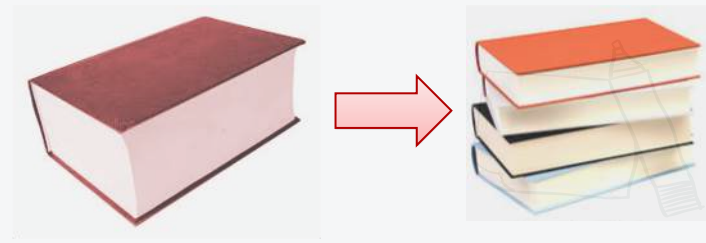
Lokalni podaci

Procedura 3

Lokalni podaci

MODULARNO PROGRAMIRANJE

Procedure srodne funkcionalnosti grupiraju se
u module koji mogu imati vlastite podatke.



CILJEVI

- Učiti o programiranju
- Učiti o proceduralnom i objektno orijentiranom programiranju
- Učiti o svojstvima objektno-orijentiranih programskih jezika
- Razmotriti program koji daje izlaz
- Naučiti kako odabrati identifikatore koje će se koristiti u programu



PROCEDURALNO I OBJEKTNO-ORIJENTIRANO PROGRAMIRANJE

Proceduralni program

- Kreira i imenuje memorijsku lokaciju računala koja čuva vrijednosti (**varijable**)
- Piše niz koraka ili operacija kao bi manipulirao tim vrijednostima

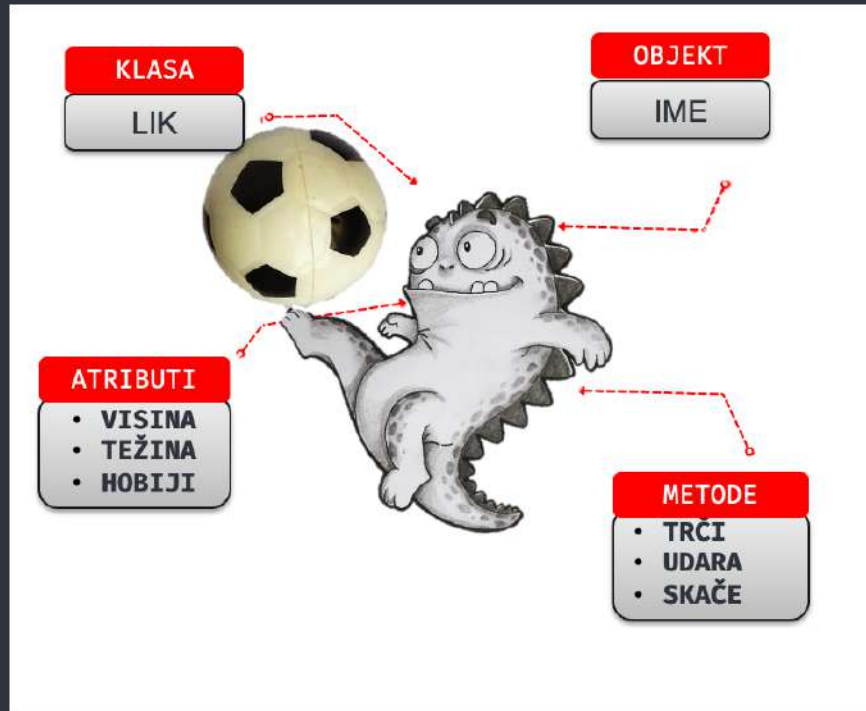
Identifikator

- Ime (jedna riječ) koje se koristi kao naziv varijable, metode, klase i slično

Procedure ili metode

- Logičke cjeline koje grupiraju pojedinačne operacije koje koristi računalni program
- **Koje su pozvane ili pokrenute** od drugih procedura ili metoda

Što susrećemo u OOP-u?





PROCEDURALNO I OBJEKTNO-ORIJENTIRANO PROGRAMIRANJE

■ Objektno-orijentirano programiranje

- **Proširenje proceduralnog** programiranja

■ Objekti

- **Slični konkretnim objektima** u stvarnom svijetu
- Sadrže **vlastite varijable** i **metode**
- **Atributi** objekta predstavljaju njegova **svojstva**
- **Stanje objekta** je skupna **vrijednost svih** njegovih **atributa** u bilo kojem trenutku
- **Ponašanje objekta** predstavlja što objekt “**čini**”

Za 'danas' je {
Dosta

|
}

Objektno orijentirano programiranje



Vidimo se idući
tjedan!