Projektovanje

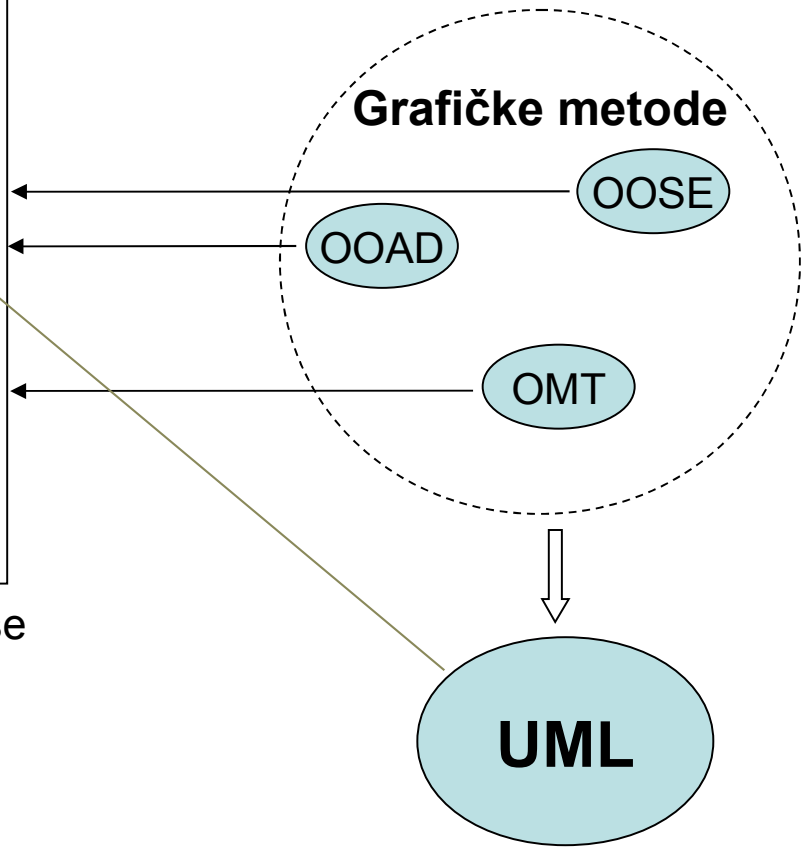
Nedovoljna apstraktost programskih jezika dovela je do pojave grafičkih metoda.

Projektovanje softvera

Problemi

- Prenos znanja i ideja
- Komunikacija
- Razumevanje

Rasprave o projektovanju softvera su lakše ukoliko postoji izvestan nivo apstrakcije.



