1. La herencia de interfaz se implementa mediante herencia pública.	V
2. Una interfaz no puede tener atributos de instancia. Una clase abstracta si puede tenerlos.	V
3. No se puede definir un bloque catch sin su correspondiente bloque try.	V
4. Una variable polimórfica puede hacer referencia a diferentes tipos de objetos en diferentes instantes de tiempo.	V
5. El downcasting siempre es seguro.	F
6. La sobrecarga basada en ámbito permite definir el mismo método en dos clases diferentes.	V
7. En el diseño mediante tarjetas CRC, utilizamos una tarjeta por cada jerarquía de herencia.	F
8. Un espacio de nombres es un ámbito con nombre.	V
9. Un sistema de tipos de un lenguaje asocia a cada tipo una expresión.	V
10. Hacer que el código sea más fácil de entender no es un motivo suficiente para refactorizarlo.	F
11. En Java, los tipos genéricos sólo se pueden aplicar a clases e interfaces.	F
12. En Java, una clase genérica puede ser parametrizada empleado más de un tipo.	V
13. Sean dos clases Base e Hija. La clase Hija hereda de Base. En Java, cuando asignamos un objeto de la clase Hija a una referencia a Base haciendo conversión de tipo explícita estamos haciendo object slicing.	F
14. Una de las principales fuentes de problemas cuando utilizamos herencia múltiple es que las clases bases hereden de un ancestro común.	V
15. Los lenguajes de programación soportan el reemplazo y el refinamiento como métodos de sobrescritura, pero no hay ningún lenguaje que proporcione ambas técnicas (JAVA solo soporta reemplazo y C++ sólo soporta refinamiento).	F
16. Una interfaz puede implementar otra interfaz.	F
17. En java es obligatorio indicar que un método de una clase derivada sobrescribe un método de la clase base con la misma signatura.	F
18. En la sobrecarga basada en ámbito los métodos pueden diferir únicamente en el tipo devuelto.	V
19. En la herencia pública la clase derivada podrá acceder a los atributos privados de la clase base de la que hereda.	F
20. Una clase abstracta se caracteriza por no tener ningún constructor.	F
21. El cambio de una condicional por el uso de polimorfismo es un ejemplo de refactorización.	V
22. Un atributo siempre tiene visibilidad pública.	F
23. El principio de segregación de interfaz indica que el código cliente no debe ser forzado a depender de interfaces que no utilice.	V
24. El usuario de un framework implementa el comportamiento	F

declarado en los interfaces del framework mediante herencia de implementación	
25. El proceso de diseño de un sistema software se debería intentar aumentar la cohesión y reducir el acoplamiento.	V
26. Cuando diseñamos sistemas orientados a objetos las interfaces de las clases que diseñamos deberían estar cerradas a la extensión y abiertas a la modificación.	F
27. La existencia de una sólida colección de pruebas unitarias es una precondición fundamental para la refactorización.	V
28. Existe un catálogo de refactorizaciones comunes de forma que el programador no se ve obligado a usar su propio criterio y metodología para refactorizar código.	V
29. Arraylist es una implementación en el Java Collection Framework de la interfaz List.	V
30. Con el uso de reflexión solo podemos invocar métodos de instancia.	F
31. La siguiente clase class S{ public Object obj; } constituye una interfaz en Java.	F
32. Desde un método de una clase derivada solamente puede invocarse un método implementado con idéntica signatura de una de sus superclases si el método en la clase base tiene enlace dinámico.	F
33. Los métodos con enlace dinámico son métodos abstractos.	F
34. Un método sobrecargado es aquel que recibe como argumento al menos una variable polimórfica.	F
35. La genericidad se considera una característica opcional de los lenguajes orientados a objetos.	V
36. 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.	V
37. En java, podemos definir constructores con distinta visibilidad en la misma clase.	V
38. Los métodos genéricos no se puede sobre sobrecargar ni sobrescribir.	F
39. Una interfaz no tiene instancia. Por ejemplo, dada la interfaz Comprable en Java, no podemos hacer new Comparable().	V
40. Una interfaz en Java obliga a que las **clases no abstractas** que la implementan definan todos los métodos que la interfaz declara.	V