12 Inheritance and Aggregation

REVIEW QUESTIONS

1. Constructors are inherited by derived classes.

b. false

3. In private inheritance, the private members of the base class become ____ in the derived class.

a. inaccessible

5. In private inheritance, the public members of the base class become ____ in the derived class.

b. private

7. In protected inheritance, the protected members of the base class become ____ in the derived class.

c. protected

9. In public inheritance, the private members of the base class become ____ in the derived class.

a. inaccessible

11. In public inheritance, the public members of the base class become ____ in the derived class.

d. public

13. What is the difference between a virtual function and a pure virtual function?

A virtual function needs to have definition; a pure virtual function has only a declaration.

15. Can a constructor be a virtual function? Explain why or why not.

No we cannot have virtual constructor because the name of the constructor in the base and derived cannot be the same.

17. Can an abstract class have a constructor or destructor although no object is created from an abstract class? Explain why or why not

Yes. It must have both constructor and destructor because an abstract class must be inherited and the inherited class constructor and destructor must call the parent constructor and destructor.

19. Can a class inherit from two abstract class? Explain why or why not.

Yes. All pure functions in both classes must be defined in the inherited class.

21. Can we assign an object of a base class to an object of a derived class? Explain why or why not.

No it is not allowed because an object of the derived class has more data item.

23. If you want to achieve polymorphism, you need to pass an object to a function by ____ or ____.

pointer or reference

25. In dynamic binding, the association between a pointer and a function is determined during ____.

run time

27. A derived class constructor (or copy constructor) must call the base class constructor in the ____.

initialization list

EXERCISES

29. Uses the default inheritance type private.

31. While in the constructor for the funny class, the compiler will only have access to the local variable, `x`, which was passed as a parameter because, by scope rules, the public `x` in the fun class is hidden by the local declaration in class funny. Thus the '`x`' variable assignment is lost once the constructor returns, which is probably not the intent of the code.

33. The following assignment statements in main will not work because of lack of accessibility to the variables in question in main:

```
fun.x   = 1            // ERROR private data
fun.y   = 2;           // ERROR : protected data
funny.x = 11;          // ERROR : private data
funny.y = 12;          // ERROR : protected in Fun,
                       //         private in Funny
funny.u = 14;          // ERROR : private data
funny.v = 15;          // ERROR : protected data
```

35. b = a is implicit downcasting, which is not allowed.

37. a = b meaningless because the pointers a and b have been defined but do not do not point to an object.

39. B* b = a is downcasting; it needs dynamic casting and polymorphism.

PROBLEMS

41. The default explicit constructors are:

```
A :: A () { }

B::B () : A() { }

C::C () : B() { }
```

43. The constructor and destructor are shown below.

```
class Fun
{
   private :
   int  x;

   protected :
   int  y;

   public:
   Fun (int n, int m);
   ~Fun (void);
}; // class Fun

class Funny : public Fun
{
   private :
   int u;

   protected :
   int v;

   public:
   Funny (int n, int m, int p, int q);
   ~Funny (void);
}; // class Funny

Fun :: Fun (int n, int m)
{
// Statements
   x  =  n;
   y  =  m;

}  // Fun (initialization constructor)

Fun :: ~Fun (void)
{

} // Fun (destructor)

Funny :: Funny (int n, int m, int p, int q)
       : Fun (n, m)
{
// Statements
   u  =  p;
   v  =  q;

} // Funny (initialization constructor)

Funny :: ~Funny (void)
{

} // Funny (destructor)
```