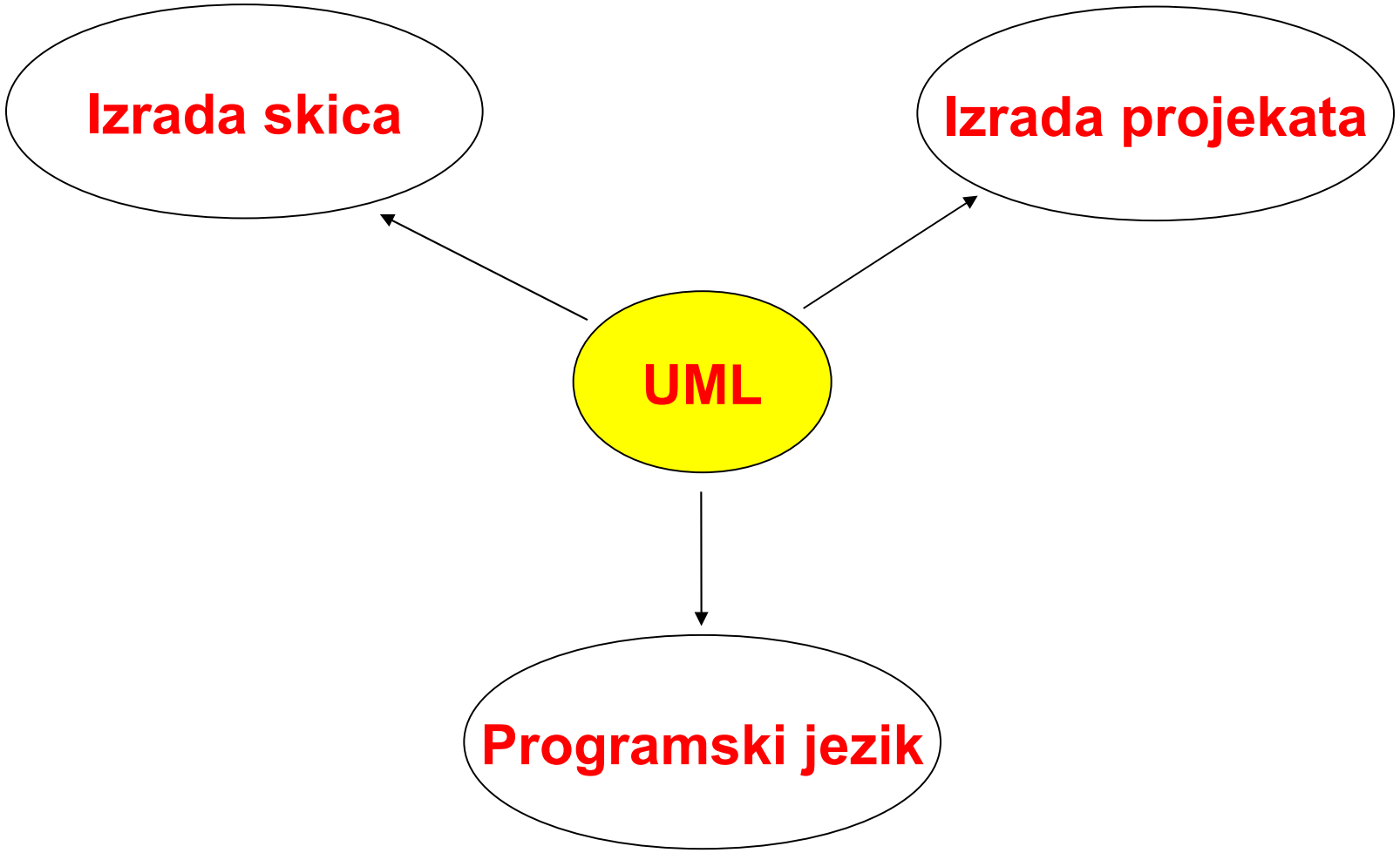
Neusaglašenost postojećih grafičkih metoda dovela je do pojave UML-a.

Korisnici UML – a

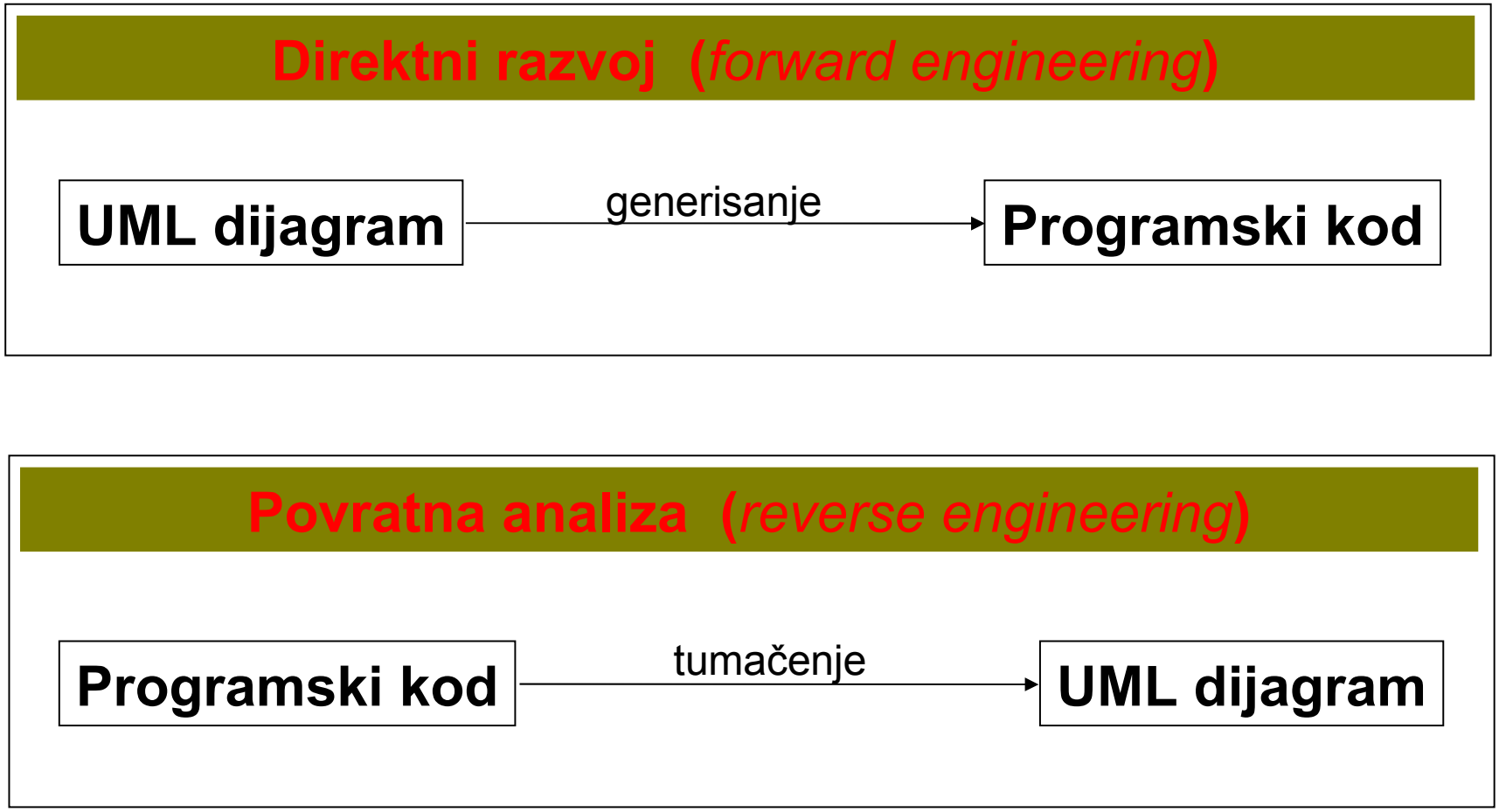
UML se koristi za modelovanje i projektovanje softverskih sistema, naročito sistema pravljenih primenom objektno-orijentisanih tehnologija

