Todas las preguntas están respondidas por mi y no puedo asegurar al 100% que sean correctas.

1. Los métodos definidos en una clase derivada nunca pueden acceder a las propiedades privadas de una clase base.
2. Una clase abstracta se caracteriza por no tener definido ningún constructor.
3. La siguiente clase: `class S {public: S(); S(const S &s); virtual ~S();};` constituye una interfaz en C++.
4. Desde un método de una clase derivada nunca puede invocarse un método implementado con idéntica signatura de una de sus clases base.
5. Los métodos virtuales son métodos abstractos.
6. Los métodos abstractos son métodos con enlace dinámico.
7. Los constructores de las clases abstractas son métodos con enlace dinámico.
8. Un atributo de clase debe tener visibilidad pública para poder ser accedido por los objetos de la clase.
9. Un método sobrecargado es aquel que recibe como argumento al menos una variable polimórfica.
10. Un método tiene polimorfismo puro cuando devuelve una variable polimórfica.
11. En C++ no podemos hacer sobrecarga de operadores para tipos predefinidos.
12. En C++, si no se define un constructor sin argumentos explícitamente, el compilador proporciona uno por defecto.
13. En la misma clase, podemos definir constructores con distinta visibilidad.
14. Si no se captura una excepción lanzada por un método, el programa no advierte que ha ocurrido algún error y continua su ejecución normalmente.
15. La instrucción `throw` permite lanzar como excepción cualquier tipo de dato.
16. Las funciones genéricas no se pueden sobrecargar.
17. Dada una clase genérica, se pueden derivar de ella clases no genéricas.
18. De una clase abstracta no se pueden crear instancias, excepto si se declara explícitamente algún constructor.
19. Una clase interfaz no puede tener atributos de instancia. Una clase abstracta sí puede tenerlos.
20. La herencia de interfaz se implementa mediante herencia pública.

1. **Verdadero**: La parte privada de una clase base no es directamente accesible desde la clase privada.
2. **Falso**: las clases abstractas se caracterizan por tener al menos un método virtual puro (abstracto).
3. **Falso**: En una interfaz todos lo métodos se declaran abstractos, y aquí, no lo están.
4. **Falso**: Un ejemplo son los métodos de clase (`static`)
5. **Falso**: Un método abstracto no puede estar implementado en su definición. Uno virtual sí.
6. **Verdadero**: Los métodos abstractos tienen por definición enlace dinámico (varían en el tiempo de ejecución).
7. Responder
8. **Falso**: El ámbito de visibilidad `protected` permite a clases derivadas acceder a métodos de la clase base (la pregunta dice que el método DEBE de ser público).
9. **Falso**: Un método sobrecargado es aquel que tiene el mismo nombre pero distintas firmas (signatures) o definiciones (herencia).
10. **Falso**: El polimorfismo puro se produce cuando una única función o método se puede aplicar a diferentes tipos de argumentos. En el polimorfismo puro hay un solo método o función y cierto numero de interpretación.
11. **Verdadero**: No, no se puede. xd
12. **Verdadero**: En Java y C++, si no se define ninguno de manera explícita, el compilador genera uno con visibilidad pública, llamado constructor de oficio. (No estoy seguro de esta).
13. **Verdadero**: Se pueden declarar constructores privados (Unidad 2, página 28).

toniaai

14. **Falso**: Si no se trata una excepción hará que el programa aborte.
15. **Verdadero**
16. **Falso**: la utilidad principal de una funcion generica es agrupar variables cuyo tipo base no está predeterminado.
17. **Verdadero**
18. **Falso**: no se puede instanciar una clase abstracta.
19. **Verdadero**: Una interfaz es un conjunto de métodos abstractos. Mientras que una clase abstracta es una clase que contiene al menos un método abstracto,.
20. **Verdadero**: una interfaz por definición tiene visibilidad pública.

