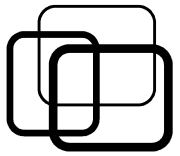
Inheritance

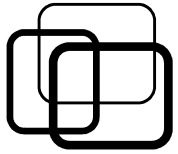
Inst. Nguyễn Minh Huy

Contents



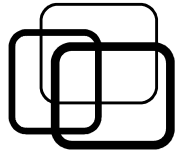
- Basic concepts.
- Access control in inheritance.
- Method overriding.
- IS-A and HAS-A relationships.

Contents



- **Basic concepts.**
- Access control in inheritance.
- Method overriding.
- IS-A and HAS-A relationships.

Basic concepts



- Redundancy problem:
  - Two classes have same information.
  - Types of redundancy:
    - Sharing: $A \cap B \neq \emptyset$.
    - Extension: $B = A + \varepsilon$.
  - Disadvantages:
    - Time.
    - Storage.
    - Change.

A
x, y, z
a, b, c

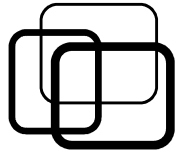
B
x, y, z
u, v, w

A
x, y, z

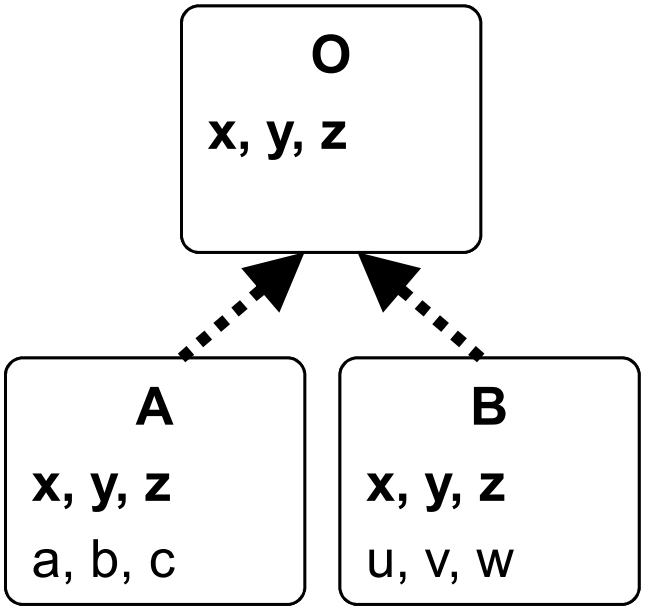
B
x, y, z
a, b, c

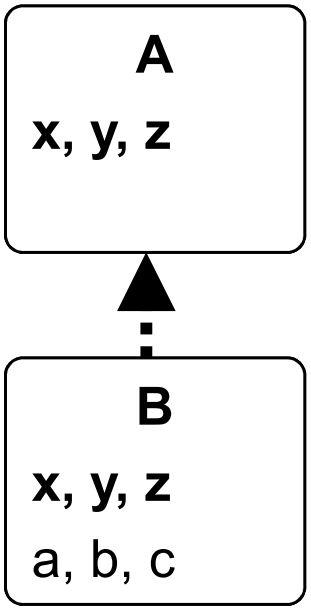
Solution: reusability!!

Basic concepts

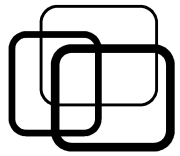


- Inheritance concept:
  - Construct new class based on existing classes.
  - **Derived class:** new class constructed on old ones.
  - **Base class:** old class used to construct new ones.
  - Derived class inherits **ALL** from base class.





Basic concepts



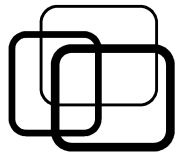
- C++ usage:

  class **<derived class> :** <*inheritance type*> **<base class>**

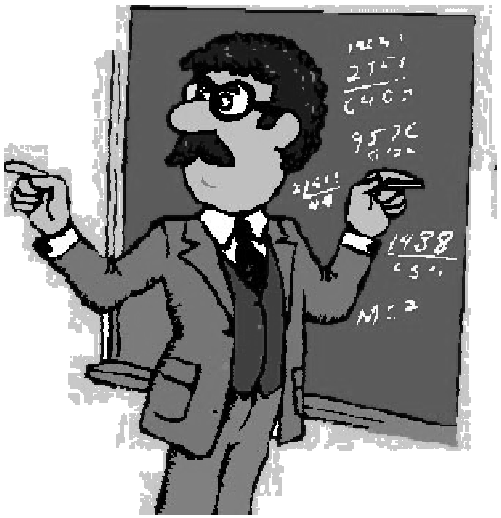
- Inheritance types:
  - public, private, protected.
- Example:

```
class A : public O
{
private:
    // New attributes of A.
public:
    // New methods of A.
};
```

Basic concepts



- Example:



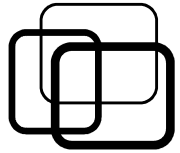
Teacher

- Attributes:
  - Name.
  - Salary.
  - Vacation.
- Methods:
  - Teach.
  - Calc salary.