- Sistem-analitičari i krajnji korisnici:
specificiraju zahtevanu strukturu i ponašanje sistema
- Rukovodioci projekata:
vođenje i usmeravanje kadrova i upravljanje resursima
- Arhitekti: projektuju sistem koji zadovoljava zahteve
- Razvojni inženjeri: transformišu arhitekturu u izvršni kod
- Kontrolori kvaliteta:proveravaju strukturu i ponašanje sistema
- Evidentičari komponenata: kreiraju i katalogiziraju komponente

Načini korišćenja UML-a



Postupci razvoja



Izrada skica

Skice opisuju pojedine aspekte sistema korišćenjem UML-a kao pomoćnog sredstva.

Osobine skica:

- predstavljaju najčešći način upotrebe UML-a
- obično se generišu neformalno i dinamički, tako da se rade brzo i na tabli
- nepotpune su i uglavnom imaju obaveštajni karakter
- omogućavaju jednostavno ispitivanje više alternativnih rešenja
- mogu se koristiti u dokumentaciji

Skice u direktnom razvoju:

- **sadrže samo nekoliko značajnih problema koji će se javiti u kodu**
- saopštavaju ideje i alternative predstojećeg posla
- vizualizuju delove projekta pre programiranja

Skice u povratnoj analizi:

- **objašnjavaju kako radi neki deo sistema (samo klase o kojima je važno razgovarati)**

Izrada UML projekata

UML projekti opisuju sistem sveobuhvatno, kako bi se programiranje koje predstoji svelo uglavnom na mehaničku aktivnost.

Osobine UML projekata:

- ❑ za izradu projekata koriste se složeni, tzv. CASE alati za računarsko projektovanje softvera
- ❑ projekti se moraju dosledno sprovoditi

Direktni razvoj:

Projekti sadrže detaljan opis sistema, obično do nivoa interfejsa podsistema (dalja razrada realizacije se prepušta programerima), na osnovu koga se piše kod.

