추상클래스와 인터페이스

|  |  |
| --- | --- |
| 1 | 추상클래스에 대한 설명 중 틀린 것을 모두 고르면?  (1) 불완전한 메서드인 추상메서드를 보유한 클래스를 추상클래스라 한다.  (2) ~~추상클래스가 되려면, 클래스가 보유한 모든 메서드가 추상메서드이어야 한다.~~  (3) ~~대부분 추상메서드는 개발자의 실수에 의해 메서드 내용이 누락될 경우 정의된다~~  (4) 추상클래스는 불완전한 클래스이므로 new 연산자에 의해 직접 메모리에 인스턴스를 생성할 수 없다.  (5) 추상클래스가 인스턴스화 되는 시점은 상속관계의 자식클래스가 부모의 추상메서드를 오버라이딩하여  메서드 를 완성한 후, 자식의 인스턴스가 생성될때이다. |
| 2 | 다음 중 틀린 것을 고르면?  (1) 추상메서드가 단 하나라도 존재할 경우 이 클래스는 추상클래스가 된다.  (2) ~~추상메서드란 메서드 정의시 { 브레이스 }는 존재하지만 아무것도 코딩되지 않은 상태의 메서드를~~  ~~의미한다.~~  (3) 추상클래스와 추상메서드임을 선언하려면 abstract 수식자를 이용한다.  (4) **~~메서드 정의시 수식자의 위치는 반드시 접근제한자 뒤에 작성함이 원칙이다.~~** |
| 3 | 다음 설명 중 틀린 것은?  (1) 개발시 클래스의 내용을 확정지을 수 없는 시점 또는 확정지어서는 않되는 경우 추상클래스를 이용  할 수 있다  (2) 추상클래스는 개발자가 클래스의 내용을 고의로 불완전한 상태로 정의해놓고, 하위 클래스를에게 불완전한  내용을 완성하도록 강제하는 수단을 제공한다.  (3) 추상클래스를 상속받는 자식은 불완전한 부모의 추상메서드를 완성할때 까지 컴파일 에러가 발생된다  (4) 추상클래스는 개발시 구현해야 할 기준 및 가이드라인을 제시할 수 있는 장치로써 유용하게 사용된다.  (5) **추상클래스를 상속받는 서브클래스 마저도 추상클래스가 될 수 있다.**  (6) ~~자식클래스는 부모인 추상클래스를 여러 개를 상속받을 수 있다.~~ |
| 4 | 다음 수식자(modifier)에 대한 설명 중 틀린 것은?  (1) abstract 수식자는 "추상적인"이란 뜻이며 메서드나 클래스에 적용할 수 있다.  (2) **final 수식자는 "마지막"이란 뜻이며 클래스에 사용될 경우 더 이상 상속을 허락하지 않겠다는 의미다.**  (3) **~~final 수식자가 메서드에 사용될 경우 오버로딩을 허용하지 않겠다는 의미다.~~ (오버라이딩 불허)**  (4) final 수식자가 변수에 사용될 경우 그 값이 더 이상 변하지 않는다의 의미이므로 상수처럼 사용 될  수 있다.  (5) static 수식자는 "정적인" 이란 뜻이며 인스턴스 멤버 여부를 결정짓는데 중요한 역할을 한다. |
| 5 | 아래는 상수가 될 수 있는 조건을 설명한 것이다.이 조건을 충족시킬 수 있으려면 아래의 변수 x를  어떤 수식자로 선언해야 할지 괄호 안을 작성해 보자.  (1) 상수는 프로그램의 실행되는 동안 그 값이 변해서는 않된다. final  (2) 상수는 모든 인스턴스간에 공유될 수 있어야 한다. static  (3) 상수는 모든 인스턴스가 접근할 수 있어야 한다. public  ( ) **int x = 3;** |
| 6 | 다음 중 틀린 것은?  (1) 클래스를 이용한 현실 반영 시 OOP언어의 ‘다중상속 금지’의 원칙이 때로는 현실을 반영하는데 있어  한계를 드러내는 경우가 있다.  (2) 우리의 현실에선 다중 상속적 모습도 많이 존재한다.  (3) 인터페이스는 추상 메서드만을 모아놓은 객체이며 클래스는 아니다.  (4) 인터페이스나 추상클래스 모두 하위 객체에게 구현을 강제할때 사용될 수 있으나, 이중 인터페이스는 클래스  의 다중 상속 금지의 문제를 해결하는 대안이 될 수 있다.  (5) ~~인터페이스는 extends 를 선언하여 구현해야 한다.~~  (6) implements 와 extends 모두 is a 관계로 본다 |
| 7 | 다음 설명 중 틀린 것을 고르면?  (1) 인터페이스를 구현하는 클래스는 메서드 재정의 의무가 주어진다.  (2) ~~인터페이스의 인스턴스를 생성하려면 new 연산자를 사용한다.~~  (3) **자바언어에서는 인터페이스를 구현한 클래스를 자식 클래스로 보기 때문에, 결과적으로 is a 관계의 성립 뿐**  **아니라,객체간 형변환 및 다형성도 그대로 적용되어진다.**  (4) ~~인터페이스는 클래스이다.~~  (5) **추상 클래스는 추상메서드를 단 하나라도 보유하면 성립하지만, 인터페이스의 경우, 보유한 모든 메서드가**  **추상메서드이어야 한다.**  (6) ~~인터페이스 선언시에도 abstract 수식자를 선언해야 한다.~~ |
