Destructors, Get and Set, and Default Memberwise Assignment
Destructors (7.16)

- The opposite of constructors
- Have the same name as the class, with a ~ in front of it
- Called whenever an object is destroyed
- A destructor has no arguments and no return value
- Only one destructor allowed!
- No need for us to explicitly declare a destructor
Example

class Test
{
    private:
        int id;
    public:
        Test(int);
        ~Test();
};

Test::Test(int i)
{
    id = i;
    cout << "constructor for " << id << " is called\n";
}

Test::~Test()
{
    cout << "destructor for " << id << " is called\n";
}
What is the Output?

```c
void funct();

int main()
{
    Test cTest1(1);
    funct();
    Test cTest3(3);

    return 0;
}

void funct()
{
    Test cTest2(2);
}
```
Set and Get Functions

• The principle of least privilege says that we should only provide outside members with access to data that is absolutely necessary.

• Data members should therefore be set to private.

• To modify and get access to that data, specific member functions need to be provided.

• These are the Set and Get functions.
Set and Get Functions

• The functions don’t need to be called set or get, but it has become commonplace to do this.

• In the time class we could have the following set functions:
  
  ◦ void setTime(int, int, int);
  ◦ void setHour(int);
  ◦ void setMinute(int);
  ◦ void setSecond(int);