

Inheritance and Polymorphism

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

1 Inheritance
extending a clock to an alarm clock
deriving a class

2 Polymorphism
virtual functions and polymorphism
abstract classes

MCS 360 Lecture 8
Introduction to Data Structures
Jan Vershelde, 10 September 2010

Inheritance and Polymorphism

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism

abstract classes

1 Inheritance

extending a clock to an alarm clock

deriving a class

2 Polymorphism

virtual functions and polymorphism

abstract classes

an alarm clock

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Recall our class `Clock` as an encapsulation
of some of the functionality of `ctime`.

```
#include "clock.h"
#include <iostream>

using namespace std;

int main()
{
    Clock c;

    cout << "The current time is "
         << c << "." << endl;

    return 0;
}
```

the file `clock.h`

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

```
#ifndef CLOCK_H
#define CLOCK_H
```

```
#include <ostream>
#include <string>
```

```
class Clock
{
```

```
    public:
```

```
        Clock();
```

```
        int get_hours() const;
```

```
        int get_minutes() const;
```

```
        int get_seconds() const;
```

```
        friend std::ostream& operator<<
```

```
            (std::ostream& os, const Clock& c);
```

```
        bool operator==(const Clock& c) const;
```

clock.h continued

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism

abstract classes

```
private:

    int hours;    /* hours in 24 hour format
                  integer in the range 0..23 */
    int minutes; /* minutes in range 0..59 */
    int seconds; /* integers in range 0..61
                  allowing for leap second */
};

#endif
```

modeling an alarm clock

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism

abstract classes

An alarm clock is

- 1 a clock with the current time;
- 2 the fixed time of the alarm.

We reuse the existing code:

- 1 extend the definition of clock;
- 2 keep the class `Clock`.

The class `Alarm_Clock` inherits from `Clock`:

- 1 `Clock` is the base class.
- 2 `Alarm_Clock` is a derived class.

Inheritance and Polymorphism

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

1 Inheritance

extending a clock to an alarm clock
deriving a class

2 Polymorphism

virtual functions and polymorphism
abstract classes

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

deriving a class

the `alarm_clock.h` file

```
#ifndef ALARM_CLOCK_H
#define ALARM_CLOCK_H

#include "clock.h"
#include <ostream>
#include <string>

class Alarm_Clock : public Clock
{
```

⇒ `Alarm_Clock` is derived from `Clock`.

If `public` is omitted, then the members of `Clock` are not visible to clients of `Alarm_Clock`.

the constructor

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

```
class Alarm_Clock : public Clock
{
    public:

        Alarm_Clock(int h, int m, int s) : Clock(),
            hours(h), minutes(m), seconds(s) {}
        /* sets the alarm clock to the time h:m:s */
}
```

The constructor of the class `Alarm_Clock` is called and the arguments of `Alarm_Clock` are copied to the data attributes `hours`, `minutes`, and `seconds`.

data and functions

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

```
int get_alarm_hours() const;  
int get_alarm_minutes() const;  
int get_alarm_seconds() const;
```

```
private:
```

```
    int hours;    /* hours in 24 hour format  
                  integer in the range 0..23 */  
    int minutes; /* minutes in range 0..59 */  
    int seconds; /* integers in range 0..61  
                  allowing for leap second */  
};  
#endif
```

An object of the class `Alarm_Clock`
stores two triplets of integers.

Unified Modeling Language

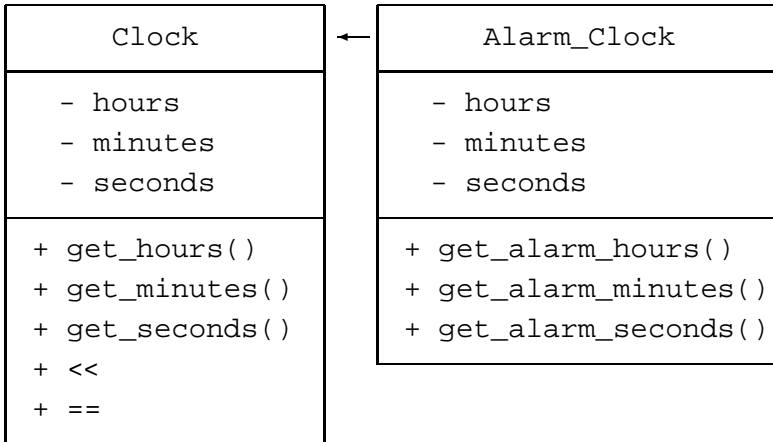
Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes



Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Because of the

```
class Alarm_Clock : public Clock
```

all clients of `Alarm_Clock` have access to the public members of `Clock`.

The data `hours`, `minutes`, `seconds` are *private* and therefore only accessible via the respective `get_hours()`, `get_minutes()`, `get_seconds()`.

To make members of a class also available to derived classes, replace `private` by `protected`.

testing Alarm_Clock

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

```
#include "clock.h"
#include "alarm_clock.h"
#include <iostream>
using namespace std;

int main()
{
    Clock c;

    cout << "The current time is "
         << c << "." << endl;

    Alarm_Clock a(21,11,8);
```

On a, we can apply public methods of Clock.

member function overloading

Inheritance

extending a clock to
an alarm clock

deriving a class

Polymorphism

virtual functions and
polymorphism

abstract classes

If our alarms occur mostly on the hour,
then we want to specify only the hour:

```
Alarm_Clock(int h) : Clock(), hours(h)
    { minutes = 0; seconds = 0; }
/* sets the alarm clock to the time h:0:0 */
```

This is an additional constructor in the class.

