Chapter 13

Object-Oriented Programming: Polymorphism

OBJECTIVES

- What polymorphism(多态) is, how it makes programming more convenient, and how it makes systems more extensible and maintainable.
- To declare and use virtual functions(虚函数) to effect polymorphism.
- The distinction between abstract and concrete classes(抽象类和具体类).
- To declare pure virtual functions(纯虚函数) to create abstract classes.

Topics

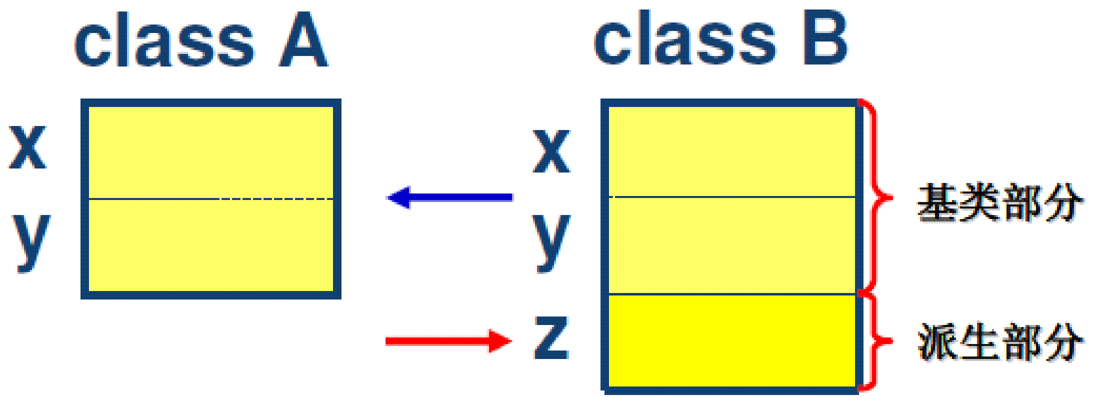
13.1 Introduction—基础

❑class B 继承 class A, 即B *is-a* A

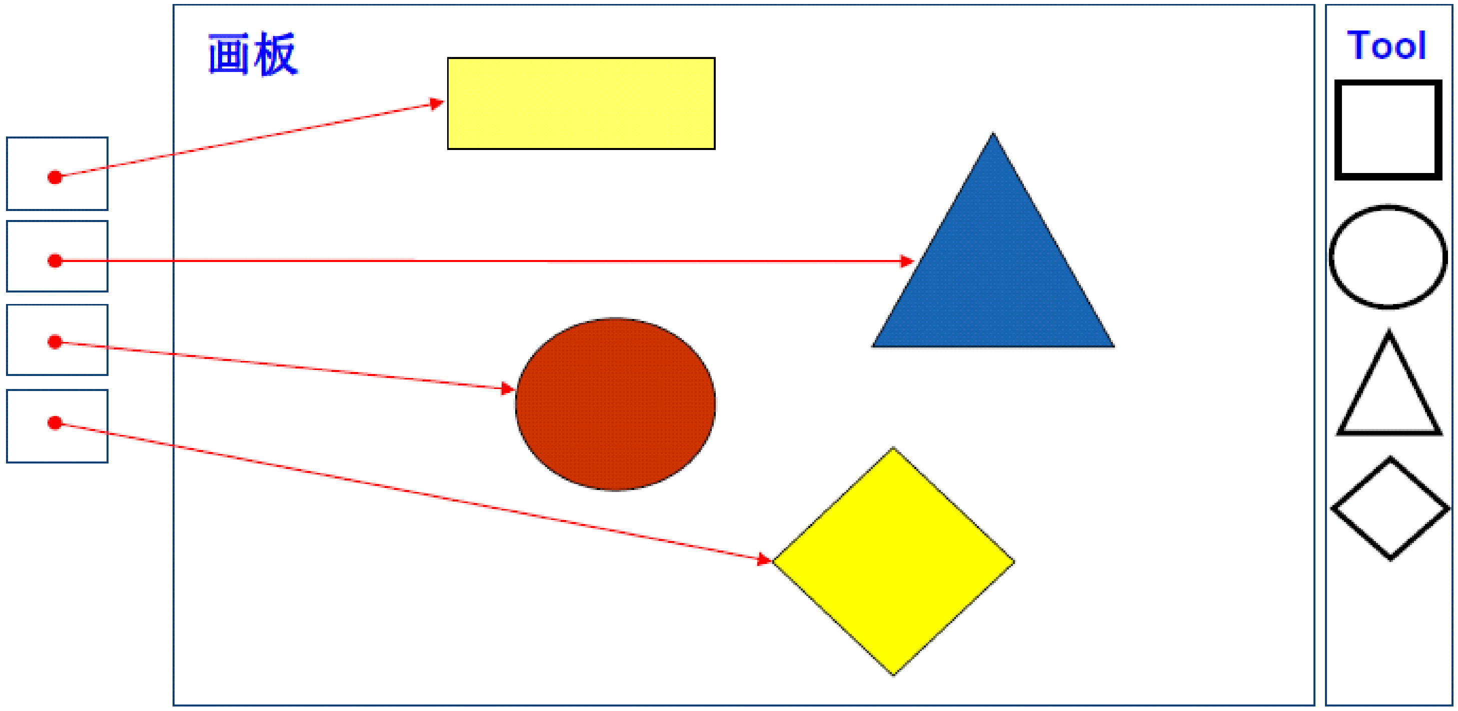
1. B big;
2. A small = big;
3. A &refSmall = big;
4. A *pSmall = &big;