1. La herencia pública permite a los métodos definidos en una clase derivada acceder a las propiedades privadas de la clase base.
2. Una clase abstracta se caracteriza por declarar al menos un método abstracto.
3. La siguiente clase: `class S {public: Object *o;};` constituye una clase interfaz en C++.
4. Los métodos abstractos siempre tienen enlace dinámico.
5. Los constructores siempre son métodos virtuales.
6. Un método tiene polimorfismo puro cuando tiene como argumentos al menos una variable polimórfica.
7. Un atributo declarado con visibilidad protegida en una clase A es accesible desde clases definidas en el mismo espacio de nombres donde se definió A.
8. Una de las características básicas de un lenguaje orientado a objetos es que todos los objetos de la misma clase pueden recibir los mismos mensajes.
9. Una operación de clase sólo puede ser invocada mediante objetos constantes.
10. En C++, si no se define ningún constructor, el compilador proporciona por defecto uno sin argumentos.
11. Dada una clase genérica, no se pueden derivar de ella clases genéricas.
12. Una clase interfaz no puede tener instancias.
13. Una clase abstracta siempre tiene como clase base una clase interfaz.
14. No se puede definir un bloque *catch* sin su correspondiente bloque *try*.
15. Tras la ejecución de un bloque *catch*, termina la ejecución del programa.
16. Una variable polimórfica puede hacer referencia a diferentes tipos de objetos en diferentes instantes de tiempo.
17. El *downcasting* estático siempre es seguro.
18. El principio de sustitución implica una coerción entre tipos de una misma jerarquía de clases.
19. La sobrecarga basada en ámbito permite definir el mismo método en dos clases diferentes.
20. Una operación de clase no puede tener enlace dinámico.

1. **Falso**: las propiedades privadas de una clase base nunca pueden ser accedidas por una clase base.
2. **Verdadero.**
3. **Falso**: Pregunta 3 del examen anterior.
4. **Verdadero**: Puesto que los métodos abstractos se implementan en distintas clases base, estos cambian en ejecución y por ello deben de tener enlace dinámico.
5. **Falso**
6. **Verdadero**: se dice que un método tiene polimorfismo puro cuando una función o método se puede aplicar a distintos tipos de argumentos, por lo tanto, si el argumento para la función es polimórfico denota que esta puede aplicarse a distintos tipos de función
7. **Falso**: un espacio de nombres agrupa declaraciones (clases, metodos, objetos...).
8. **Verdadero**: Todos los objetos de una misma clase son instancias iguales, por lo tanto pueden recibir lo mismo.
9. **Falso**: una operación de clase (`static`) puede ser ejecutada sin necesidad de instanciar la clase que la contiene.
10. **Verdadero**: (PREGUNTA 12 DEL EXAMEN ANTERIOR)
11. **Falso**: (PREGUNTA 17 DEL EXAMEN ANTERIOR)
12. **Verdadero**: Una clase interfaz no se puede instanciar pero si se puede instanciar una clase derivada de una.
13. **Falso**: Una clase derivada de una interfaz implementa los métodos definidos en la clase base (interfaz). Una clase abstracta no incluye definición de la misma puesto que la implementación

de las clases abstractas también se realiza en las derivadas. Es decir, no se puede no implementar una función de una clase interfaz en una clase derivada de esta.

14. **Verdadero**
15. **Falso:** los bloques catch los usamos para capturar excepciones y evitar que se finalice el programa.
16. **Verdadero**: Definición de variable poliformica.
17. **Falso**: Unidad 6. Páginas 50+
18. **Verdadero**: el principio de sustitución (buscar por LSP) se define como la propiedad de sustituir cualquier clase hijo por padre. Es decir, convertir un tipo por otro, o lo que conocemos como coerción.
19. **Verdadero**: un ejemplo son los métodos de una interfaz, que son definidos en distintas clases a pesar de mantener el mismo nombre.
20. **Verdadero**: una operación de clase no tiene necesidad de ser instanciada para ejecutarse y únicamente puede ser accedida por atributos de clase. Con esto podemos deducir que una operación de clase no puede tener enlace dinámico puesto que si no, debería ser instanciada y usada en distintos ámbitos para variar en ejecución. (Mas o menos, si se entienden los tipos y operaciones de clase al igual que los tipos de enlace se entiende la respuesta, creo).

