KOMPATIBILNOST TIPOVA. POLIMORFIZAM

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

KOMPATIBILNOST TIPOVA

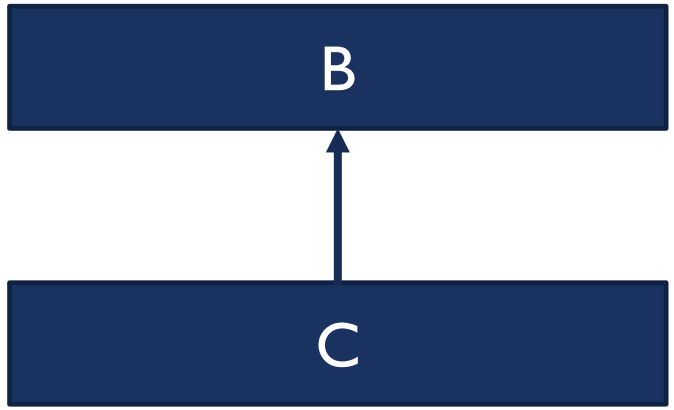
- Java je strogo tipiziran jezik

  kompatibilnost tipova se proverava tokom prevodjenja

- Pri operaciji dodele, inicijalizaciji, prenosu parametara i vraćanju rezultata metoda tip izraza mora biti kompatibilan tipu promenljive kojoj se izraz dodeljuje.
- Tip reference koja je rezultat izraza mora biti
  - Tip promenljive kojoj se dodeljuje
  - Njegov podtip
  - ili `null`

KONVERZIJA

- Nadtipovi su manje posebni, smatraju se širim i nalaze se više u hijerahiji tipova
- Konverzija proširivanja (naviše, nagore)
  - Podtip se konvertuje u nadtip
  - Automatski
- Konverzija sužavanja (naniže, nadole)
  - Nadtip se konvertuje u podtip
  - Eksplicitno (cast)

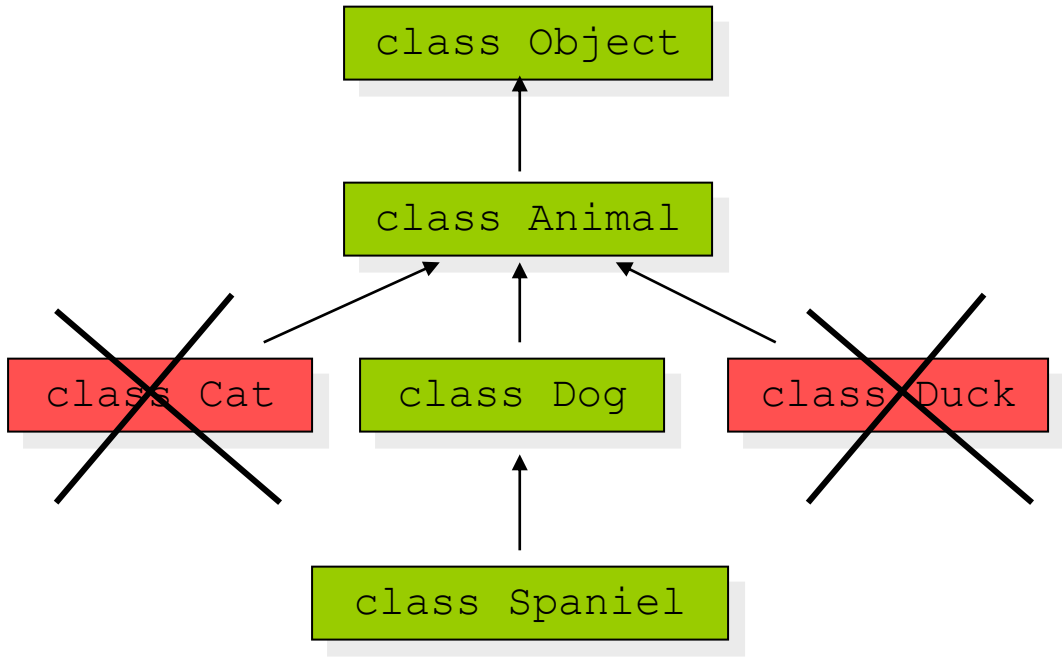



```
class D extends B { … }
…
D d 1 = new D();
B b1=new B();
B b = d1;        // naviše
D d = (D)b1;  // naniže, obavezan cast
```