1. big = small;
2. B *pBig = &small;

错误



13.1 Introduction—需求



- 基类Shape: 派生出长方形、椭圆形、三角形、菱形等
- 用vector或者array来保存指针
  - • 基类指针vs 派生类指针
- 当需刷新画板时, 枚举指针并调用各自的draw函数

13.1 Introduction

- 用户通过键盘输入多个员工信息, 统计收入数据:
- (1) CommissionEmployee

  name, ssn, grossSales, commisionRate
- (2) BasePlusCommissionEmployee

  name, ssn, grossSales, commisionRate, baseSalary
- 用vector或者array来保存指向员工对象的指针
  - • CommissionEmployee Pointer
- 希望通过这些指针来调用各自的earnings()函数以进行统计

13.1 Introduction

- Polymorphism(多态):
- 通过指向派生类的基类指针, 调用派生类的函数; 将不同的派生类对象都当作基类来处理, 可以屏蔽不同派生类对象之间的差异, 写出通用的代码, 进行通用化编程, 以适应需求的不断变化
- Virtual Function(虚函数)
- Pure Virtual Function(纯虚函数): 没有给出实现的虚函数
- Abstract Class(抽象类) vs Concrete Class(具体类)

Topics

- 13.1 Introduction
- **13.2 Relationships Among Objects in an Inheritance Hierarchy**
- 13.3 Abstract Classes and Pure virtual Functions
- 13.4 Case Study: Payroll System Using Polymorphism

13.2 Relationships Among Objects in an Inheritance Hierarchy

- 13.2.1 Invoking Base-Class Functions from Derived -Class Objects (基类指针指向派生类, 调用基类函数)
- 13.2.2 Aiming Derived-Class Pointers at Base-Class Objects (派生类指针指向基类, 错误)
- 13.2.3 Derived-Class Member-Function Calls via Base -Class Pointers (基类指针指向派生类, 调用派生类函数, 错误)
- 13.2.4 Virtual Functions (应用虚函数, 解决上述问题)
- 13.2.5 Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers (基类/派生类对象和指针之间的赋值)

13.2 Relationships Among Objects in an Inheritance Hierarchy

- 基类CommissionEmployee

  void print() const;
- 派生类BasePlusCommissionEmployee

  void print() const;
- 通过指向派生类的基类指针, 调用的是基类的函数
- 结论: 对于普通成员函数, 调用基类还是派生类的函数, 取决于句柄的类型, 而不是句柄指向的实际对象类型

13.2 Relationships Among Objects in an Inheritance Hierarchy

- 派生类指针指向基类对象

  Compilation Error

程序解读（P417）

13.2 Relationships Among Objects in an Inheritance Hierarchy

- 13.2.1 通过指向派生类的基类指针, 调用的是基类的函数;
- 可否调用派生类自有的函数?