1. La herencia protegida permite a los métodos de la clase derivada acceder a las propiedades privadas de la clase base.
2. Una clase abstracta se caracteriza por no tener definido ningún constructor.
3. La siguiente clase en C++: `class S {public: virtual ~S()=0;};` define una interfaz.
4. Los métodos virtuales son métodos abstractos.
5. Los métodos abstractos siempre tienen enlace dinámico.
6. Los constructores siempre tienen enlace dinámico.
7. En C++, un atributo de clase debe declararse dentro de la clase con el modificador **`static`**.
8. Un método sobrecargado es aquel que recibe como argumento al menos una variable polimórfica.
9. Un método tiene polimorfismo puro cuando devuelve una variable polimórfica.
10. Un atributo declarado con visibilidad protegida en una clase A es accesible desde clases definidas en el mismo espacio de nombres donde se definió A.
11. Dada la siguiente definición de clase en C++:

```
class TClase {
  public:
    TClase(int dim);
  private:    int var1;
};
```

    La instrucción `TClase c1;` no da error de compilación e invoca al constructor por defecto.
12. Una de las características básicas de un lenguaje orientado a objetos es que todos los objetos de la misma clase pueden recibir los mismos mensajes.
13. Hablamos de encapsulación cuando agrupamos datos junto con las operaciones que pueden realizarse sobre esos datos.
14. Una operación de clase sólo puede ser invocada mediante objetos constantes.
15. En C++ los constructores se pueden declarar como métodos virtuales.
16. En la misma clase, podemos definir constructores con distinta visibilidad.
17. Si no se captura una excepción lanzada por un método, el programa no advierte que ha ocurrido algún error y continua su ejecución normalmente.
18. En C++, es obligatorio especificar qué excepciones lanza una función mediante una cláusula **`throw`** tras la declaración de la función.
19. Dada una clase genérica, se pueden derivar de ella clases no genéricas.
20. Una clase interfaz no debe tener atributos de instancia. Una clase abstracta sí puede tenerlos.

1. **Falso**: Las propiedades privadas de una clase base nunca son accesibles por la clase derivada.
2. **Falso**: Una clase abstracta puede tener un constructor. Las clases abstractas se caracterizan por tener al menos un método abstracto.
3. **Verdadero**: cumple todas las especifaciones para definir una clase interfaz en c++ (Unidad 5 página 77)
4. **Falso**: un método virtual puro sí es un método abstracto, sin embargo, un método virtual puede contener una implementación que puede ser sustituida por una clase derivada. Un método abstracto nunca contiene implementación en la clase base.
5. **Verdadero**: un método abstracto necesita de enlaces dinámicos para poder ser abstracto.
6. **Falso**
7. **Verdadero**: definir un atributo como `static` indica que este es de clase.
8. **Falso**: Un método sobrecargado puede recibir variables poliformicas pero se caracteriza por poder recibir cualquier tipo de variable (signatura). La respuesta es falsa por que puede recibir algo más que variables poliformicas y no solo poliformicas. El método que recibir una variable poliformica se define como método con poliformismo puro.
9. **Falso**: Decimos que un método tiene poliformismo puro cuando recibe almenos una variable polifórmica.
10. **Falso**: Pregunta 7 del anterior examen.
11. Responder
12. **Verdadero**

