Dynamic Memory with Classes

ITP 165 – Fall 2015

Week 11, Lecture 2

Reviewing the Clock class (Clock.h)

```
#pragma once

// Represents a clock in military time
class Clock {
private:
   int mHours;
   int mMinutes;
   int mSeconds;
public:
   // Default constructor (sets to midnight)
   Clock();
   // Resets clock to midnight
   void reset();
   // Prints the clock H:M:S
   void print();
   // Advances clock by one second
   void tick();
   // Get/set hours
   int getHours();
   void setHours(int newHours);
   // Returns true if this object has the same
   // time as the other object.
   bool isEqual(Clock& other);
};
```

Reviewing the Clock class (Clock.cpp)

```cpp
#include "Clock.h"
#include <iostream>

// Default constructor
Clock::Clock() {
   reset();
}

// Resets clock to midnight
void Clock::reset() {
   mHours = 0;
   mMinutes = 0;
   mSeconds = 0;
}

// Prints the clock H:M:S
void Clock::print() {
   std::cout << mHours << ":";
   std::cout << mMinutes << ":";
   std::cout << mSeconds << std::endl;
}

// Gets hours
int Clock::getHours() {
   return mHours;
}

// Sets hours
void Clock::setHours(int newHours) {
   if (newHours >= 0 && newHours <= 23) {
      mHours = newHours;
   }
}
```

```cpp
// Advances clock by one second
void Clock::tick() {
   mSeconds++;
   if (mSeconds == 60) {
      mSeconds = 0;
      mMinutes++;
      if (mMinutes == 60) {
         mMinutes = 0;
         mHours++;
         if (mHours == 24) {
            mHours = 0;
         }
      }
   }
}

// Returns true if this object has the same
// time as the other object.
bool Clock::isEqual(Clock& other) {
   if (mHours == other.mHours &&
      mMinutes == other.mMinutes &&
      mSeconds == other.mSeconds) {
      return true;
   }
   else {
      return false;
   }
}
```

Reviewing the Clock class (main.cpp)

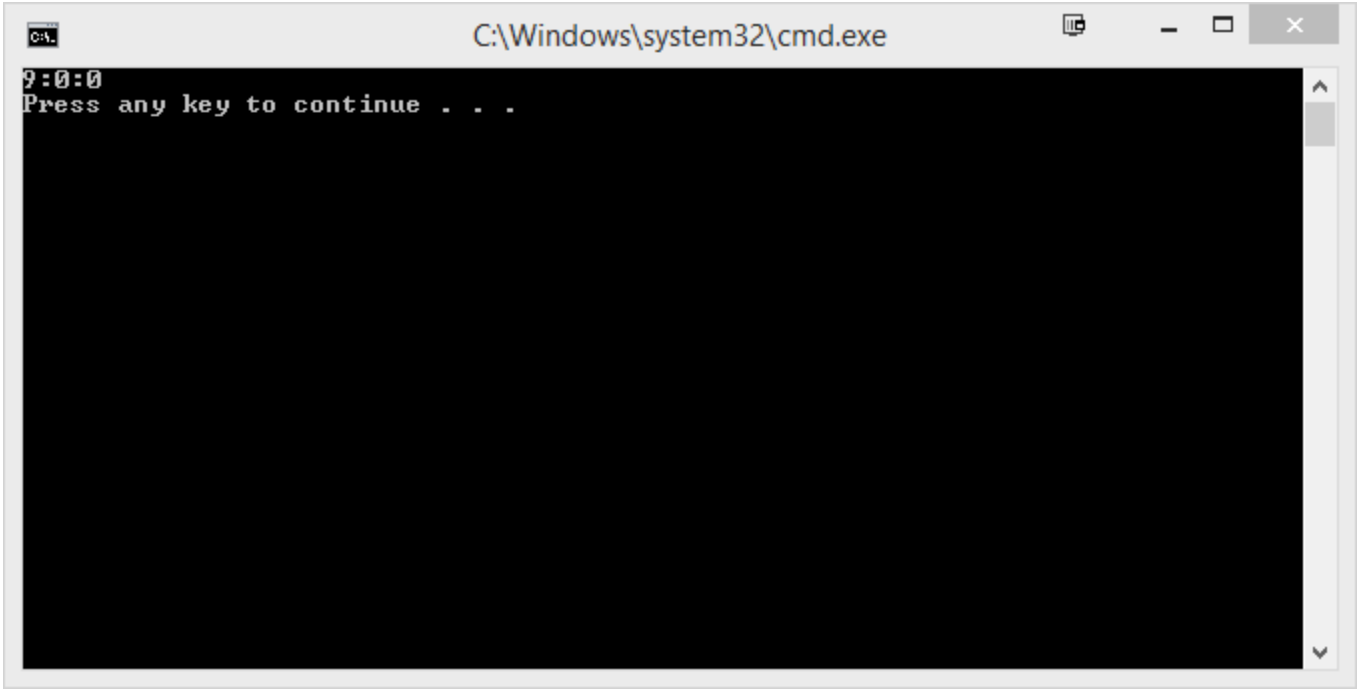
```
#include "Clock.h"

int main() {
    Clock myClock;
    myClock.setHours(9);

    myClock.print();

    return 0;
}
```

