



Objectives

In this Lecture, you will:

- Learn about classes
- Learn about `private`, `protected`, and `public` members of a class
- Explore how classes are implemented
- Become aware of accessor and mutator functions
- Examine constructors and destructors

Objectives (cont'd.)

- Learn about the abstract data type (ADT)
- Explore how classes are used to implement ADTs
- Become aware of the differences between a `struct` and a `class`
- Learn about information hiding
- Explore how information hiding is implemented in C++
- Learn about the `static` members of a class

Classes

- Object-oriented design (OOD): a problem solving methodology
- Objects: components of a solution
- Class: a collection of a fixed number of components
- Member: a component of a class

Classes (cont'd.)

- Class definition:
 - Defines a data type; no memory is allocated
 - Don't forget the semicolon after the closing brace
- Syntax:

```
class classIdentifier
{
    classMembersList
};
```

Classes (cont'd.)

- Class member can be a variable or a function
- If a member of a `class` is a variable
 - It is declared like any other variable
 - You cannot initialize a variable when you declare it
- If a member of a `class` is a function
 - Function prototype is listed
 - Function members can (directly) access any member of the `class`

Classes (cont'd.)

- Three categories of class members:
 - `private` (default)
 - Member cannot be accessed outside the `class`
 - `public`
 - Member is accessible outside the class
 - `protected`

Unified Modeling Language Class Diagrams

- Unified Modeling Language (UML) notation:
used to graphically describe a class and its members
 - `+`: member is public
 - `-`: member is private
 - `#`: member is protected

Unified Modeling Language Class Diagrams (cont'd.)

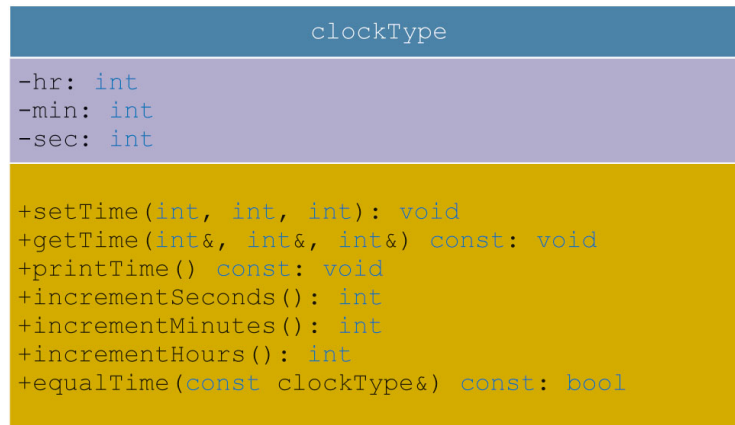


FIGURE 10-1 UML class diagram of the `clockType` class

Variable (Object) Declaration

- Once defined, you can declare variables of that `class` type

```
clockType myClock;
```

- A `class` variable is called a class object or class instance

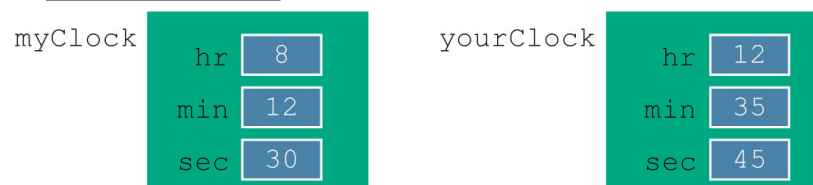


FIGURE 10-2 Objects `myClock` and `yourClock`

Accessing Class Members

- Once an object is declared, it can access the `public` members of the class
- Syntax:

`classObjectName.memberName`

 - The dot (.) is the **member access operator**
- If an object is declared in the definition of a member function of the `class`, it can access the `public` and `private` members

Built-in Operations on Classes

- Most of C++'s built-in operations do not apply to classes
 - Arithmetic operators cannot be used on class objects unless the operators are overloaded
 - Cannot use relational operators to compare two class objects for equality
- Built-in operations that are valid for class objects:
 - Member access (.)
 - Assignment (=)

Assignment Operator and Classes

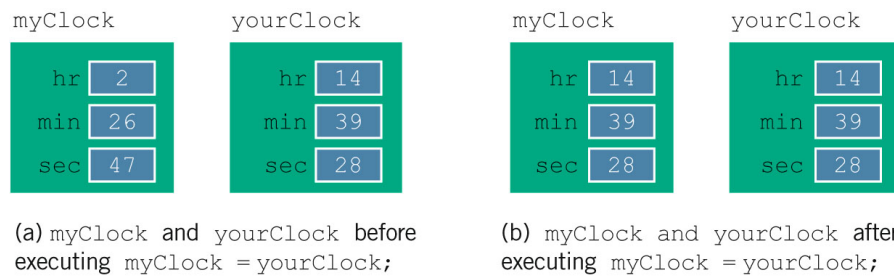


FIGURE 10-3 `myClock` and `yourClock` before and after executing the statement `myClock = yourClock;`

Class Scope

- An object can be automatic or static
 - Automatic: created when the declaration is reached and destroyed when the surrounding block is exited
 - Static: created when the declaration is reached and destroyed when the program terminates
- Object has the same scope as other variables

Class Scope (cont'd.)