Multiple methods with the same name but different bodies is called member function overloading.

One application of overloading is to provide default values or modes of operation for mathematical functions.

Inheritance and Polymorphism

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

1 Inheritance

extending a clock to an alarm clock
deriving a class

2 Polymorphism

virtual functions and polymorphism
abstract classes

virtual functions

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Adding an operation to `clock.h`:

```
virtual std::string to_string() const;  
/* returns a string representation of a clock */
```

In `clock.cpp`:

```
std::string Clock::to_string() const  
{  
    using std::ostringstream;  
    ostringstream s;  
    s << std::setfill('0')  
      << std::setw(2) << hours << ":"  
      << std::setw(2) << minutes << ":"  
      << std::setw(2) << seconds;  
    return s.str();  
}
```

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Using `to_string` shortens the definition of `<<`.

In `clock.cpp`:

```
std::ostream& operator<<
    (std::ostream& os, const Clock& c)
{
    os << c.to_string();

    return os;
}
```

- Because `to_string()` is virtual in `Clock`, we can provide another definition in a derived class.
- An derived class inherits `<<`.
- In `<<`, the proper definition of `to_string()` is called, depending on the operand of `<<`.

overriding functions

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Also in Alarm_Clock we have to_string():

```
std::string Alarm_Clock::to_string() const
{
    using std::ostringstream;
    ostringstream s;

    s << std::setfill('0')
      << std::setw(2) << hours << ":"
      << std::setw(2) << minutes << ":"
      << std::setw(2) << seconds
      << " set at " << Clock::to_string();

    return s.str();
}
```

We use to_string() of Clock() when we define to_string() of Alarm_Clock().

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

```
int main()
{
    Clock c;

    cout << "The current time is "
          << c << "." << endl;

    Alarm_Clock a(21,11,8);

    cout << "The alarm is "
          << a << "." << endl;
}
```

Note that << is defined only once!

more polymorphism

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

We first define a pointer to the base class:

```
Clock *d;  
  
d = new Alarm_Clock(12,50,0);  
  
cout << "to_string on Clock : "  
      << d->Clock::to_string() << endl;  
cout << "to_string on Alarm_Clock : "  
      << d->to_string() << endl;
```

The pointer `d` refers to an alarm clock after the `new`.
Different definitions of `to_string()` apply.

dynamic casting

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

After declaring

```
Clock *e;
```

we can change the type of the pointer:

```
Alarm_Clock *f;
```

```
f = dynamic_cast<Alarm_Clock*>(e);
```

Now we can process `f` through
the methods of the class `Alarm_Clock`.

a virtual destructor

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

A last addition to `clock.h`:

```
virtual ~Clock() {}  
/* needed for delete in derived classes */
```

The code defined in the destructor is executed when the `delete` method is called.

The `delete` deallocates memory allocated for objects declared as pointers and constructed via `new`.

Inheritance and Polymorphism

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

1 Inheritance

extending a clock to an alarm clock
deriving a class

2 Polymorphism

virtual functions and polymorphism
abstract classes

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

An abstract class

- 1 cannot be instantiated: no creation of objects,
- 2 must have a least one virtual function.

An abstract function is a virtual function without body.

Some applications:

- ordering types in a hierarchy
- with an abstract ring of operations
we can define algorithms (e.g.: GCD) for any ring

the food pyramid

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Every food has an amount of calories:

- 1 the data attribute is a double, kept private,
- 2 a public function returns its value.

Substances that provide energy in food are carbohydrates, proteins, fats.

Of interest is a function to return the percentage of each of these substances in a food object.

The `const = 0` at the end of

```
virtual double pct_fat() const = 0;
```

must be there for a *pure* virtual function in an abstract class.

the class Food

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

```
#ifndef FOOD_H
#define FOOD_H
```

```
class Food
{
    private:

        double calories;

    public:

        double get_calories() const
            { return calories; }

        virtual double pct_carbohydrates() const = 0;
        virtual double pct_protein() const = 0;
        virtual double pct_fat() const = 0;
};
#endif
```

derived classes

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

On `MyPyramid.gov`, there are 6 groups:
Grain, Vegetable, Fruit, Milk, Meat & Beans, Oils.

Fruits are important sources of potassium, dietary fiber, and vitamin C. As data attributes for the abstract class `Fruit` we could store the amount of these nutrients.

The `Fruit` class still breaks up into Juices and Solids.

Summary + Assignments

Inheritance

extending a clock to
an alarm clock
deriving a class

Polymorphism

virtual functions and
polymorphism
abstract classes

Started Chapter 3: *Inheritance and Class Hierarchies*.

A virtual `to_string()` is very convenient for `<<`.

Assignments:

- 1 Define a class `Bank_Account` to hold the balance of a bank account. The balance is a private float. The initial balance is given to the constructor. Write public functions to withdraw and deposit. A `to_string()` method returns a string to format the balance with 2 places after the decimal point. Use `to_string()` in `<<`. Test the class with a small program.
- 2 Derive from `Bank_Account` a class `Savings_Account`. An extra data attribute is the interest rate, set at the creation of the account. Write a function to update the balance with the yield from the interest. Override the `to_string()` method to display the interest rate along with the balance. Show that by the virtual `to_string()` in `Bank_Account`, there is no need to define `<<` for a savings account.