기능의 확장(extends)으로서의 상속

|  |  |
| --- | --- |
| 1 | 상속에 대한 설명 중 맞는 것은?  (1) **~~상속의 대상이 되는 부모 클래스는 둘 이상 둘 수 있으므로, 하나의 클래스에서 사용 가능한 extends 의~~**  **~~선언 개수는 제한이 없다~~ 한번만 가능 (자바에서는 다중상속 금지)**  (2) **~~자식 클래스와 부모 클래스에 동일한 이름의 멤버변수가 존재할 때, 상속 선언 시 문제가 될 수 있다.~~**  (3) ~~상속 시 부모 클래스가 가진 모든 멤버 변수와 멤버 메서드는 자식클래스 파일에 덮어 씌워진다~~  (4) ~~상속의 효과로 데이터가 확장되는 쪽은 부모이다.~~  (5) 객체가 일반적으로 갖는 공통성 있는 특징 및 기능들은 부모클래스에 정의해 놓으면 하위 클래스에서는  동일한 코드를 중복 정의 하지 않아도 되므로, 상속을 이용하면 코드의 재사용성이 높아진다. |
| 2 | 다음 설명 중 올바른 것을 모두 고르면?  class Duck extends Bird{  }  (1) 새를 부모로 둔 오리 클래스를 정의한 것이다.  (2) 이 클래스가 올바로 컴파일 되려면 Bird 클래스가 선행되어 컴파일 되어 있어야 할 것이다.  ~~(3) Bird 를 부모 클래스라 하며 자바에서는 Parent 클래스라 표현한다.~~  ~~(4) Duck 을 자식 클래스라 하며 자바에서는 Child 클래스라 표현한다.~~  (5) 상속을 통해 Duck은 자신이 보유한 멤버변수, 멤버메서드 이외에 부모의 멤버변수, 메서드까지도 내것  처럼 사용이 가능하므로 자식의 기능이 확장된 것으로 볼 수 있다. |
| 3 | 다음 설명 중 올바르지 않은 것을 고르면?  (1) 상속을 이용하면, 코드를 중복 정의 하지 않아도 되며 개발시 재사용이 높아진다.  (2) **~~부모가 자신이 보유한 멤버변수 및 메서드를 private 으로 선언해도 상속받는 자식은 언제든지 부모의~~**  **~~멤버 변수와 메서드에 접근이 가능하다.~~**  (3) ~~extends 키워드는 한 클래스내에서 여러 번 사용이 가능하며, 선언 횟수에 대한 제한은 없다.~~  (4) 현실에서 다중 상속이 존재하지 않듯, OOP언어에서도 다중 상속은 원칙 상 허용되지 않는다.  (5) 상속 시 자식은 부모클래스의 생성자도 물려받게 된다 |
| 4 | 아래 클래스에 대한 설명 중 틀린 것은?  class Bird {  private String name=”새”;  boolean fly;  public void fly(){  System.out.println(“부모가 날아요”);  }  }  class Duck extends Bird {  String name=”난 오리”;  public Duck(){  (가)  }  public void fly(){  System.out.println(“자식이 날아요”); // 오버라이딩  }  }  class UseTest {  public static void main(String[] args){  Bird b=new Bird(); (나)  Duck d=new Duck(); (다)  }  }  (1) ~~부모인 Bird에 이미 name 변수가 존재함에도 불구하고, 자식인 Duck 클래스에 동일한 이름의 name 변수~~  ~~를 선언하였으므로 컴파일 에러가 발생할 것이다.~~  (2) ~~부모 클래스에 이미 존재하는 메서드를 자식이 100% 동일한 메서드를 정의하였으므로, 컴파일 에러가~~  ~~발생할 것이다.~~  (3) ~~(나)에서 Bird 클래스의 인스턴스 뿐 아니라, 상속 관계의 자식인 Duck 클래스의 인스턴스도 생성된다~~.  (4) (가)에서 개발자가 명시하지 않더라도 이미 부모 클래스의 인스턴스 생성을 위한 생성자 호출 코드가  숨겨져 있다고 보아야 한다.  (5) (다)에 의해 Duck 뿐만 아니라, Bird의 인스턴스도 메모리에 함께 생성된다. |
| 5 | 아래 클래스에 대한 설명 중 맞는 것을 모두 고르면?  class Bird {  protected String name=”새”;  boolean fly;  public Bird(boolean f) {  fly = f;  }  }  class Duck extends Bird {  String sound=”quack”;  public Duck(){  super(true);  }  }  class UseTest {  public static void main(String[] args){  Bird b=new Bird(); (가)  b.sound=”JJack JJack”; (나)  Duck d=new Duck(); (다)  }  }  (1) ~~(가)는 Bird 클래스를 메모리에 올리는 과정이므로 아무런 문제가 없다.~~  (2) (~~나)의 코드는 문제가 없이 실행될 것이다.~~  (3~~)~~ **~~(다)의 코드는 Duck 클래스를 메모리에 올리는 과정이므로 아무런 문제가 없다.~~**  **(4) ~~Duck 클래스는 부모인 Bird 클래스의 name 변수는 상속받지 못한다~~**  **상속관계 및 같은 디렉토리는 사용 가능**  (5) 현실과 마찬가지로, 자식보다 부모의 존재가 시점 상 우선해야 하므로, 상속 관계에 있는 자식 클래스를  메모리에 올리는 순간, 부모 클래스의 인스턴스화 작업이 먼저 선행되어진다. 따라서 눈에 보이지는  않지만 Duck 클래스의 디폴트 생성자에는 개발자가 명시하지 않더라도 Bird 클래스를 인스턴스화 시키기  위한 코드인 super() 가 생략되어 있다. |