- A member of the `class` is local to the `class`
- Can access a `class` member outside the `class` by using the `class` object name and the member access operator (`.`)

Functions and Classes

- Objects can be passed as parameters to functions and returned as function values
- As parameters to functions
 - Objects can be passed by value or by reference
- If an object is passed by value
 - Contents of data members of the actual parameter are copied into the corresponding data members of the formal parameter

Reference Parameters and Class Objects (Variables)

- Passing by value might require a large amount of storage space and a considerable amount of computer time to copy the value of the actual parameter into the formal parameter
- If a variable is passed by reference
 - The formal parameter receives only the address of the actual parameter

Computer Programming II

17

Reference Parameters and Class Objects (Variables) (cont'd.)

- Pass by reference is an efficient way to pass a variable as a parameter
 - Problem: when passing by reference, the actual parameter changes when formal parameter changes
 - Solution: use `const` in the formal parameter declaration

Computer Programming II

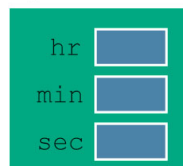
18

Implementation of Member Functions

- Must write the code for functions defined as function prototypes
- Prototypes are left in the class to keep the class smaller and to hide the implementation
- To access identifiers local to the class, use the scope resolution operator ::

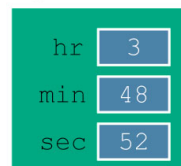
Implementation of Member Functions (cont'd.)

myClock



(a) myClock before executing
`myClock.setTime(3, 48, 52);`

myClock



(b) myClock after executing
`myClock.setTime(3, 48, 52);`

FIGURE 10-4 myClock before and after executing the statement `myClock.setTime(3, 48, 52);`

Implementation of Member Functions (cont'd.)

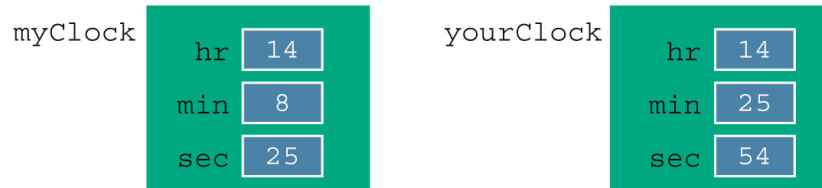


FIGURE 10-5 Objects `myClock` and `yourClock`

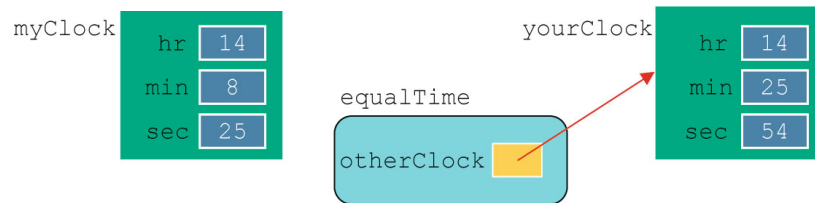


FIGURE 10-6 Object `myClock` and parameter `otherClock`

Implementation of Member Functions (cont'd.)

- Once a `class` is properly defined and implemented, it can be used in a program
 - A program that uses/manipulates objects of a class is called a client of that class
- When you declare objects of the `class` `clockType`, each object has its own copy of the member variables (`hr`, `min`, and `sec`)
 - Called instance variables of the class
 - Every object has its own instance of the data

Example

```
MyClass
- fullName: string
- age: int
- Saving: int
+ increaseSpeed(int): void
```

Accessor and Mutator Functions

- Accessor function: member function that only accesses the value(s) of member variable(s)
- Mutator function: member function that modifies the value(s) of member variable(s)
- Constant function:
 - Member function that cannot modify member variables
 - Use `const` in function heading

Order of `public` and `private` Members of a Class

- C++ has no fixed order in which to declare `public` and `private` members
- By default, all members of a class are `private`
- Use the member access specifier `public` to make a member available for `public` access

Constructors

- Use constructors to guarantee that member variables of a class are initialized
- Two types of constructors:
 - With parameters
 - Without parameters (default constructor)
 - Name of a constructor = name of the class
 - A constructor has no type

Constructors (cont'd.)