- Attributes:
  - Name.
  - Salary.
  - Vacation.
  - **Classroom.**
- Methods:
  - Teach.
  - Calc salary.
  - **Meet students.**

HRTeacher

Basic concepts



- Example:

```
class Teacher
{
private:
        char      *m_name;
        float     m_salary;
        int       m_vacation;
public:
        Teacher(char  *name,
                  float  salary,
                  int  vacation);
        void teach();
        float calcSalary();
};
```

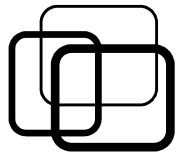
Derived class

Base class

```
class HRTeacher : public Teacher
{
private:
        char      *m_classRoom;
public:
        HRTeacher(char  *name,
                  float  salary,
                  int  vacation,
                  char  *classRoom);
        void meetStudents();
};
```

HRTeacher inherits ALL attributes and methods from Teacher

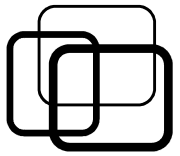
Basic concepts



- Example:

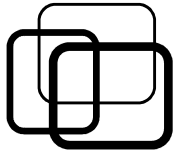
```
int main()
{
        Teacher   t1("John", 1000, 5);
        t1.teach();
        float  sal1 = t1.calcSalary();

        HRTeacher   t2("Peter", 2000, 3, "Room 101");
        t2.teach();                          // Reuse from Teacher.
        t2.meetStudents();
        float  sal2 = t2.calcSalary();       // Reuse from Teacher.
}
```

Contents

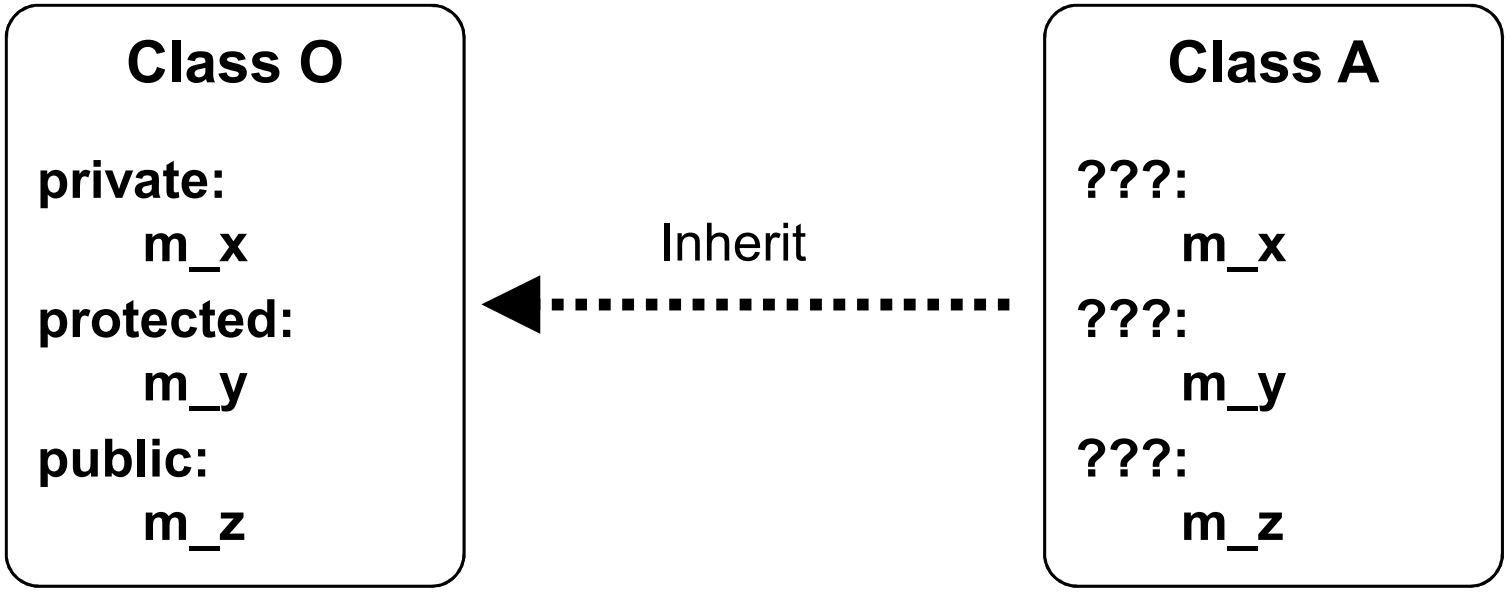


- Basic concepts.
- **Access control in inheritance.**
- Method overriding.
- IS-A and HAS-A relationships.