KONVERZIJA SUŽAVANJA - CAST

- Cast objekata je moguć u slučaju da radimo cast izmedju klasa koje pripadaju istom lancu nasledjivanja (dakle, jedan tip mora da bude podtip drugog u bilo kom redosledu).
- Java **zadržava sve informacije o originalnoj klasi kojoj objekat pripada** !!!

```
Spaniel aPet = new Spaniel("Fang"); // Kreiraj objekat tipa Spaniel

Animal theAnimal = (Animal)aPet;

Animal theAnimal = aPet;

   // cast nagore - upward cast

Dog aDog = (Dog)theAnimal;

  // cast nadole

  // downward cast
```

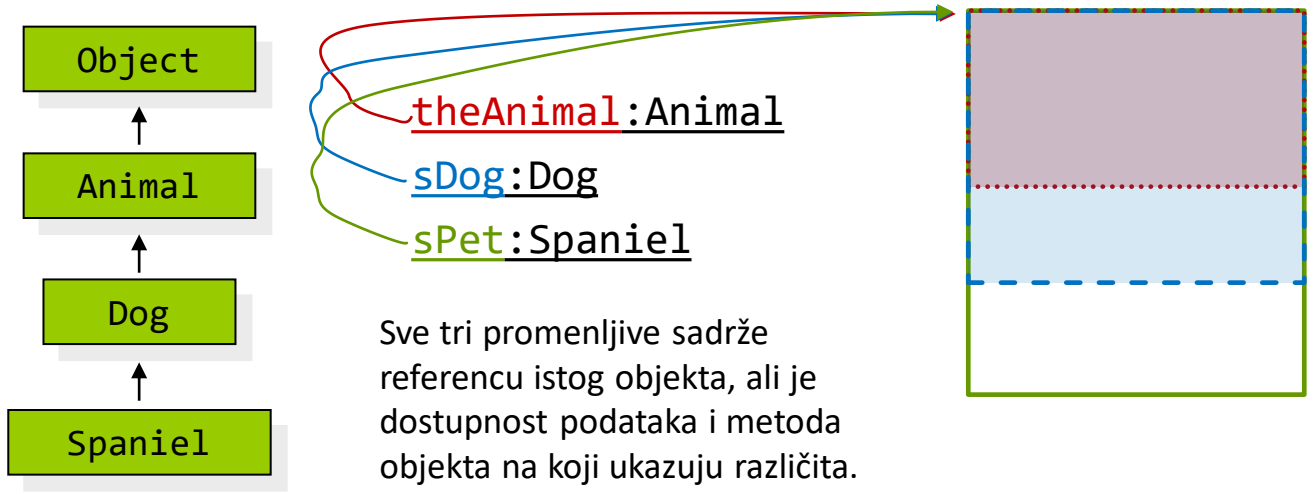


KONVERZIJA SUŽAVANJA - CAST

```
Spaniel aPet = new Spaniel("Fang");
Animal theAnimal = aPet;
Dog aDog = (Dog)theAnimal;
```

- aDog se može koristiti **SAMO** za poziv *overrided* metoda iz klase `Spaniel`. Dakle, ako `Spaniel` ima i dodatne metode, oni neće biti dostupni sa aDog refence.
- ako je potrebno pozvati metod specifičan za Spaniel klasu koristeći referencu aDog, NEOPHODAN je cast aDog-a na Spaniel.

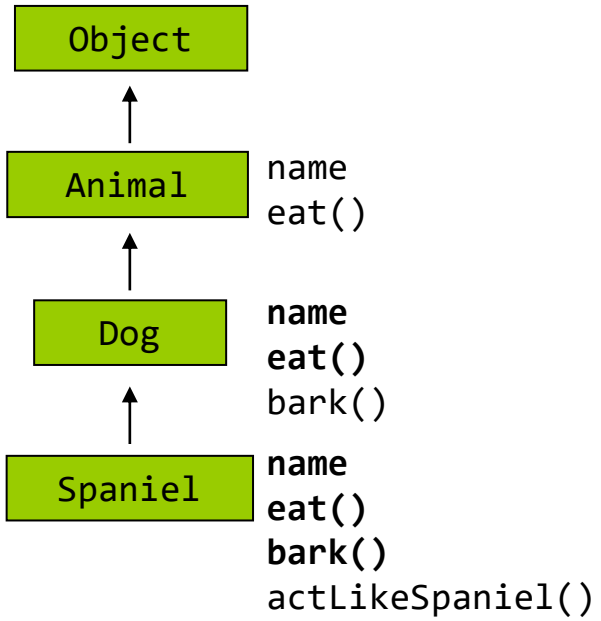
```
((Spaniel)aDog).specmetod();
```



Sve tri promenljive sadrže referencu istog objekta, ali je dostupnost podataka i metoda objekta na koji ukazuju različita.