| 8 | 1. **public** **abstract** **class** Shape {  2. **private** **int** x;  3. **private** **int** y;  4. **public** **abstract** **void** draw();  5. **public** **void** setAnchor(**int** x, **int** y) {  6. **this**.x = x;  7. **this**.y = y;  8. }  9. }  Which two classes use the Shape **class** correctly? (Choose two.)  ~~A.~~ **~~public~~****~~class~~** ~~Circle~~ **~~implements~~** ~~Shape {~~  **~~private~~****~~int~~** ~~radius;~~  ~~}~~  B. **public** **abstract** **class** Circle **extends** Shape {  **private** **int** radius;  }  ~~C.~~ **~~public~~****~~class~~** ~~Circle~~ **~~extends~~** ~~Shape {~~  **~~private~~****~~int~~** ~~radius;~~  **~~public~~****~~void~~** ~~draw();~~  ~~}~~  ~~D.~~ **~~public~~****~~abstract~~****~~class~~** ~~Circle~~ **~~implements~~** ~~Shape {~~  **~~private~~****~~int~~** ~~radius;~~  **~~public~~****~~void~~** ~~draw();~~  ~~}~~  E. **public** **class** Circle **extends** Shape {  **private** **int** radius;  **public** **void** draw() {}  }  ~~F.~~ **~~public~~****~~abstract~~****~~class~~** ~~Circle~~ **~~implements~~** ~~Shape {~~  **~~private~~****~~int~~** ~~radius;~~  **~~public~~****~~void~~** ~~draw() {}~~  ~~}~~ |
| 9 | Which Man **class** properly represents the relationship "Man has a best friend who is a Dog"?  ~~A.~~ **~~class~~** ~~Man~~ **~~extends~~** ~~Dog { }~~  ~~B.~~ **~~class~~** ~~Man~~ **~~implements~~** ~~Dog { }~~  ~~C.~~ **~~class~~** ~~Man {~~ **~~private~~** ~~BestFriend dog; }~~  D. **class** Man { **private** Dog bestFriend; }  ~~E.~~ **~~class~~** ~~Man {~~ **~~private~~** ~~Dog<bestFriend>; }~~  ~~F.~~ **~~class~~** ~~Man {~~ **~~private~~** ~~BestFriend<dog>; }~~ |
| 10 | 11. **abstract** **class** Vehicle { **public** **int** speed() { **return** 0; }  12. **class** Car **extends** Vehicle { **public** **int** speed() { **return** 60; }  13. **class** RaceCar **extends** Car { **public** **int** speed() { **return** 150; }  ...  21. RaceCar racer = **new** RaceCar();  22. Car car = **new** RaceCar();  23 Vehicle vehicle = **new** RaceCar();  24 System.out.println(racer.speed() + ", " + car.speed() + ", " + vehicle.speed());  What is the result?  A. 0, 0, 0  B. 150, 60, 0  C. Compilation fails.  **D. 150, 150, 150**  E. An exception is thrown at runtime. |
| 11 | **interface** Foo {}  **class** Alpha **implements** Foo {}  **class** Beta **extends** Alpha {}  **class** Delta **extends** Beta {  **public** **static** **void** main( String[] args ) {  Beta x = **new** Beta();  16. //insert code here 16  }  }  Which code, inserted at line 16, will cause a java.lang.ClassCastException?  A. Alpha a = x;  **B. Foo f = (Delta)x;**  C. Foo f = (Alpha)x;  D. Beta b = (Beta)(Alpha)x; |