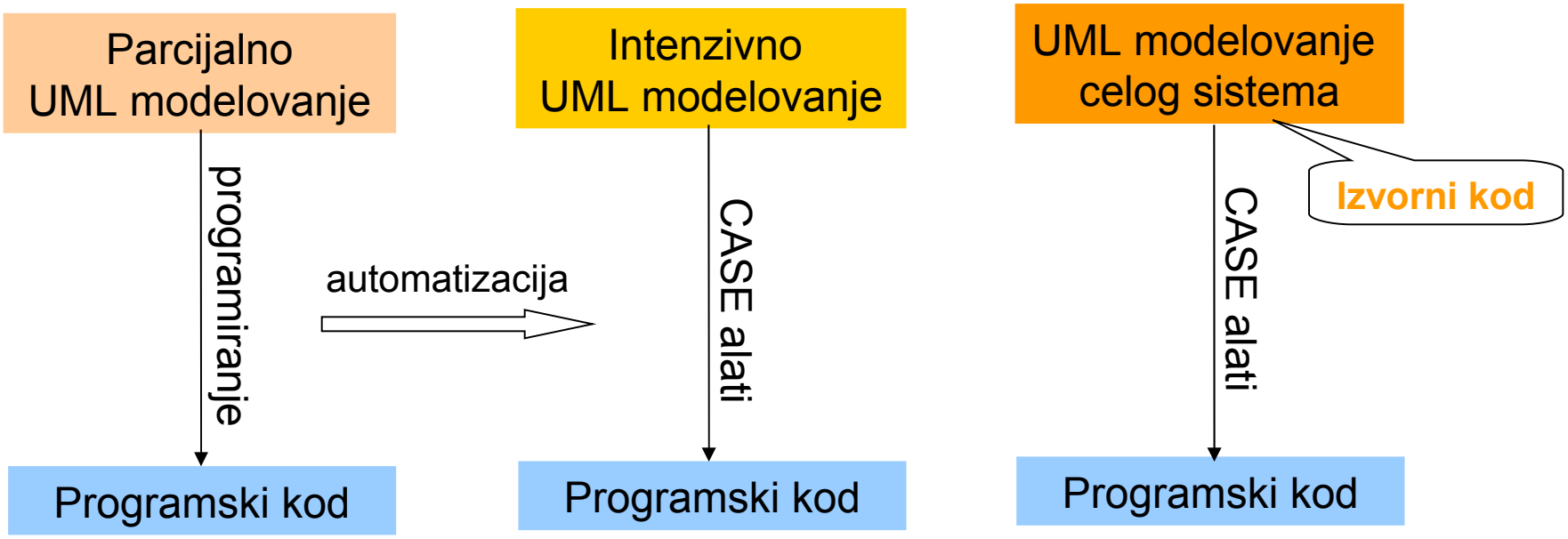
CASE alati omogućavaju crtanje dijagrama i čuvanje informacija u memoriji.

Povratna analiza:

Projekti sadrže detaljne informacije o kodu u obliku papirne ili elektronske dokumentacije. Mogu prikazati svaki detalj neke klase u grafičkom obliku koji programer razume.

CASE alati čitaju izvorni kod, tumačenja stavljaju u memoriju i generišu dijagrame.

UML kao programski jezik

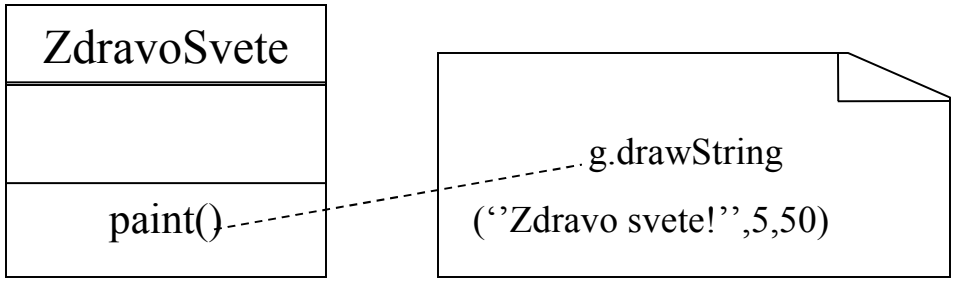


UML postaje programski jezik u slučaju kada se celokupni sistem modelira UML dijagramima, a zatim se ti dijagrami primenom CASE alata neposredno prevode u izvršni kod. Tada UML postaje izvorni kod, što odgovara programskom jeziku.
Ovaj način upotrebe UML-a još nije doživeo punu praktičnu afirmaciju.

