CS250 Intro to CS II

Spring 2019

Topics

- Virtual Functions
- Pure Virtual Functions
- Abstract Classes
- Concrete Classes
- Binding Time, Static Binding, Dynamic Binding
- Overriding vs Redefining

- Chapter 10

Abstract Class

- A class with a *pure virtual function*
- Add getMonthlyPay() to Employee
  - Only HourlyEmployee or MonthlyEmployee can implement this.
  - Must be able to call getMonthlyPay() from base class pointer

```cpp
// base class pointer
Employee *pcEmp = nullptr;

char choice;

cin>> choice;

switch(choice)
{
  case 'H':
    pcEmp = new HourlyEmployee();
    break;

  case 'S’:
    pcEmp = new SalariedEmployee();
    break;
}

cout << pcEmp->getMonthlyPay(); // ?????
```

Abstract Class

Pure Virtual Functions

- A class is made abstract by having one or more pure virtual functions associated with the class as follows:
  - `virtual void functionName () = 0;`
- Each derived class must provide its own draw function that overrides the draw function of the abstract class

Abstract Class Example

```
class Employee
{
  public:
    Employee (std::string name = "", std::string ssn = "");

    std::string getName () const;
    std::string getSSN () const;

    virtual void print (std::ostream &rcOutStream) const;
    virtual bool read(istream &rcIn);

    virtual double getMonthlyPay() const = 0;

  private:
    std::string mName,
                mSSN;
};
```

Concrete Class

- A concrete class is any class that can be instantiated
  - An object of a concrete class can be created

Of Employee, HourlyEmployee, and SalariedEmployee, which are abstract and which are concrete? Why?

Concrete Class Example

```
class SalariedEmployee : public Employee {

  public:

    SalariedEmployee();
    SalariedEmployee(string name, string ssn, double salary);

    virtual void print(ostream &rcOut) const;
    virtual bool read(istream &rcIn);

    virtual double getMonthlyPay() const;

  private:

    double mSalary;
};
```

```cpp
double SalariedEmployee::getMonthlyPay() const {
  return mSalary / 12;
}
```

Virtual Functions

- A virtual function
  - Allows the derived class the ability to override the function and
  - Must have an implementation
- A pure virtual function
  - Requires the derived class to override the function
  - *May* have an implementation in the base class

Binding Time

- Binding time - the time at which something becomes known
- Static Binding - binding time that happens during compilation (e.g. a variable’s type)
- Dynamic Binding - binding time that happens during runtime (e.g. the heap address of some dynamically allocated piece of memory)

Redefining vs Overriding

- A derived class can “redefine” a base class member (static binding)

- A derived class that redefines a virtual function of a base class is said to “override” the base class function (dynamic binding)

What happens?

```
ostream& operator<<(ostream &rcOut, const Employee &rcEmp)
{
  rcEmp.print(rcOut);
  return rcOut;
}
```

```
// base class pointer
Employee *pcEmp = nullptr;

char choice;


cin>> choice;
switch(choice)
{
  ...
}

pcEmp->read(cin);
cout << *pcEmp;
```