Access control in inheritance

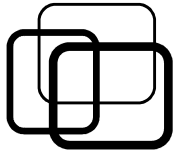


- Class A inherits from class O:
  - A inherits all attributes and methods from O.
  - Do access control changed during inheritance?



➔ **Decided by inheritance type!!**

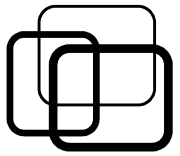
Access control in inheritance



- Access control in inheritance:

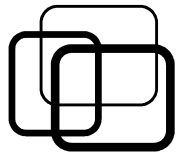
Accesibility	**public inheritance**	**protected inheritance**	**private inheritance**
public	public	protected	private
protected	protected	protected	private
private	***inaccessible***	***inaccessible***	***inaccessible***

Contents



- Basic concepts.
- Access control in inheritance.
- **Method overriding.**
- IS-A and HAS-A relationships.

Method overriding

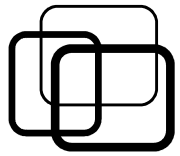


- Partial inheritance:
  - Do not want to inherit all from base??
  - **Derived class can change what are inherited!!**

    ➔ Method overriding (method redefining).

Derived class inherits ALL attributes and methods from base class EXCLUDING overridden methods!!

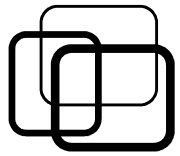
Method overriding



- Example:
  - HRTeacher inherits Teacher.
  - HRTeacher salary differs from Teacher.
    - Teacher salary = Salary – Vacation * 10.
    - HRTeacher salary = Teacher salary + Bonus 100.

  ➔ Redefine calcSalary() in HRTeacher class.

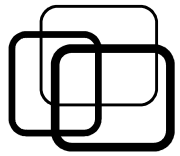
Method overriding



- Ví dụ:

```
class Teacher
{
private:
        char    *m_name;
        float   m_salary;
        int     m_vacation;
public:
        Teacher(char  *name, float  salary, int   vacation);
        void teach();
        float calcSalary()
        {
                return m_salary – m_vacation * 10;
        }
};
```

Method overriding

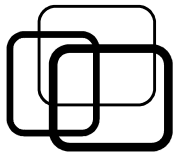


- Ví dụ:

```
class HRTeacher : public Teacher
{
private:
        char      *m_classRoom;
public:
        HRTeacher(char  *name,
                  float  salary,
                  int  vacation,
                  char  *classRoom);
        void meetStudents();
        float calcSalary()
        {
            // Redefine and reuse from base class.
            return Teacher::calcSalary() + 100;
        }
};
```

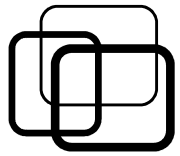
```
int main()
{
        Teacher   t1("John", 1000, 5);
        float  sal1 = t1.calcSalary();

        HRTeacher   t2("Peter", 2000, 3,
                                "Room 101");
        float  sal2 = t2.calcSalary();
}
```

Contents



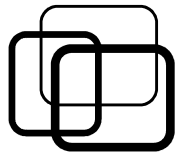
- Basic concepts.
- Access control in inheritance.
- Method overriding.
- **IS-A and HAS-A relationships.**

IS-A and HAS-A relationships



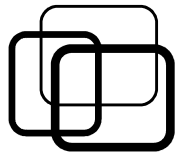
- IS-A relationship:
  - Class A IS-A class B
    - A is a special case of B.
    - A and B are same type.
- Example:
  - HRTeacher is a special Teacher.
  - Square is a special case of Rectangle.
  - Cat is a special case of Animal.

IS-A and HAS-A relationships



- HAS-A relationship:
  - Class A HAS-A class B
    - A contains B.
    - B is a part of A.
- Example:
  - Vehicle contains Wheel.
  - Page is a part of Book.

IS-A and HAS-A relationships



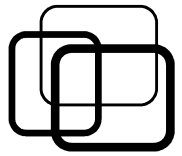
- Dr. Guru advises: **Class Construction Rule**
  - A IS-A B.

    ➔ Let A inherit B.
  - A HAS-A B.

    ➔ Let B be an attribute of A.
- Example:

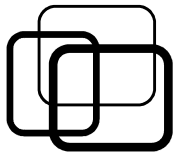
```
class Cat : public Animal { };
class Vehicle
{
private:
        Wheel   *m_wheels;
};
```

Summary



- Inheritance concepts:
  - Construct new class based on existing classes.
  - Derived class inherits ALL from base class.
- Access control in inheritance:
  - Inheritance type control access of inherited members.
- Method overriding:
  - Redefine inherited methods.
- IS-A and HAS-A relationships:
  - IS-A: A is a special case of B.
  - HAS-A: A contains B.