Opis jednog apleta (ZdravoSvete) pomoću UML-a

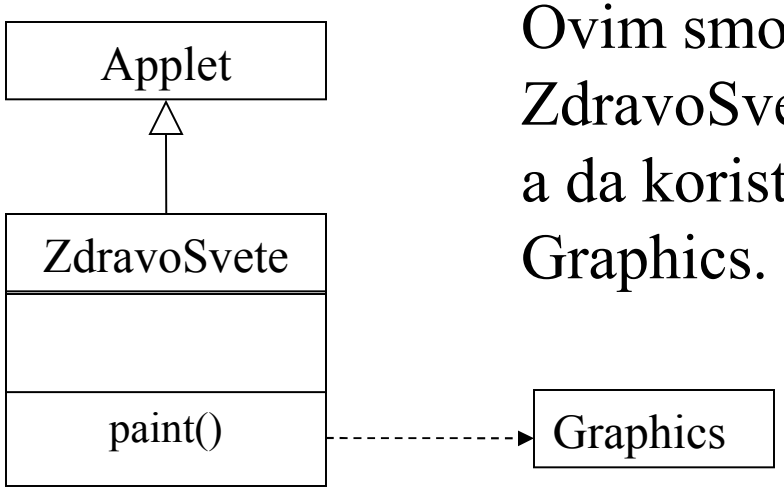
Da bismo videli kako se konkretno primenjuju UML-dijagrami, iskoristićemo razne tipove dijagrama za opis jednostavnog Java-apleta ZdravoSvete:

```
import java.awt.*;
 public class ZdravoSvete extends java.applet.Applet{
  public void paint (Graphics g) {
   g.drawString("Zdravo svete!", 5, 50);
   }
 }
```



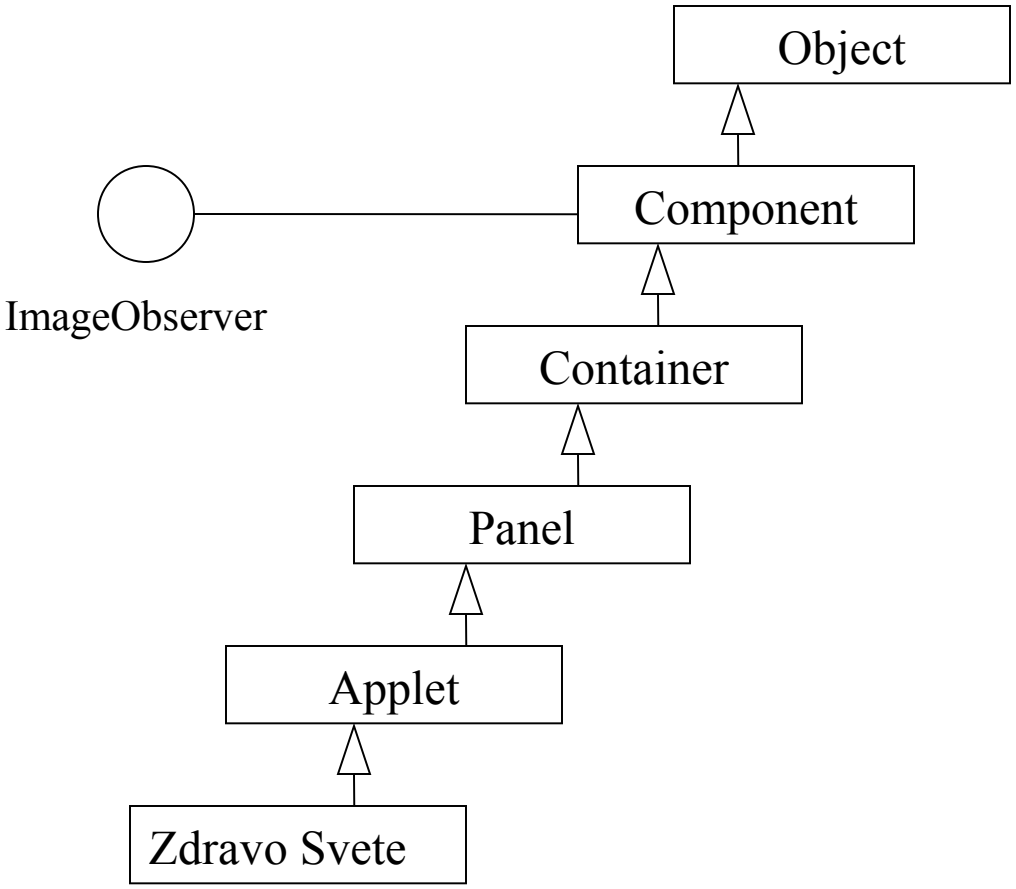
Opis osnovne klase
i metoda u apletu.