  Compilation Error

- 结论:通过对象句柄, 仅能调用该句柄类型的成员函数

程序解读（P418）

13.2 Relationships Among Objects in an Inheritance Hierarchy

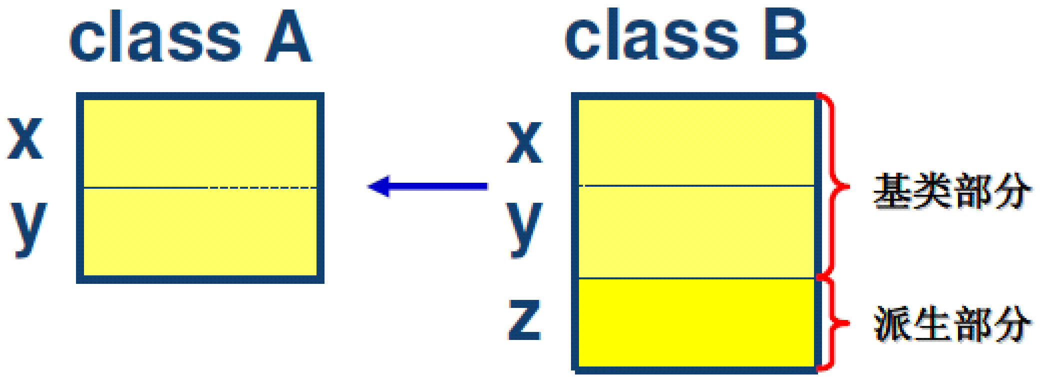
- 解决办法: downcasting
- If the address of a derived-class object (派生类对象地址) has been assigned to a pointer of one of its direct or indirect base classes (基类指针), it is acceptable to cast that base-class pointer back to a pointer of the derived-class type.

Downcasting

- ❑class B 继承 class A, 即B *is-a* A

```
1.  B big;                              正
2.  A small = big;                      确
3.  A &refSmall = big;  ——→  B &refBig = (B &) refSmall;
4.  A *pSmall = &big;   ——→  B *pBig = (B *) pSmall;
```

From A to B



Type Fields and switch Statements

- 如何知道vector中当前的基类指针需要downcast为哪种派生类指针?

13.2.4 Virtual Functions

- With virtual functions, the type of the object being pointed to, not the type of the handle, determines which version of a virtual function to invoke.

- 虚函数: 调用哪个(基类/派生类)虚函数, 由对象类型而不是句柄类型决定.

13.2.4 Virtual Functions ---语法

- 基 类

```
class Shape{
public:
    virtual void draw() const;
};
```

- 派生类

```
class Rectangle : public Shape{
    virtual void draw() const;
};
```

可省略. 只要基类声明函数为虚函数, 则所有派生类的该函数均为虚函数

13.2.4 Virtual Functions

- 虚函数用于继承结构中的基类和派生类, 以实现多态.
- 派生类中覆盖(**Overridden**)的虚函数和基类中的虚函数必须函数签名和返回值均相同.
- 函数定义时不需要**virtual**关键词.

```
1. Rectangle rect;
2. Shape *p = &rect;
3. p->draw();
```

13.2.4 Virtual Functions

- 调用虚函数的两种情况:
- 通过指向派生类的基类指针(或引用)调用, 程序会在执行时(execution time)根据对象类型动态选择合适的派生类函数– 动态绑定 (dynamic binding)或延迟绑定(late binding).

- 通过对象名和点操作符调用, 程序在编译时 (compile time)即根据对象类型确定函数– 静态绑定(static binding).

程序解读（P420）

13.2.4 Virtual Functions ---若干限制

- 只有类成员才能声明为虚函数
- 静态成员函数不能是虚函数
- 构造函数不能是虚函数
- 析构函数可以是虚函数

13.2.5 Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers

	base-class pointer	derived-class pointer
base-class object	OK	ERROR
derived-class object	OK	OK

```
class A{
public:
    void testfuc(){
        func( );
    }

    void func(){
        cout << "A::func called ";
        vfunc();
    }

    virtual void vfunc(){
        cout << "A::vfunc." << endl;
    }
};
```

```
class B : public A{
public:
    void func(){
        cout << "B::func nothing called." << endl;
    }
    virtual void vfunc(){
        cout << "B::vfunc." << endl;
    }
};
```

```
int main()
{
    A a;
    a.func();
    B b;
    b.func();
    b.testfuc();
```

```
    A *p = &b;
    p->vfunc();
    p->testfuc();
    p->func();

    return 0;
}
```

```
A::func called A::vFunc.
B::func nothing called.
A::func called B::vfunc.
B::vfunc.
A::func called B::vfunc.
A::func called B::vfunc.
```

Topics

- 13.1 Introduction
- 13.2 Relationships Among Objects in an Inheritance Hierarchy
- **13.3 Abstract Classes and Pure virtual Functions**
- 13.4 Case Study: Payroll System Using Polymorphism

13.3 Abstract Classes and Pure virtual Functions

```
class Shape{
public:
    virtual void draw() const;
};
```

- 一些成员函数对于基类来说是没有意义的, 将其声明为虚成员函数的目的是要求派生类给出其实现.

