Capitulo 9

This activity contains 24 questions.

1.

Section 9.2 Time Class Case Study

9.2 Q1: Member access specifiers (public and private) can appear:

- ○ *a.d. Outside a class definition.*
- ○ *a.c. In any order and multiple times, if they have brackets separating each type.*
- ○ *In any order and multiple times.*
- ○ *a.b. In any order (public first or private first) but not multiple times.*

2.

9.2 Q2: Which of the following preprocessor directives does not constitute part of the preprocessor wrapper?

- ○ *#define.*
- ○ *#ifndef.*
- ○ *#endif.*
- ○ *#include.*

3.

9.2 Q3: Member function definitions:

- ○ *Always require the binary scope operator (::).*
- ○ *Must use the binary scope operator in their function prototype.*
- ○ *Can use the binary scope operator anywhere, but become public functions.*
- ○ *Require the binary scope operator only when being defined outside of the definition of their class.*

4.

9.2 Q4: Parameterized stream manipulator setfill specifies the fill character that is displayed when an output is displayed in a field wider than the number of characters or digits in the output. The effect of setfill applies:

- ○ *Until explicitly set to a different setting.*
- ○ *Only to the current value being displayed.*
- ○ *Only to outputs displayed in the current statement.*
- ○ *Until the output buffer is flushed.*

5. 9.2 Q5: Every object of the same class:

- ◯ Gets a copy of every member function.
- ◯ Gets a copy of every member function and member variable.
- ◯ Gets a copy of every member variable.
- ◯ Shares pointers to all member variables and member functions.

6. 9.2 Q6: Classes cannot:

- ◯ Be derived from other classes.
- ◯ Be used to model attributes and behaviors of objects.
- ◯ a.d. Include objects from other classes as members.
- ◯ Initialize data members in the class definition.

7. Section 9.3 Class Scope and Accessing Class Members

9.3 Q1: Variables defined inside a member function of a class have:

- ◯ Class or block scope, depending on whether the binary scope resolution operator (::) is used.
- ◯ File scope.
- ◯ Block scope.
- ◯ Class scope.

8. 9.3 Q2: A class-scope variable hidden by a block-scope variable can be accessed by preceding the variable name with the class name followed by:

- ◯ ->.
- ◯ ::.
- ◯ ..
- ◯ :.

9. Section 9.4 Separating Implementation from Interface
,br> 9.4 Q1: When independent software vendors provide class libraries to clients, they typically give the __________ for the class's

interface and the __________ for the class's implementation.

- ○ *Object module file, source code file.*
- ○ *Source code file, source code file.*
- ○ *Object module file, object module file.*
- ○ *Source code file, object module file.*

10. 9.4 Q2: Which of the following is not true about separating a class's interface and implementation?

- ○ *Private data members are included in the header file.*
- ○ *Inline member function definitions are included in the header file.*
- ○ *Changes in the class's implementation will affect the client.*
- ○ *Changes in the class's interface will affect the client.*

11. Section 9.5 Access Functions and Utility Functions

9.5 Q1: The type of function a client would use to check the balance of his bank account would be:

- ○ *A predicate function.*
- ○ *A utility function.*
- ○ *An access function.*
- ○ *A constructor.*

12. 9.5 Q2: Utility functions:

- ○ *Are intended to be used by clients of a class.*
- ○ *Are part of a class's interface.*
- ○ *Are separate member functions that support operations of the class's other member functions.*
- ○ *Are a type of constructor.*

13. Section 9.6 Time Class Case Study: Constructors with Default Arguments

9.6 Q1: A default constructor:

- ○ *Does not perform any initialization.*

- ○ *Both (a) and (b).*
- ○ *Is the constructor generated by the compiler when no constructor is provided by the programmer.*
- ○ *Is a constructor with all default arguments.*

14. *9.6 Q2: If a member function of a class already provides all or part of the functionality required by a constructor or another member function then:*

- ○ *Call that member function from this constructor or member function.*
- ○ *That member function is unnecessary.*
- ○ *This constructor or member function is unnecessary.*
- ○ *Copy and paste that member function's code into this constructor or member function.*