POZIV PREPISANIH METODA

```
Spaniel aPet = new Spaniel("Fang"); // Kreiraj objekat tipa Spaniel
Animal theAnimal = aPet;
Dog aDog = (Dog)theAnimal;
aDog.bark();
```



Slanjem poruke

`aDog.bark();`

biće pozvan metod objekta na koji ukazuje referenca aDog, a to je objekat klase `Spaniel`.

Nestatički metodi se vezuju dinamički!

POLIMORFIZAM

- mogućnost da varijablom određenog tipa referenciramo objekte različitih tipova i da automatski pozivamo metode koje su specifične za tip objekta na koji varijabla referencira
- uslovi
  - Poziv metoda podklase kroz varijablu bazne klase
  - Pozvana metoda mora biti i član bazne klase
  - Signatura metode i povratni tip moraju biti isti i u baznoj i u izvedenoj klasi
  - Atribut pristupa ne sme biti restriktivniji u izvedenoj klasi nego što je u baznoj klasi
  - Polimorfizam se odnosi samo na metode – reference baznog tipa mogu se koristiti samo za pristup podacima članovima baznog tipa

DODATAK – LINKING AND LOADING

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

STATIČKO I DINAMIČKO VEZIVANJE / EARLY AND LATE BINDING

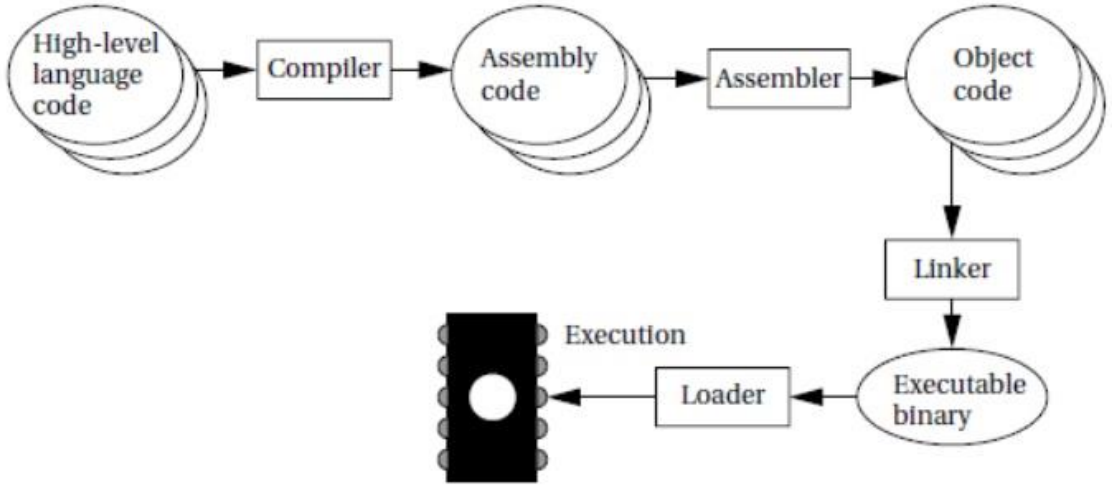
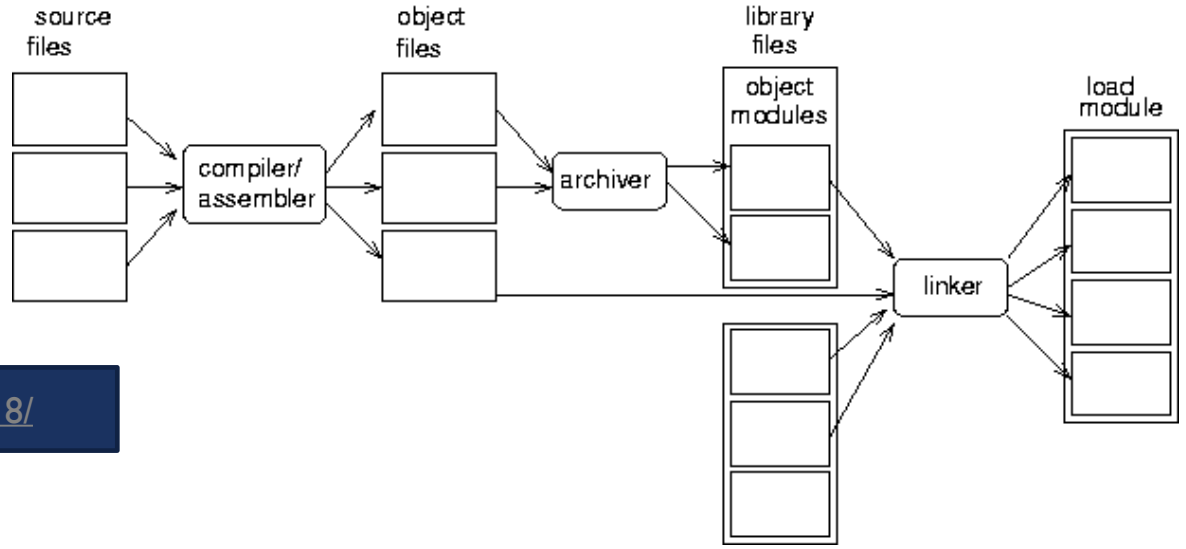