13. **Falso**: hablamos de encapsulamiento al hecho de ocultar la implementación de métodos. Es una característica de los frameworks.
14. **Falso:** una operación de clase puede ser ejecutado sin necesidad de instanciar (supongo que la explicación de esto es que podemos ejecutar la operación sin necesidad de tener un objeto, es decir, sin necesidad de instanciar. Al tener la posibilidad de no tener el objeto podemos decir que el objeto puede no ser constante).
15. **Falso**: un método virtual nos permite tener un comportamiento polifórmico. Esto choca con la necesidad de tener un tipo determinado cuando construimos ese tipo de objeto determinado. (http://stackoverflow.com/questions/733360/why-do-we-not-have-a-virtual-constructor-in-c)
16. **Verdadero**: podemos tener un constructor por defecto y uno privado. (Unidad 2, páginas 27-28)
17. **Falso**: si no se captura una excepción el programa finaliza su ejecución.
18. **Falso**: se puede no especificar que excepciones se lanzan en la declaración.
19. **Verdadero**: si tenemos una clase genérica `ejemplo<T>` podemos definir una clase no genérica `class prueba extends ejemplo<int>`.
20. **Verdadero.**

1. Un mensaje tiene cero o más argumentos. Por el contrario, una llamada a procedimiento/función tiene uno o más argumentos.
2. La interpretación de un mismo mensaje puede variar en función del receptor del mismo y/o del tipo de información adicional que lo acompaña.
3. En una agregación, la existencia de un objeto *parte* depende de la existencia del objeto *todo* que lo contiene.
4. Una forma de mejorar el diseño de un sistema es reducir su acoplamiento y aumentar su cohesión.
5. Los inicializadores en C++ tienen el formato *nombreAtributo(valor)* y se colocan entre la lista de argumentos y el cuerpo de cualquier método, permitiendo asignar valores a los atributos de instancia de la clase en la que se define dicho método.
6. Un atributo estático ocupa una zona de memoria que es compartida por todos los objetos de la clase en la que se define, aunque no por los objetos de cualquier clase derivada de ella.
7. Un atributo de clase debe tener visibilidad pública para poder ser accedido por los objetos de la clase.
8. La herencia múltiple se produce cuando de una misma clase base heredan varias clases derivadas.
9. Un atributo protegido en la clase base es también protegido en cualquier clase que derive de dicha clase base, independientemente del tipo de herencia utilizado.
10. Cuando se crea un objeto de una clase D que deriva de una clase B, el orden de ejecución de los constructores es siempre *B() D()*.
11. Una clase abstracta es una clase que no permite definir instancias de ella.
12. Un interfaz es una clase abstracta con al menos un método de instancia abstracto.
13. Un método de clase (estático) no puede tener enlace dinámico.
14. Un método virtual en C++ siempre tiene enlace estático.
15. El principio de sustitución implica una coerción entre tipos de una misma jerarquía de clases.
16. La llamada a un método sobrescrito se resuelve en tiempo de compilación.
17. El *downcasting* estático siempre es seguro.
18. Los métodos definidos en una clase genérica son a su vez genéricos.
19. No se puede definir un bloque *catch* sin su correspondiente bloque *try*.
20. La instrucción *throw* (en C++) sólo permite lanzar objetos de clase *exception* o de clases derivadas de ella.

1. **Falso**: una llamada a un procedimiento o función puede no tener argumentos.
2. **Verdadero.**
3. **Falso**: en las agregaciones todo parte la parte se declara fuera del todo.
4. **Verdadero**: (está en la teoria).
5. Responder
6. **Falso**
7. **Falso**: un objeto de la clase puede acceder a una variable de clase privada a través de una operación de clase.
8. **Falso**: la herencia múltiple se produce cuando una clase derivada hereda de varias clases base.
9. **Falso**: si se usa la herencia privada el atributo pasa a tener visibildad privada.
10. **Verdadero**: cuando creamos un objeto de una clase derivada, primero se ejecuta el constructor
11. **Verdadero**: una clase abstracta no puede ser instanciada.
12. **Falso**: una interfaz es un conjunto de métodos abstractos, no una clase
13. **Verdadero**: un metodo de clase no necesita instanciación para ejecutarse por lo tanto, al no depender del tiempo de ejecución el método de enlace es estático
14. **Falso**: un método virtual en c++ es un método abstracto, es decir, un método con enlace dinámico y que varía según la ejecución.
15. **Verdadero**: pregunta 15 del examen anterior.
16. **Falso**: la sobreescritura se produce en tiempo de ejecución. Ocurre en relaciones de herencia de métodos con enlace dinámico.