| 12 | **class** Foo {  **public** **int** a = 3;  **public** **void** addFive() { a += 5; System.*out*.print("f "); }  }  **class** Bar **extends** Foo {  **public** **int** a = 8;  **public** **void** addFive() {  **this**.a += 5; System.*out*.print("b " );  }  }  Invoked with:  Foo f = **new** Bar();  f.addFive();  System.out.println(f.a);  What is the result?  **A. b 3**  B. b 8  C. b 13  D. f 3  E. f 8  F. f 13  G. Compilation fails.  H. An exception is thrown at runtime. |
| 13 | **class** ClassA {}  **class** ClassB **extends** ClassA {}  **class** ClassC **extends** ClassA {}  and:  ClassA p0 = **new** ClassA();  ClassB p1 = **new** ClassB();  ClassC p2 = **new** ClassC();  ClassA p3 = **new** ClassB();  ClassA p4 = **new** ClassC();  Which three are valid? (Choose three.)  **A. p0 = p1;**  B. p1 = p2;  **C. p2 = p4;**  D. p2 = (ClassC)p1;  E. p1 = (ClassB)p3;  **F. p2 = (ClassC)p4;** |
| 14 | 10. **interface** Jumper { **public** **void** jump(); } ...  20. **class** Animal {} ...  30. **class** Dog **extends** Animal {  31. Tail tail;  32. } ...  40. **class** Beagle **extends** Dog **implements** Jumper{  41. **public** **void** jump() {}  42. } ...  50. **class** Cat **implements** Jumper{  51. **public** **void** jump() {}  52.  }  Which three are **true**? (Choose three.)  A. Cat is-a Animal  B. Cat is-a Jumper  C. Dog is-a Animal  D. Dog is-a Jumper  E. Cat has-a Animal  F. Beagle has-a Tail  G. Beagle has-a Jumper |
| 15 | QUESTION 25 Given:  **class** One {  **void** foo() { } // public < protected  }  **class** Two **extends** One {  14. // insert method here  }  Which three methods, inserted individually at line 14, will correctly complete **class** Two? (Choose three.)  A. **int** foo() { /\* more code here \*/ }  B. **void** foo() { /\* more code here \*/ }  C. **public** **void** foo() { /\* more code here \*/ }  D. **private** **void** foo() { /\* more code here \*/ }  E. **protected** **void** foo() { /\* more code here \*/ } |
| 16 | 1. **class** X {  2. X() { System.*out*.print(1); }  3. X(**int** x) {  4. **this**(); System.out.print(2);  5. }  6. }  7. **public** **class** Y **extends** X {  8. Y() { **super**(6); System.*out*.print(3); }  9. Y(**int** y) {  10. **this**(); System.out.println(4);  11. }  12. **public** **static** **void** main(String[] a) { **new** Y(5); }  13. }  What is the result?  A. 13  B. 134  C. 1234  D. 2134  E. 2143  F. 4321 |
| 17 | 11. **public** **interface** Status {  12. /\* insert code here \*/ **int** *MY\_VALUE* = 10;  13. }  Which *three* are valid on line 12? (Choose three.)  A. **final**  B. **static**  C. **native**  D. **public**  E. **private**  F. **abstract**  G. **protected** |