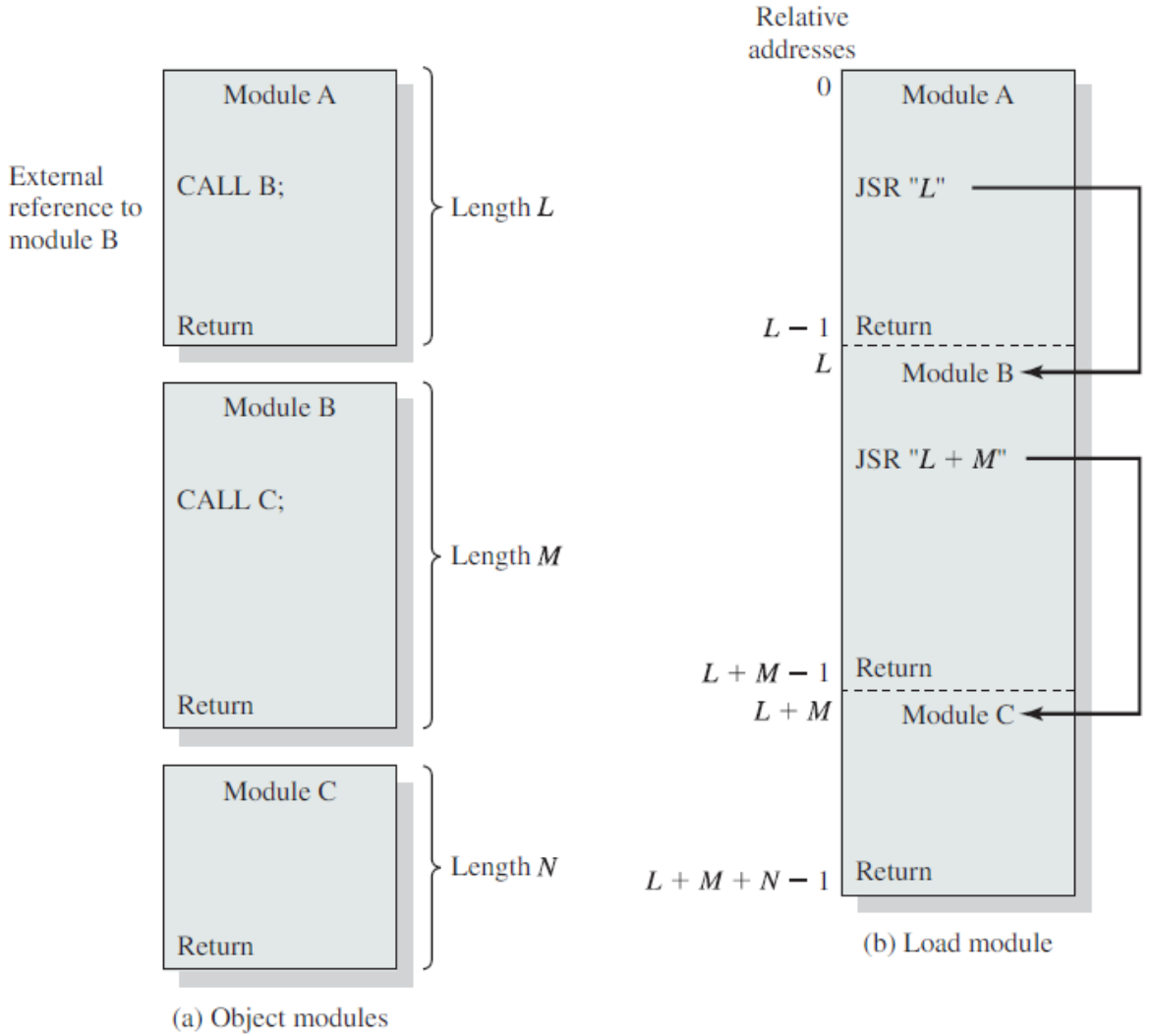
Fig 2.16 Program generation from compilation through loading.



