

\*C++M

Jeżeli nie chcemy, żeby dana klasa była użyta do tworzenia klasy pochodnej, to my definiujemy klasę małązko **FINAL**

## KOLEJNOŚĆ WYWOŁYWANIA KONSTRUKTORÓW

- 1 SENIORZY (konstruktory klasy podstawowej)
- 2 GOŚCIE (konstruktory obiektów pochodzących z klasy podstawowej)
- 3 KLASA POCHODNA

DZIEDZICZENIE OD 71 RODZICA

WIELODZIEDZICZENIE

DEFINIOWANIE NOWEJ KLASY PRZY WYKORZYSTANIU KLASY ISTNIEJĄCEJ

DOTYCZY KLAS CYPÓW, NIE (KONKRETYCH) OBIEKTÓW

**DZIEDZICZENIE**  
TWORZENIE KLAS POCHODNYCH  
USTALENIE HIERARCHII

LISTA POCHODZENIA KLASY

`class Pochodna : public class Podstawa`

- inf. 2 jakiej klasy wyodził się klasa
- jakim sposobem jest dziedziczona

KLASA POCHODNA (KLASA POTOMEK)

- definiowanie dodatkowych danych
- definiowanie dodatkowych funkcji
- zdefiniować stałe (najczęściej funkcje stałe istniejące już w klasie podstawowej (kolekta tego, co nam nie odpowiada) ZASTĘPIENIE STANOWI

KLASA PODSTAWOWA

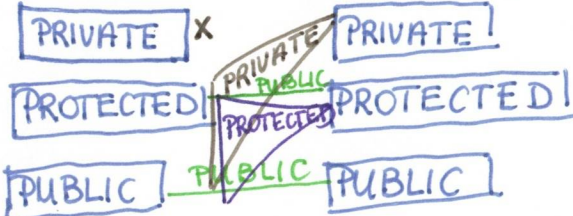
(KLASA, KTORĄ SIĘ DZIEDZICZY)

• SKŁADNIKI O ETYKIETACH:

PRIVATE - brak dostępu z klasy pochodnej

PROTECTED - ma dostęp rodzina - klasy pochodne

PUBLIC - publiczne



- KONSTRUKTORÓW
- OPERATORÓW PRZYPISANIA
- DESTRUKTORÓW

NIE

DZIEDZICZYM

## FUNKCJE PRZECIADAJĄCE

- TA SAMA NAZWA
- TEN SAM ZAŁOŻENIE WAŻNOŚCI (np. definicja tej samej klasy)

## FUNKCJE WIRTUALNE (ZACIERAJĄCE SIĘ)

- TA SAMA NAZWA
- TA SAMA LISTA ARGUMENTÓW
- LEŻĄ W RÓŻNYCH KLASACH TEJ SAMEJ HIERARCHII (a ta f. w klasie podstawowej ma słowo kluczowe virtual)

## FUNKCJE ZASTĘPIAJĄCE

- TA SAMA NAZWA
- LEŻĄ W RÓŻNYCH KLASACH TEJ SAMEJ HIERARCHII
- MAJĄ DOWOLNE (nawet różne) LISTY ARGUMENTÓW

LISTY ARGUMENTÓW