- A class can have more than one constructor
 - Each must have a different formal parameter list
- Constructors execute automatically when a class object enters its scope
- They cannot be called like other functions
- Which constructor executes depends on the types of values passed to the class object when the class object is declared

Example 2

```
MyClass
- fullName: string
- age: int
- Saving: int

+ increaseSpeed(int): void
+ Mycalss()
+ Mycalss(string name)
```

Invoking a Constructor

- A constructor is automatically executed when a class variable is declared
- Because a class may have more than one constructor, you can invoke a specific constructor

Invoking the Default Constructor

- To invoke the default constructor:

```
className classObjectName;
```

- Example:

```
clockType yourClock;
```

Invoking a Constructor with Parameters

- Syntax:

```
className classObjectName(argument1, argument2, ...);
```

- Number and type of arguments should match the formal parameters (in the order given) of one of the constructors
 - Otherwise, C++ uses type conversion and looks for the best match
 - Any ambiguity causes a compile-time error

Constructors and Default Parameters

- A constructor can have default parameters
 - Rules for declaring formal parameters are the same as for declaring default formal parameters in a function
 - Actual parameters are passed according to same rules for functions
- Default constructor: a constructor with no parameters or with all default parameters

Classes and Constructors: A Precaution

- If a class has no constructor(s), C++ provides the default constructor
 - However, object declared is still uninitialized
- If a class includes constructor(s) with parameter(s), but not the default constructor
 - C++ does not provide the default constructor

Arrays of Class Objects (Variables) and Constructors

- If you declare an array of class objects, the class should have the default constructor

```
clockType arrivalTimeEmp[100];
```

```
arrivalTimeEmp
```

```
arrivalTimeEmp[0]
```

```
arrivalTimeEmp[1]
```

```
arrivalTimeEmp[49]
```

```
arrivalTimeEmp[98]
```

```
arrivalTimeEmp[99]
```

```
arrivalTimeEmp[49]
```

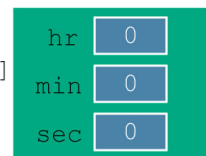


FIGURE 10-8 Array arrivalTimeEmp

Destructors

- Destructors are functions without any type
- The name of a destructor is the character '~' followed by class name
 - For example:


```
~clockType();
```
- A class can have only one destructor
 - The destructor has no parameters
- Destructor automatically executes when the class object goes out of scope

Lab Exercise 1

- Write the implementation of the following class diagram, and implement all the member functions.
- Create a function *display* that takes a student object and display its information using accessor functions.
- In the *main* function, create two objects of the class, and use the mutator functions to initialize the member variable for each object.
- Call the display method for each object to show the information for each student

Student	
+ firstName: string	
+ lastName: string	
+ level: int	
+ CGPA: float	
+setFirstName(string fName): void	
+setLastName(string Lname): void	
+setLevel(int lev): void	
+setCGPA(float): void	
+getFirstName(): string	
+getLastName(): string	
+getLevel(): int	
+getCGPA(): float	

A struct Versus a class

- By default, members of a `struct` are `public`
 - `private` specifier can be used in a `struct` to make a member private
- By default, the members of a `class` are `private`
- `classes` and `structs` have the same capabilities

Computer Programming II

37

A struct Versus a class (cont'd.)

- In C++, the definition of a `struct` was expanded to include member functions, constructors, and destructors
- If all member variables of a `class` are `public` and there are no member functions
 - Use a `struct`

Computer Programming II

38

Information Hiding

- Information hiding: hiding the details of the operations on the data
- Interface (header) file: contains the specification details
 - File extension is `.h`
- Implementation file: contains the implementation details
 - File extension is `.cpp`
- In header file, include function prototypes and comments that briefly describe the functions
 - Specify preconditions and/or postconditions

Information Hiding (cont'd.)

- Implementation file must include header file via `include` statement
- In `include` statement:
 - User-defined header files are enclosed in double quotes
 - System-provided header files are enclosed between angular brackets

Information Hiding (cont'd.)

- Precondition: A statement specifying the condition(s) that must be true before the function is called
- Postcondition: A statement specifying what is true after the function call is completed

Executable Code

- To use an object in a program
 - The program must be able to access the implementation
- Visual C++ 2012 Express, Visual Studio 2012, and C++ Builder put the editor, compiler, and linker into a package
 - One command (build, rebuild, or make) compiles program and links it with the other necessary files
 - These systems also manage multiple file programs in the form of a project

Static Members of a Class

- Use the keyword `static` to declare a function or variable of a class as `static`
- A `public static` function or member of a class can be accessed using the class name and the scope resolution operator
- `static` member variables of a class exist even if no object of that `class` type exists

Static Members of a Class (cont'd.)

- Multiple objects of a class each have their own copy of non-static member variables
- All objects of a class share any static member of the class

Summary

- Class: collection of a fixed number of components
- Members: components of a class
 - Accessed by name
 - Classified into one of three categories:
 - `private`, `protected`, and `public`
- Class variables are called class objects or, simply, objects

Summary (cont'd.)

- The only built-in operations on classes are assignment and member selection
- Constructors guarantee that data members are initialized when an object is declared
 - Default constructor has no parameters
- Destructor automatically executes when a class object goes out of scope
 - A class can have only one destructor
 - The destructor has no parameters

Summary (cont'd.)

- Abstract data type (ADT): data type that separates the logical properties from the implementation details
- A `public static` member, function or data, of a class can be accessed using the class name and the scope resolution operator
- Static data members of a class exist even when no object of the class type exists
- Instance variables: non-static data members