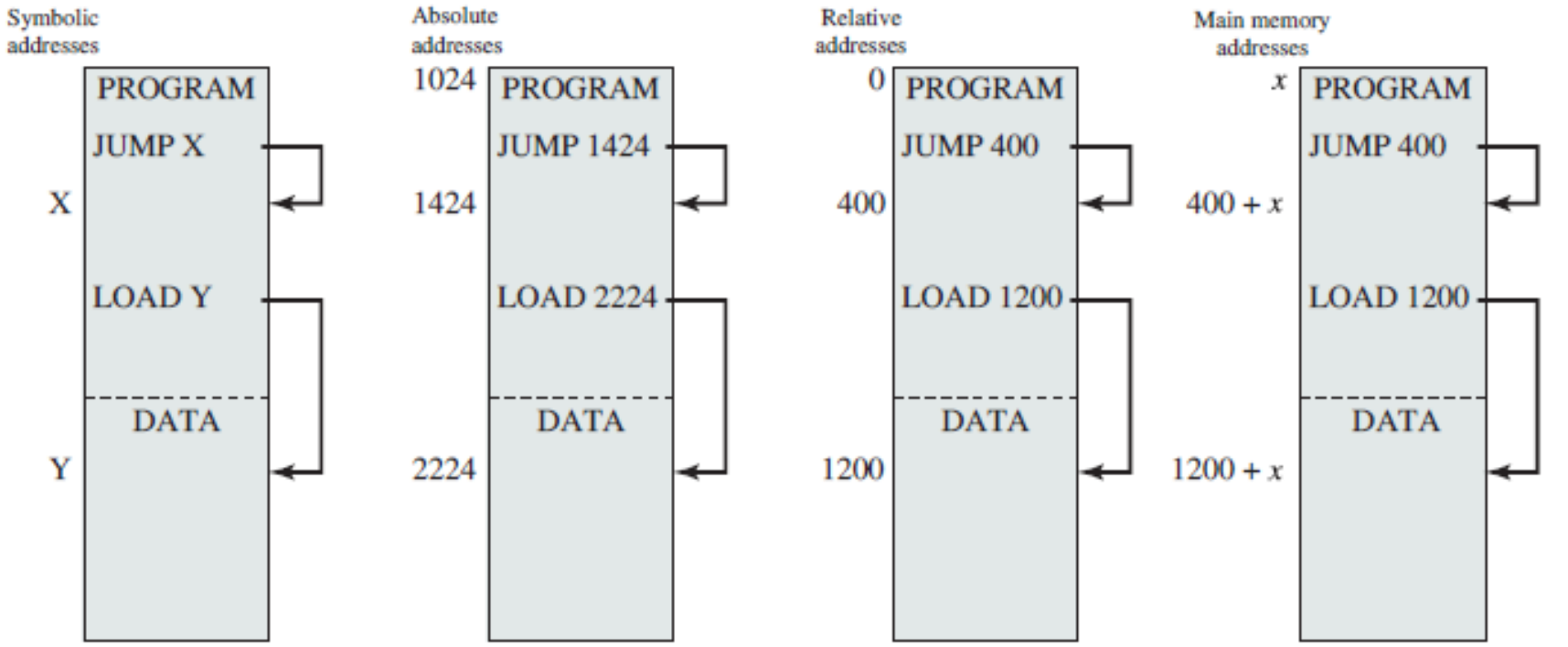
Više na https://www.brainkart.com/article/Assembly-and-Linking_11818/

LINKING - VEZIVANJE

- Aplikacija sadrži jedan ili više objektnih modula. Moduli se međusobno referenciraju/pozivaju. Treba razrešiti pozive.
- Uloga linkera da je od kolekcije objektnih modula napravi Load modul i preda ga loaderu na učitavanje.