Pored osnovne klase, u apletu se koriste klase Applet i Graphics i svaka od njih ima određenu ulogu. To se ne vidi iz prethodnog dijagrama, ali to možemo navesti na sledeći način:

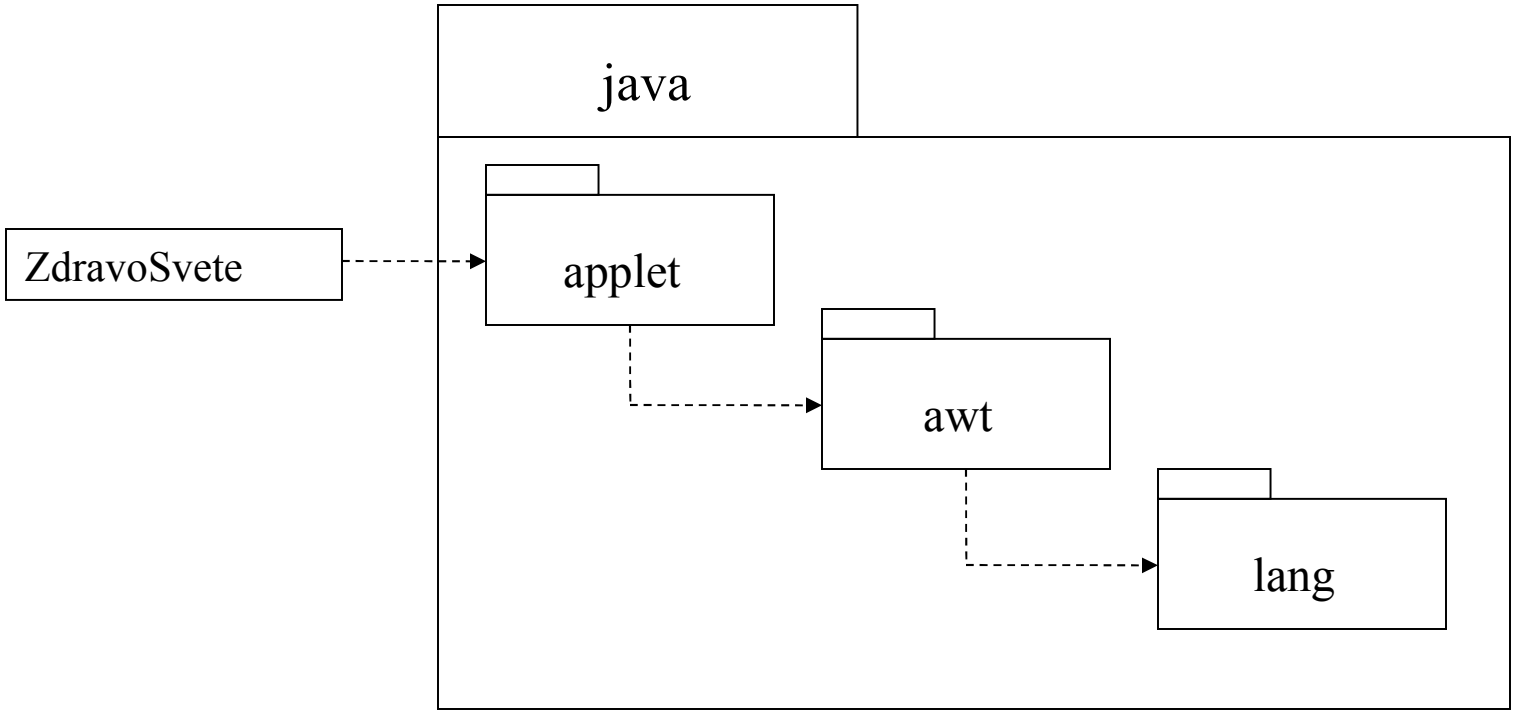


Ovim smo kazali da je klasa ZdravoSvete potklasa klase Applet, a da koristi (zavisi od) klase Graphics.

