Hijerarhijska dekompozicija

- Ovakav pristup dosta se koristi i u složenim programima na jeziku C (npr. implementaciji operativnih sistema) na sledeći način:

```
struct ListElem {...};
struct Runnable {...};
struct Drawable {...};

struct Task {
  ListElem listElem;
  Runnable runnable;
  Drawable drawable;
  ...
};
```

- Dakle, instanca strukture *Task* u sebi sadrži podstrukture koje predstavljaju odgovarajuće interfejse. Kada instancu strukture *Task* treba koristiti u nekom od ovih konteksta, dostavlja se pokazivač na odgovarajuću ugrađenu podstrukturu:

```
Task* aTask = ...;

List* taskList = ...;

addAtTail(taskList, &aTask->listElem);
```

- Sa idejom da podrži ovakve načine korišćenja, ali i da implementacija u njima bude podjednaka (i podjednako efikasna) kao ova na jeziku C, jezik C++ zapravo ima koncept *izvedenih* klasa (*derived class*), sa sledećim značenjem:
  - klasa može biti *izvedena* iz više osnovnih klasa koje su navedene u zaglavlju definicije klase, iza dvotačke
  - svaki objekat izvedene klase u sebi sadrži po jedan podobjekat svake od tih osnovnih klasa
  - specifikator pristupa (*public*, *protected*, *private*) označava dostupnost dog podobjekta, na isti način kao i za članove te klase

Hijerarhijska dekompozicija

- Na taj način, ako je osnovna klasa u zaglavlju definicije izvedene klase označena kao *javna* (*public*) osnovna klasa, ovaj podobjekat osnovne klase dostupan je (implicitnom) konverzijom pokazivača (ili reference) na tu izvedenu klasu u pokazivač (ili referencu) na tu osnovnu klasu na svim mestima gde su dostupne i te klase:

```
class ListElem {…};

class Task : public ListElem {…};

Task* aTask = …;

List* taskList = …;

taskList->addAtTail(aTask);
```

- Osim toga, svi javni članovi javne osnovne klase jesu i javni članovi izvedene klase, pa se može raditi npr:

```
aTask->insert(…)
```

- Zaštićeni članovi osnovne klase dostupni su u izvedenoj klasi, ali ne i van nje i ostaju zaštićeni i dalje; privatni članovi osnovne klase nisu dostupni u izvedenoj klasi
- Dakle, važi pravilo supstitucije i sve što se može uraditi sa objektom osnovne klase može se uraditi i sa objektom izvedene klase - *javno izvođenje klasa* na jeziku C++ jeste jezički koncept koji realizuje *nasleđivanje* kao objektni koncept