13.3 Abstract Classes and Pure virtual Functions

- ❑ Pure Virtual Function(纯虚函数)

  A pure virtual function is specified by placing "= 0" in its declaration, as in

  ```
  virtual void draw() const = 0;
  ```

- ❑ 对于纯虚函数, 不需要在类源码中给出其实现.

13.3 Abstract Classes and Pure virtual Functions

- Abstract Class(抽象类): 包含一个或多个纯虚函数的类. 无法实例化, 但可以声明指针和引用, 只能用于继承.
- 1. Shape obj;   // Error, 不能实例化
- 2. Rectangle objRectangle;
- 3. Shape *ptr = &objRectangle; // OK, 可指针
- 4. Shape &ref = objRectangle;   // OK, 可引用

- Concrete Class(具体类): 不包含纯虚函数, 可以实例化

13.3 Abstract Classes and Pure virtual Functions

- 成员函数是否声明为虚函数, 取决于是否需要多态性支持
- 虚函数是否声明为纯虚函数, 取决于该函数对于当前类是否有意义, 以及当前类是否需要实例化

基类	派生类
虚函数 has an implementation	gives the derived class the option of overriding the function
纯虚函数 does not provide an implementation	requires the derived class to override the function (for that derived class to be concrete; otherwise the derived class remains abstract)

Topics

13.4 Case Study: Payroll System Using Polymorphism

- 目的: 输出各类员工的基本信息和薪金信息
- Salaried employees (普通薪金制员工)

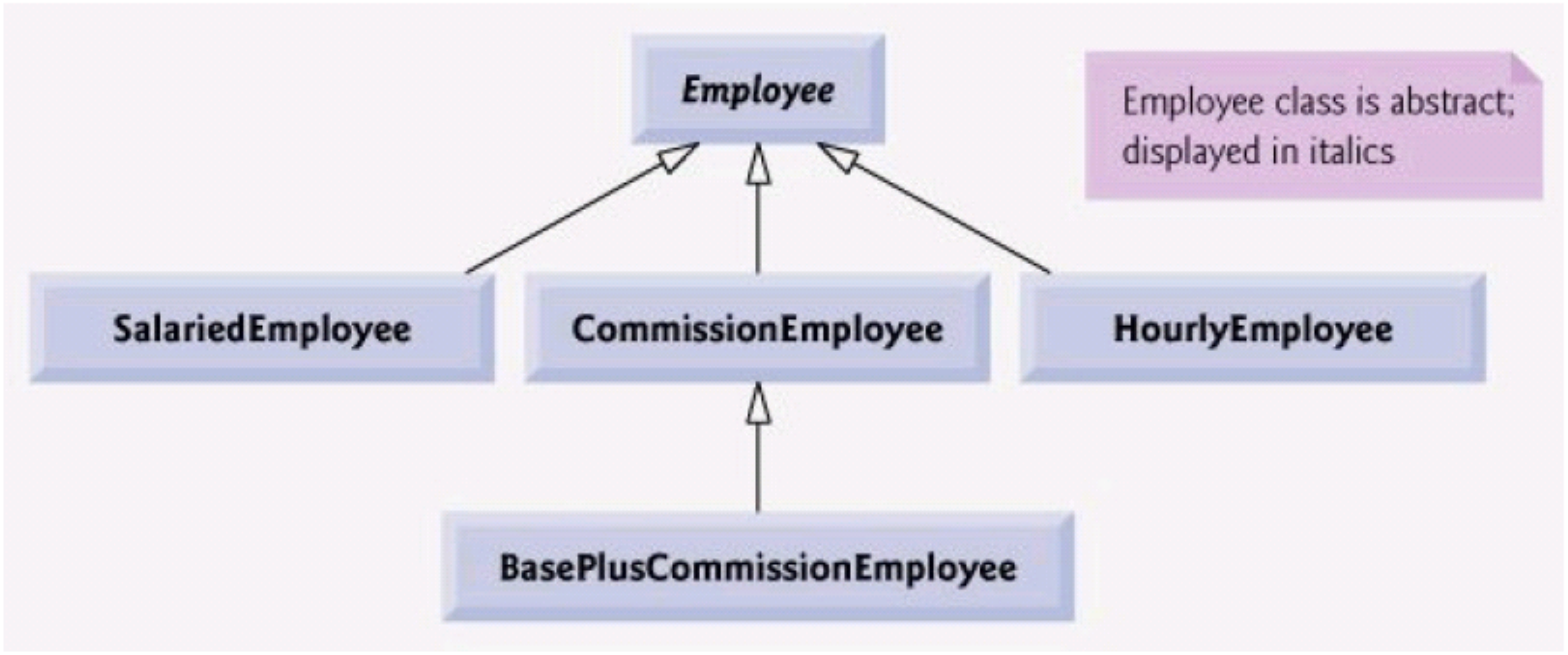
  Name, SSN, Weekly Salary
- Hourly employees (计时工)