LINKING - VEZIVANJE



(a) Object module

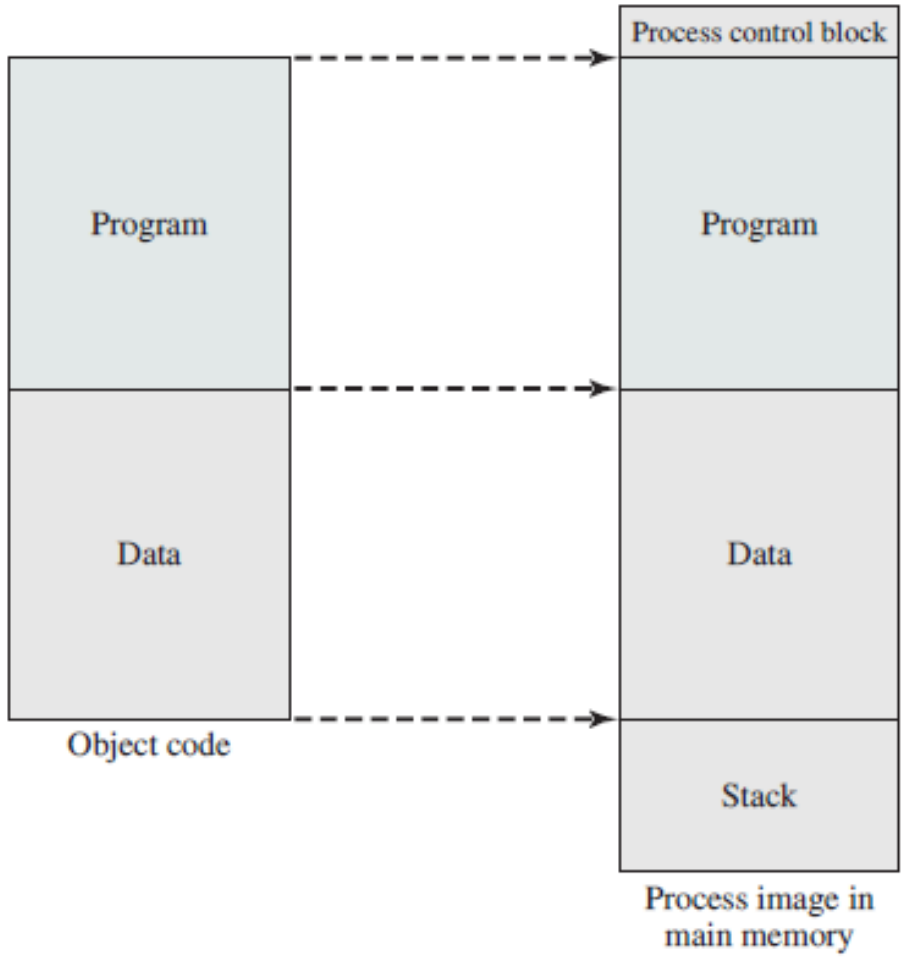
(b) Absolute load module

(c) Relative load module

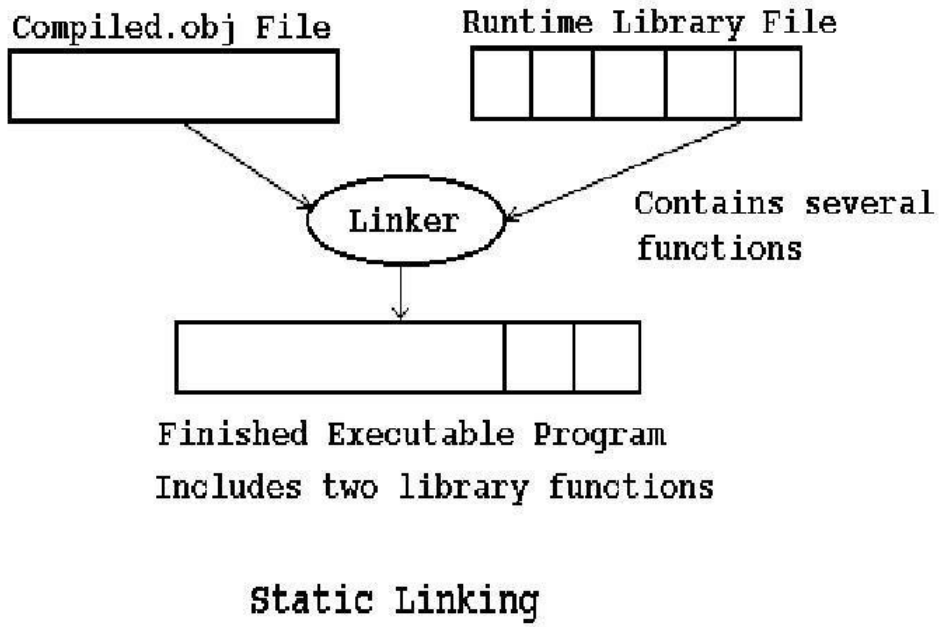
(d) Relative load module loaded into main