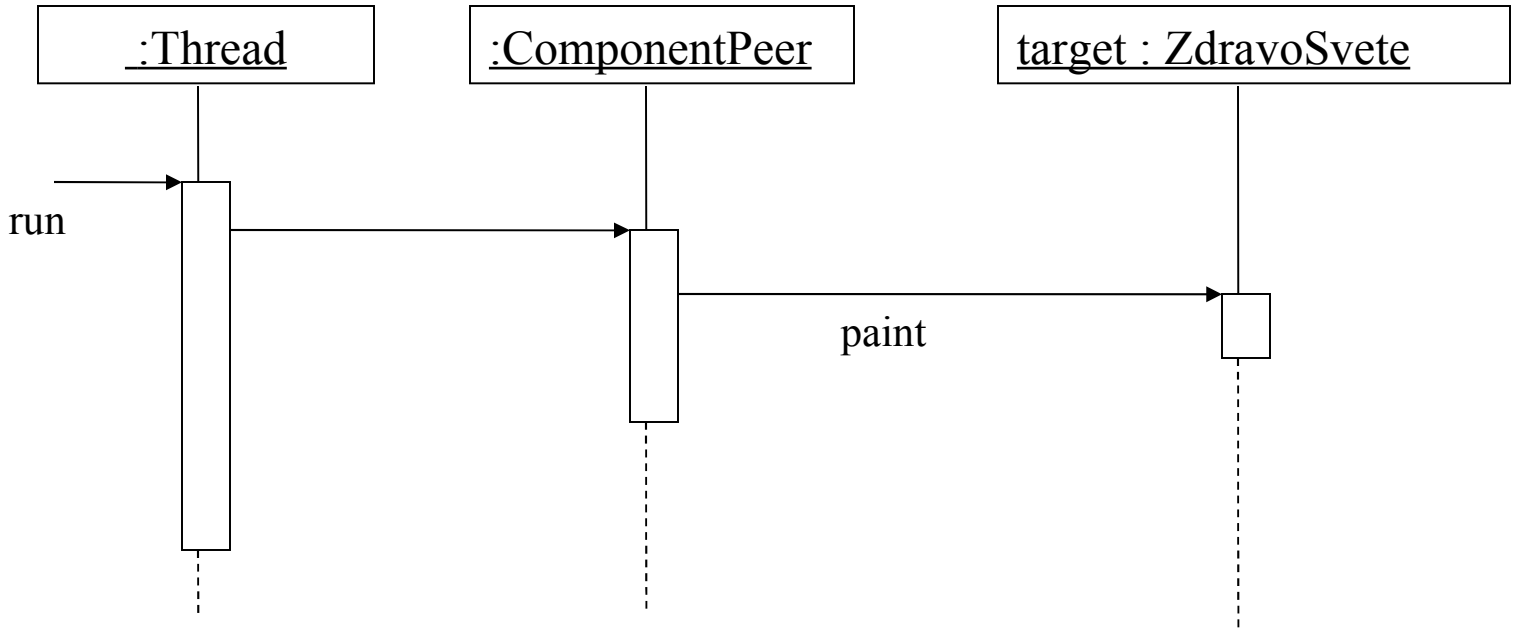
Međutim, ako bismo želeli da prikažemo kako sve ovo izgleda u okviru Java-sistema, trebalo bi prikazati i nadklase i interfejse koje nasleđuje klasa Applet:



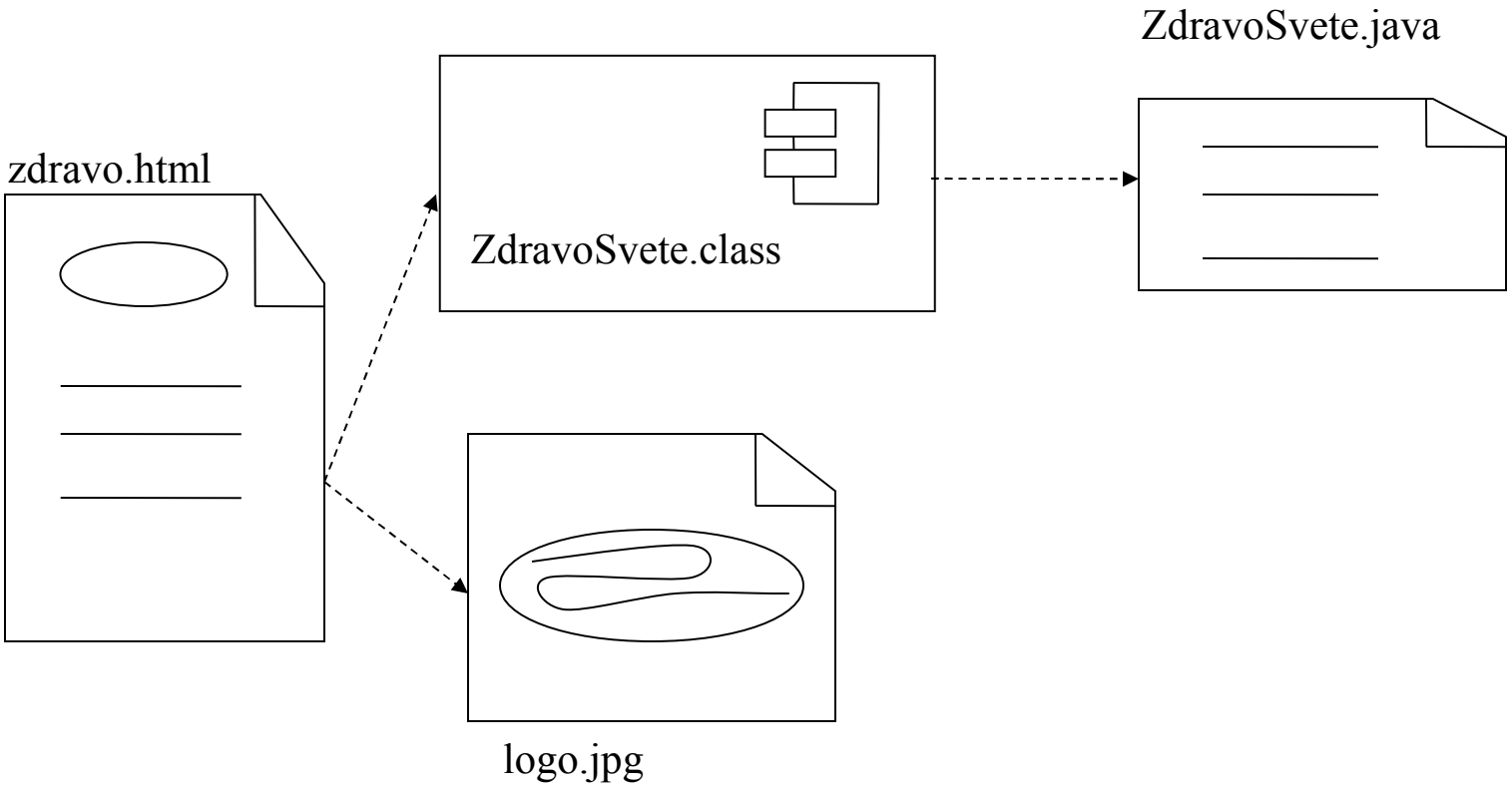
Iz navedenih dijagrama ne vidi se kako se vrši grupisanje, ali i to možemo prikazati pomoću UML-a koristeći dijagrame za grupisanje.



Da bismo videli kako aplet funkcioniše u vremenu, koriste se sekvencni dijagrami



Iz navedenih dijagrama ne vidi se kako se fizički implementira aplet. Aplet najčešće koristimo u okviru Web-strane, gde se mogu koristiti i neki drugi fajlovi. Za prikaz toga mogu se koristiti komponentni dijagrami (čak i kolaboracioni).



UML-alati

Neki od UML razvojnih alata

Rational Rose
Borland Together
Appolo for Eclipse
Enterprise Arhitect
Eclipse Uml2Tools
ArgoUML
MonoUML,
StarUML,
Microsoft Visio
Rational Software Architect
Visual Paradigm for UML

StarUML 5.0 http://staruml.sourceforge.net/en/download.php

Sistemski zahtevi (minimalni)

- Intel® Pentium® 233MHz/ Linux/W2K/128 MB RAM
- 110 MB HDD

StarUML™ je platforma za modelovanje softvera.

Podrzava UML pocev od verzije 1.4 i prihvata UML 2.0 notaciju

MODULI

- Generator modul (generisanje dokumenta i koda).
- Java modul (podrska za Java profile, J2SE/J2EE Framework, code generation, reverse engineering).
- C++ modul (podrska za C++ profile, MFC Framework, code generation, reverse engineering).
- C# modul (podrska za C# profile, .NET BCL framework, code generation, reverse engineering).
- Rose modul (podrska za citanje Rational Rose datoteka).
- Pattern modul (podrska za design pattern).

Argouml

Postoje razni alati koji omogućavaju korišćenje UML-a. Jedan od raspoloživih je **Argouml.** http://www.argouml.org/

Argouml – alat urađen u programskom jeziku Java i potpuno zasnovan na Javi. Pored toga u izradi ovog softvera intenzivno je korišćena XML-tehnologija, tj. standardi zasnovani na XML-u.

Pokreće se iz java-okruženja komandom:

java –jar argouml.jar

Argouml omogućava čuvanje celih projekata i izdvajanje dijagrama u fajlove različitog formata (jpg, svg, ...).

Argouml podržava 7 tipova dijagrama:

- klasne (class)
- korisničke (use case)
- stanja (state)
- aktivnosti (activity)
- kolaboracione (collaboration)
- razvojne (deployment)
- sekvencne (sequence)