  Name, SSN, Wage per hour, Hours
- Commission employees (佣金制员工)

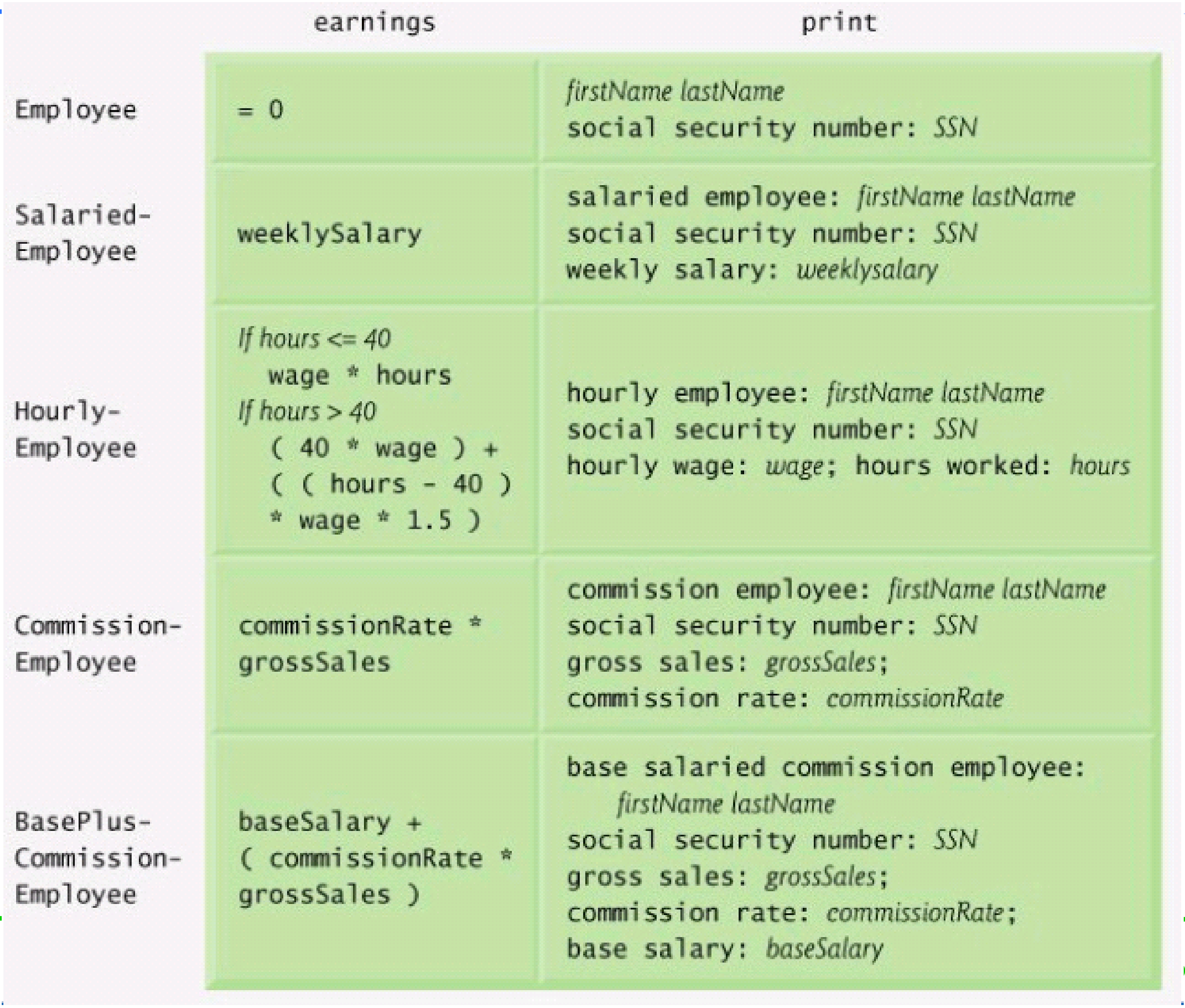
  Name, SSN, Gross sales amount, Commission rate
- Base-salary-plus-commission employees (带底薪的佣金制员工)