memory starting at location x

LOADING - UČITAVANJE

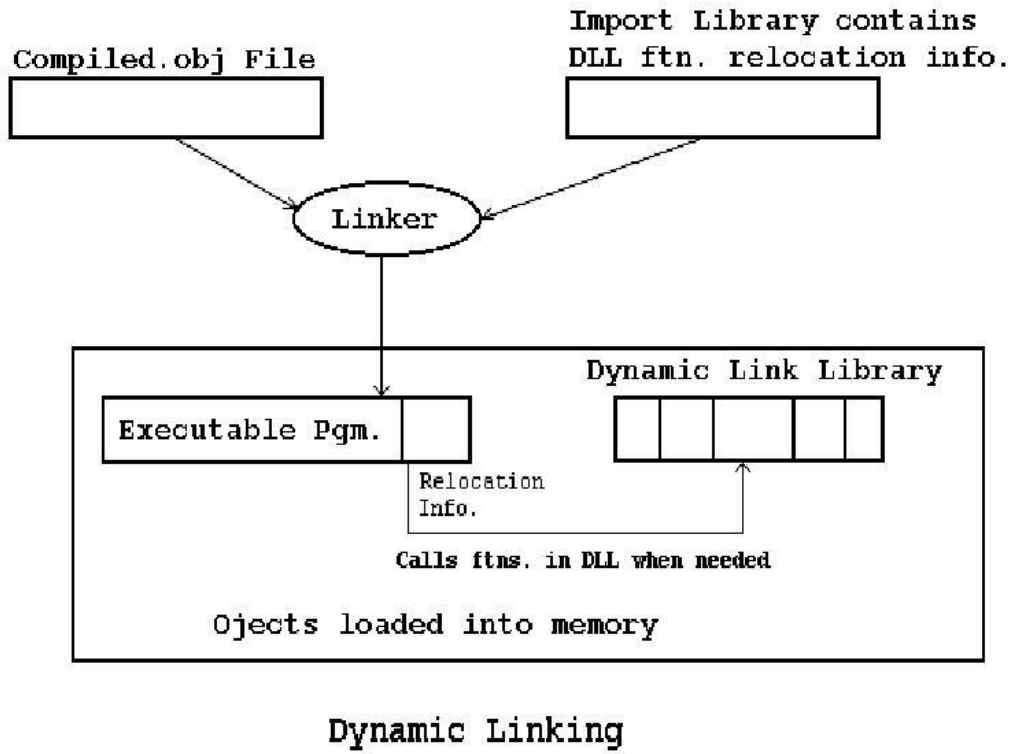
- Prvi korak u krereiranju aktivnog procesa jeste njegovo **učitavanje** u glavnu memoriju i kreiranje slike procesa.