Clock in action



What about a Clock pointer?

```
#include "Clock.h"

int main()
{
    Clock myClock;
    Clock* clockPtr = &myClock;

    // What would be the equivalent code
    // with clockPtr?
    //myClock.setHours(9);
    //myClock.print();

    return 0;
}
```

Working with class pointers

- Since `clockPtr` is a pointer, we have to dereference it before we can access any member function

- To do that we would write:

```
(*clockPtr).setHours(9);
(*clockPtr).print();
```

- Note: C++ requires the parenthesis because of order-of-operations

```
(*clockPtr).setHours(9) != *clockPtr.setHours(9)
```

Class pointers and the arrow operator

- C++ provides a shortcut to “dereference and access the member”

- Using explicit dereference and call member function:

```
(*clockPtr).setHours(9);
```

- Using shortcut to dereference and call member function:

```
clockPtr->setHours(9);
```

Using a clock pointer

```
#include "Clock.h"

int main()
{
    Clock myClock;
    Clock* clockPtr = &myClock;

    clockPtr->setHours(9);
    clockPtr->print();

    return 0;
}
```

Putting a clock on the heap

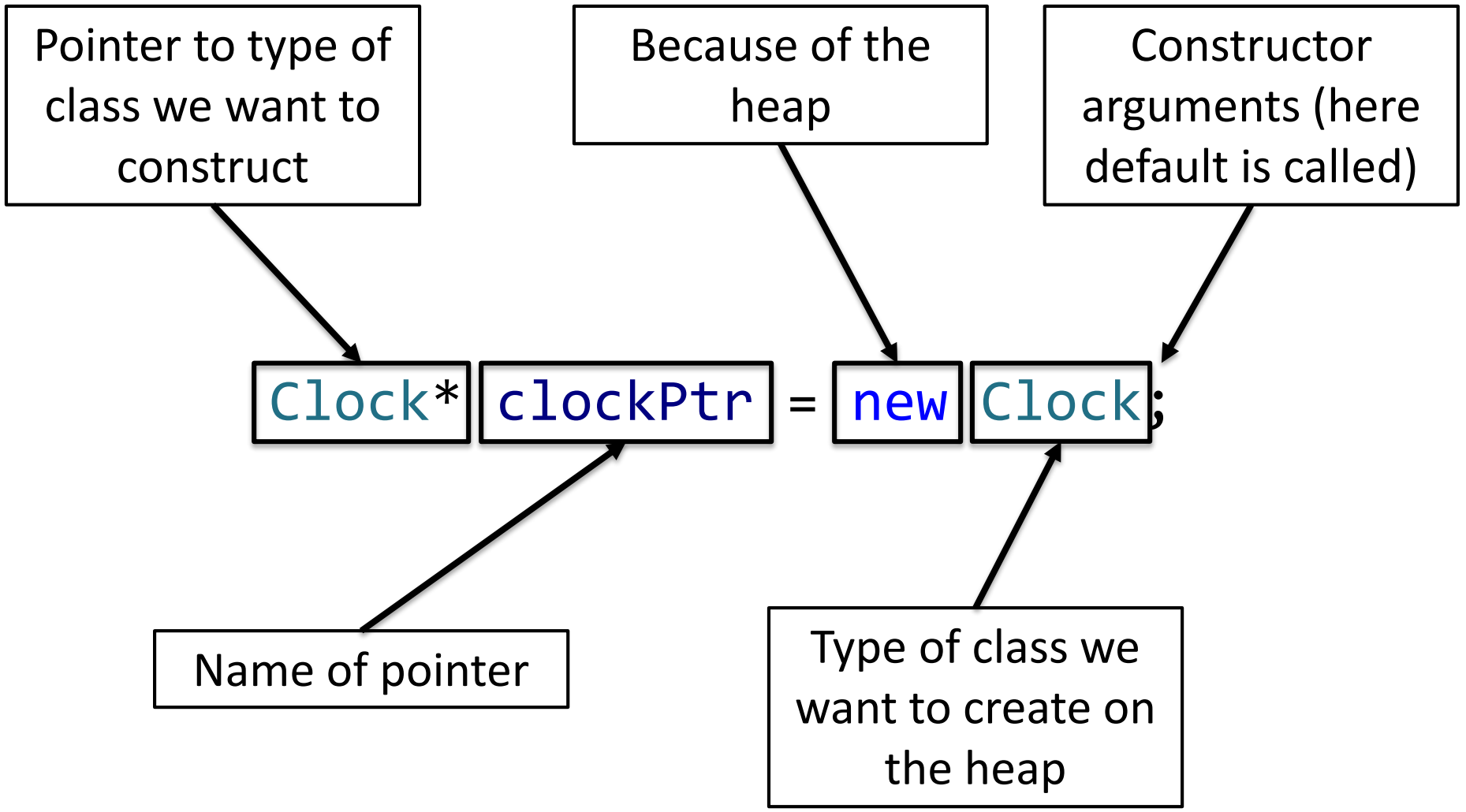
```
#include "Clock.h"

int main()
{
    Clock* clockPtr = new Clock;

    clockPtr->setHours(9);
    clockPtr->print();

    return 0;
}
```

Creating a class variable on the heap



Lab Practical #19