toniaai

17. **Falso**.
18. **Falso**: se pueden definir métodos no genéricos en una clase no genérica
19. **Verdadero.**
20. **Falso.**

1. **En el cuerpo de una operación de clase no se puede acceder a ningún atributo/operación de instancia de los objetos pasados como parámetro**
2. **En una composición un objeto componente puede formar parte de más de un compuesto.**
3. **La interpretación de un mismo mensaje puede variar en función del receptor del mismo y/o de la información adicional que le acompaña.**
4. **Los destructores en C++ pueden aceptar cualquier número de parámetros.**
5. **Los TAD´s representan una perspectiva orientada a datos, mientras que los objetos reflejan una perspectiva orientada a servicios.**
6. **Un atributo estático ocupa una zona de memoria que es compartida por todos los objetos de la clase en la que se define, aunque no por los objetos de cualquier clase derivada de ella.**
7. **Un atributo privado en la clase base no es directamente accesible en la clase derivada, independientemente del tipo de herencia utilizado.**
8. **Un constructor acepta cualquier tipo de modificador (static, const, public, etc)**
9. **Una clase es una especificación abstracta de una estructura de datos y de las operaciones que se pueden realizar con ella.**
10. **Una operación constante sólo puede ser invocada por un objeto constante.**

1. **Falso**: los atributos o operadores de clase son accesibles desde cualquier otro lado
2. **Falso**: en una composición, el todo es "propietario" de todos sus objetos parte.
3. **Verdadero**: contestada varias veces más arriba
4. **Falso**: no se pueden sobrecargar los constructores de c++
5. **Verdadero**
6. **Falso**: ya ha sido contestado anteriormente
7. **Verdadero**.
8. **Falso**.
9. **Falso**: una clase es un modelo que define un conjunto de variables -el estado, y métodos apropiados para operar con dichos datos
10. **Falso**: el objeto no tiene por que ser constante.

1. **El puntero this es un puntero constante al objeto que recibe el mensaje**
2. **Implementar la forma canónica ortodoxa de una clase es una condición necesaria (aunque no suficiente) para controlar que el valor de un atributo de clase que cuenta el número de instancias de dicha clase esté siempre en un estado consistente.**
3. **La existencia de una relación todo-parte entre dos clases implica necesariamente que el objeto ´todo´ maneja la creación/destrucción de objetos de tipo ´parte´**
4. **La forma canónica de la clase está formada por el constructor, el constructor de copia, el destructor y el operador de asignación**
5. **Si la clase no proporciona un constructor sin parámetros, el compilador en C++ genera uno de oficio**
6. **Tanto composición como herencia son mecanismos de reutilización del software**
7. **Un atributo de clase debe declararse dentro de la clase con el modificador const**
8. **Un atributo de clase público puede ser accedido desde fuera de la clase a través de un objeto de la clase, un puntero o referencia al mismo o mediante el nombre de la clase seguido del operador de ámbito**
9. **Una clase derivada puede añadir nuevos métodos/atributos propios de la clase derivada, pero no modificar los métodos heredados de la clase base**
10. **Una interfaz es la definición de un protocolo para cierto comportamiento, sin especificar la implementación de dicho comportamiento**

1. **Verdadero**: explicado en la teoria.
2. **Falso**: con declarar un atributo de clase `static` es suficiente.
3. **Falso**: si es una relación de agrecación todo-parte de agregación no es necesario. Si es de composición sí.
4. **Verdadero**: teoria.
5. **Verdadero**
6. **Verdadero**: ambos métodos se basan en reutilizar código ya creado, de una forma u otra.
7. **Falso**: un atributo de clase se declara con el modificador `static`
8. **Verdadero**
9. **Falso**: dependiento del tipo de herencia se puede modifica los métodos heredados (abstracta)
10. **Verdadero**

toniaai