STATIČKO I DINAMIČKO VEZIVANJE / EARLY AND LATE BINDING

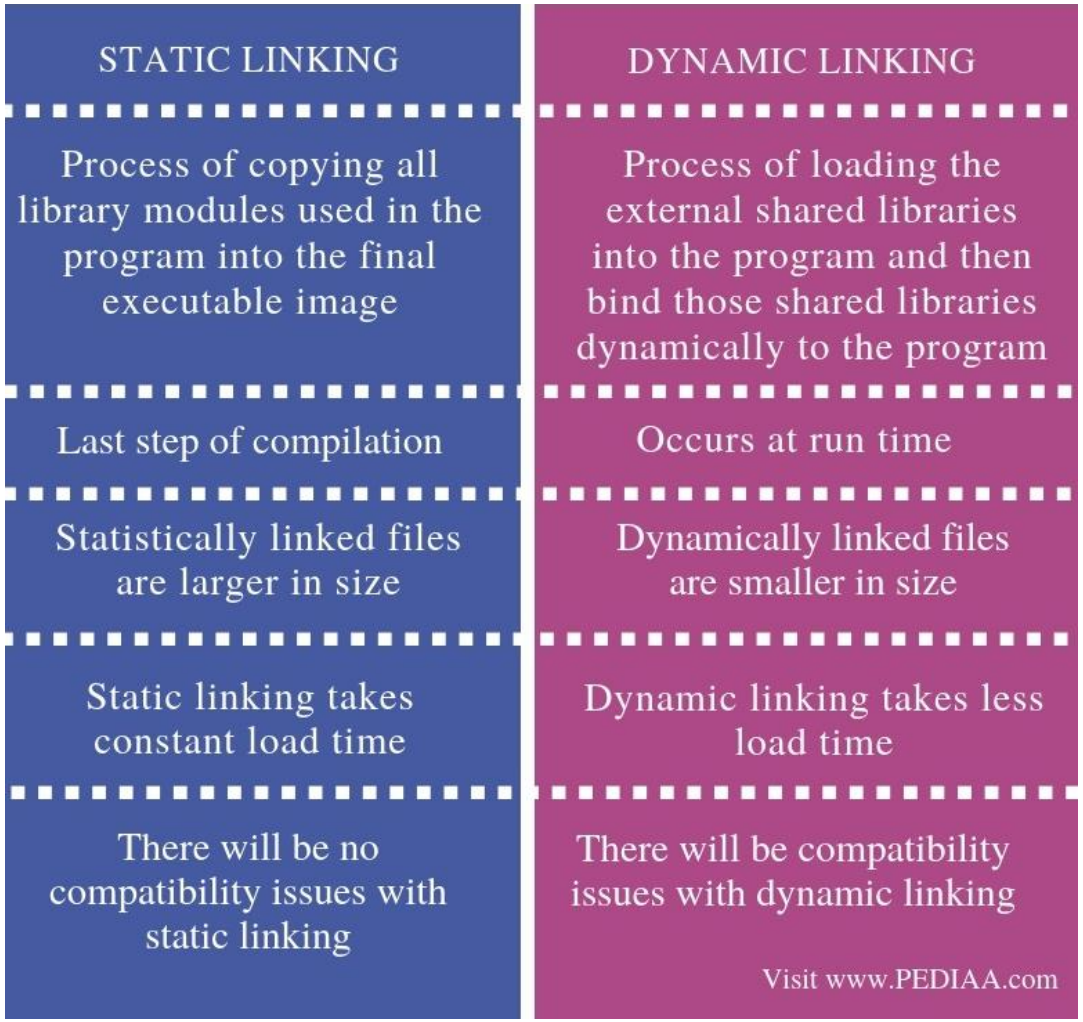


Static Linking



Dynamic Linking

STATIČKO I DINAMIČKO VEZIVANJE / EARLY AND LATE BINDING



Binding is a word that is used in more than one context. It always has to do with the connecting of one thing to another however when the act of binding happens can vary.

There is a concept of Binding Time or the point at which some component is bound to some other component. A basic list of binding time is: (1) binding at compile time, (2) binding at link time, (3) binding at load time, and (4) binding at run time.

Binding at compile time happens when the source code is compiled. For C/C++ there are two main stages, the Preprocessor which does source text replacement such as define replacement or macro replacement and the compilation of the source text which converts the source text into machine code along with the necessary instructions for the linker.

Binding at link time is when the external symbols are linked to a specific set of object files and libraries. You may have several different static libraries that have the same set of function names but the actual implementation of the function is different. So you can choose which library implementation to use by selecting different static libraries.

Binding at load time is when the loader loads the executable into memory along with any dynamic or shared libraries. The loader binds function calls to a particular dynamic or shared library and the library chosen can vary.

JAVA - LINKING

Statički metodi – statičko vezivanje

Nestatički metodi – dimamičko vezivanje