Practice

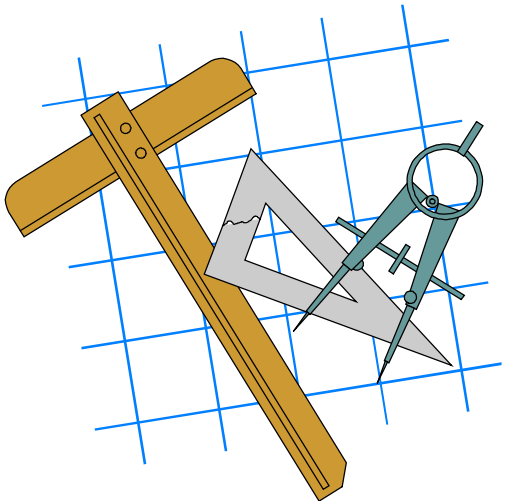


- ## Practice 7.1:

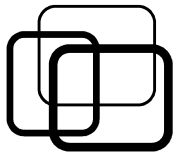
  Identify relationships (IS-A or HAS-A) of following pairs of classes.

  Write declarations for each pair.

  1. Square / Rectangle.
  2. Polygon / Edge.
  3. Manager / Employee.
  4. Circle / Ellipse.
  5. Airplane / Engine.
  6. Setence / Word.
  7. Cosmetic / Goods.
  8. Rice / Food.
  9. Library / Book.
  10. Cartoon / Movie.



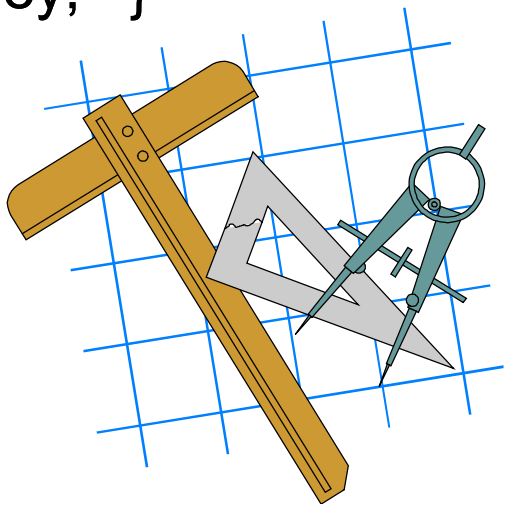
Practice



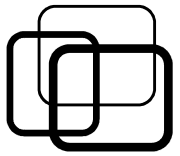
- Practice 7.2:

  Given class **Account** as follow:

```
class Account
{
private:
        float      m_balance;
public:
        float getBalance() {   return m_balance;   }
        void deposit(float  money) {   m_balance += money;   }
        bool withdraw(float money) {
            if (money > m_balance)
                    return false;
            m_balance -= money;
            return true;
        }
};
```



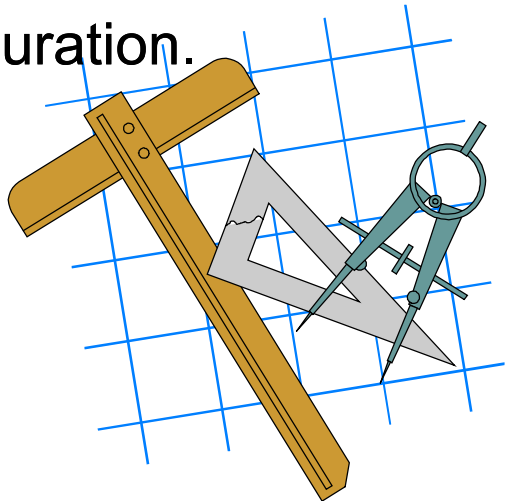
Practice



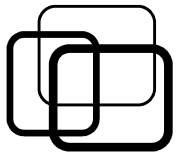
- Practice 7.2:

  Construct class **SavingAccount** based on class **Account** as follow:

  - Additional information:
    - Period (months).
    - Rate (yearly percentage).
    - Duration from last deposit or withdraw (months).
  - Calculate interest at current time (based on duration).
  - Deposit: update balance with interest, reset duration.
  - Withdraw: update balance with interest, reset duration.
  - Increase duration by one month.



Practice



- ## Practice 7.3:

  A motor-bike consumes 2 lit of fuel for 100 km, consumes more 0.1 lit for every additional 10 kg of goods.
  A truck consumes 20 lit of fuel for 100 km, consumes more 1 lit for every additional 1000 kg of goods.

  Using inheritance to construct class **MotorBike** and **Truck** that can do the followings:

  - Add a weight of goods to the vehicle.
  - Remove a weight goods from the vehicle.
  - Add an amount of fuel to the vehicle.
  - Run the vehicle a length of km.
  - Get the current fuel left in the vehicle.