15. *Section 9.7 Destructors*

9.7 Q1: Which of the following is not true of a constructor and destructor of the same class?

- ○ *They are both usually called once per object created.*
- ○ *They both have the same name aside from the tilde (~) character.*
- ○ *Both are called automatically, even if they are not explicitly defined in the class.*
- ○ *They both are able to have default arguments.*

16. *9.7 Q2: Which of the following is not true of a destructor?*

- ○ *It performs termination housekeeping.*
- ○ *If the programmer does not explicitly provide a destructor, the compiler creates an "empty" destructor.*
- ○ *It is called before the system reclaims the object's memory.*
- ○ *It releases the object's memory.*

17. *Section 9.8 When Constructors and Destructors Are Called*

9.8 Q1: Given the class definition:

```
class CreateDestroy
{
public:
   CreateDestroy() { cout << "constructor called, "; }
   ~CreateDestroy() { cout << "destructor called, "; }
};
```

What will the following program output?

```
int main()
{
   CreateDestroy c1;
   CreateDestroy c2;
   return 0;
}
```

- ○ *constructor called, destructor called, .*
- ○ *constructor called, constructor called, destructor called, destructor called, .*
- ○ *constructor called, destructor called, constructor called, destructor called, .*
- ○ *constructor called, constructor called, .*

18. *9.8 Q2: Given the class definition:*

```
class CreateDestroy
{
public:
   CreateDestroy() { cout << "constructor called, "; }
   ~CreateDestroy() { cout << "destructor called, "; }
};
```

What will the following program output?

```
int main()
{
   for ( int i = 1; i <= 2; i++ )
      CreateDestroy cd;
   return 0;
}
```

- ○ *constructor called, constructor called, destructor called, destructor called, .*
- ○ *constructor called, destructor called, constructor called, destructor called, .*
- ○ *Nothing.*
- ○ *constructor called, constructor called, .*

19.

Section 9.9 Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member

9.9 Q1: Returning references to non-const, private data:

- ○ *Allows private functions to be modified.*
- ○ *Allows private member variables to be modified, thus "breaking encapsulation."*
- ○ *Results in a compiler error.*
- ○ *Is only dangerous if the binary scope resolution operator (::) is used in the function prototype.*

20.

9.9 Q2: A client changing the values of private data members is:

- ○ *Possible using public functions and references.*
- ○ *Never possible.*
- ○ *Only possible by calling private member functions*
- ○ *Only possible if the private variables are not declared inside the class.*

21.

Section 9.10 Default Memberwise Assignment

9.10 Q1: The assignment operator (=) can be used to:

- ○ *Copy data from one object to another.*
- ○ *Test for equality.*
- ○ *Copy a class.*
- ○ *Compare two objects.*

22.

Section 9.11 Software Reusability

9.11 Q1: Many ___________ exist which help to develop programs from portable, carefully tested and widely available components.

- ○ *Object libraries.*
- ○ *Structured program environments.*
- ○ *Driver files.*
- ○ *Class libraries.*

23. *Section 9.12 (Optional) Software Engineering Case Study: Starting to Program the Classes of the ATM System*

9.12 Q1: Associations in a class diagram that have no navigability arrows at all indicate:

- ○ *That the two classes are the same.*
- ○ *That navigation can proceed in either direction across the association.*
- ○ *That operations performed by this association do not return values.*
- ○ *Inheritance from the same base class.*

24. *9.12 Q2: Which of the following is not true about declaring references to objects of other classes inside a class definition?*

- ○ *If the class names for the other objects are used only to declare these references, a forward declaration can replace the #include statement usually used to include those classes' header files.*
- ○ *Each reference only requires enough memory to store the memory address of the object it references.*
- ○ *These references can represent directional associations from a UML class diagram.*
- ○ *These references can be initialized inside the class definition.*

Clear Answers / Start Over   Submit Answers for Grading

Answer choices in this exercise appear in a different order each time the page is loaded.