  Name, SSN, Gross sales amount, Commission rate, Base Salary

13.4 Case Study: Payroll System Using Polymorphism


Employee
Employee class is abstract; displayed in italics
SalariedEmployee
CommissionEmployee
HourlyEmployee
BasePlusCommissionEmployee

13.4.1 Creating Abstract Base Class Employee



	earnings	print
Employee	= 0	*firstName lastName* social security number: *SSN*
Salaried- Employee	weeklySalary	salaried employee: *firstName lastName* social security number: *SSN* weekly salary: *weeklysalary*
Hourly- Employee	*If hours <= 40* wage * hours *If hours > 40* (40 * wage) + ((hours - 40) * wage * 1.5)	hourly employee: *firstName lastName* social security number: *SSN* hourly wage: *wage*; hours worked: *hours*
Commission- Employee	commissionRate * grossSales	commission employee: *firstName lastName* social security number: *SSN* gross sales: *grossSales*; commission rate: *commissionRate*
BasePlus- Commission- Employee	baseSalary + (commissionRate * grossSales)	base salaried commission employee: *firstName lastName* social security number: *SSN* gross sales: *grossSales*; commission rate: *commissionRate*; base salary: *baseSalary*

13.4.1 Creating Abstract Base Class Employee

- Employee Class
- • Name, SSN: 各类员工的共有属性
- • print(): 输出Name, SSN等基本信息— 虚函数
- • earnings(): 没有意义, 要求派生类实现— 纯虚函数

程序解读（P427）

13.4.1 Creating Abstract Base Class SalariedEmployee

- SalariedEmployee, 继承Employee
- • weeklySalary: 普通薪金制员工的独有属性
- • print(): Override基类函数, 输出基本信息和薪酬信息
- • earnings(): 必须Override基类的纯虚函数, 计算薪酬

程序解读（P429）

13.4.1 Creating Abstract Base Class HourlyEmployee

- HourlyEmployee, 继承Employee
- • Wage, hours: 计时工的独有属性
- • print(): Override基类函数, 输出基本信息和薪酬信息
- • earnings(): 必须Override基类的纯虚函数, 计算薪酬

13.4.1 Creating Abstract Base Class CommissionEmployee

- CommissionEmployee, 继承Employee
- • grossSales, commisionRate: 佣金制员工的独有属性
- • print(): Override基类函数, 输出基本信息和酬信息
- • earnings(): 必须Override基类的纯虚函数, 计算薪酬

程序解读（P431）

13.4.1 Creating Abstract Base Class BasePlusCommissionEmployee

- BasePlusCommissionEmployee, 继承CommissionEmployee Class, 间接继承Employee Class
- • baseSalary: 带底薪的佣金制员工的独有属性
- • print(): Override基类CommissionEmployee 函数, 输出基本信息和薪酬信息
- • earnings(): 选择Override基类的虚函数, 计算薪酬

程序解读（P432）

13.4.6 Demonstrating Polymorphic Processing

- 实例化四种类型员工, 建立四个对象
- • 通过对象名调用print和earnings函数(静态绑定)
- • 通过基类指针调用print和earnings函数(动态绑定)
- • 通过基类引用调用print和earnings函数(动态绑定)

程序解读（P434）

Summary

- 虚函数和多态
- 静态绑定和动态绑定
- 纯虚函数
- 抽象类和具体类

Homework

- 实验必选题目：
  13.15，13.16
- 实验任选题目：

- 作业题目：