클래스 간 관계

|  |  |
| --- | --- |
| 1 | 아래의 (A)에서 Computer 클래스는 메모리에 올라갔으나, 부품으로 보유한 Ram, HardDisk 는 아직  메모리에 올라가지 못한 상태이다. (B)라인에 알맞는 코드를 작성하여 com의 부품도 메모리에  올라가도록 처리하라.  class Computer{  Ram ram;  HardDisk disk;  public Computer(){  ram = new Ram();  disk = new HardDisk ();  }  }  class UseComputer{  public static void main(String[] args){    Computer com=new Computer(); (A)  com.ram = new Ram();  com.disk = new HardDisk();  (B)  }  } |
| 2 | 현실에 존재하는 모든 사물은 단독으로 존재할 수 없으며 여러 객체들로 구성된다.  객체 지향 언어인 자바는 이러한 객체들 간의 관계를 크게 2가지로 보고 있는데, 이를 ( )관계와  ( )관계라 한다.  아래의 경우에 해당되는 관계를 채워 넣으세요.   |  |  | | --- | --- | | 자전거는 바퀴, 핸들, 체인 등 개별적 부품으로 구성되어 있다. |  | | 오리는 새이다. |  | |  |  | |  |  |   예를 들어, 볼펜은 스프링, 볼펜 심 , 플라스틱 몸체 등으로 이루어져 있다.  OOP에서는 이와 같이 특정 사물들을 부품으로 보유할 수 있는 관계를 ( ) 관계라 한다. |
| 3 | 빈 칸을 채우세요.  Ex) 오리 -------- 새  (1) 한국인 --------- ( 사람 )  (2) 승용차 -------- ( 차 )  (3) 얼룩말 --------- ( 말 )  (4) 손목시계 -------- ( 시계 )  (5) 종이컵 -------- ( 컵 )  (6) 운동화 --------- ( 신발 )  (7) 외과,내과 의사 ---- ( 의사 )  (8) 민법 -------------( 법 )  (9) 피아노 -------------( 악기 )  (10) 동화책 -------------( 책 ) |
| 4 | 현실의 자동차, 택시, 버스, 트럭을 자바 언어로 반영하되 재사용성을 고려해서 설계해 보세요  class Car{  private class nCar{  }  private class taxi{  }  private class bus{  }  private class truck{  }  String kinds= “”;    public void setKinds(val){  tihs.kinds = val;  }  } |
| 5 | 현실에 존재하는 객체들 중 상위 객체로부터 파생된 경우를 흔히 볼 수 있다.  즉 자신보다 상위 개념의 객체가 존재하는 경우를 자바와 같은 OOP언어에서는 ( 상속 is a ) 관계라 하며, extends 키워드로 표현한다.  이때 상위클래스를 부모클래스라 하며 ( super )클래스라 하고, 자식클래스를 ( sub ) 클래스라 한다. |
| 6 | 다음 설명 중 올바른 것을 모두 고르면?  class Duck extends Bird{  }  (1) 새를 부모로 둔 오리 클래스를 정의한 것이다. o  (2) extends를 is a 로 해석해도 문장에 무리가 없다 o .  (3) 클래스 선언을 Bird extends Duck 으로 순서를 바꾸어도 무리가 없다. x  (4) 이 클래스가 올바로 컴파일 되려면 먼저 Bird 클래스가 컴파일 되어 있어야 할 것이다. o  (5) Bird 를 부모클래스라 하며 자바에서는 parent 클래스 라고 표현한다. x  (6) Duck 을 자식클래스라 하며 자바에서는 child 클래스 라고 표현한다. x |
| 7 | 다음 설명 중 맞는 것은?  class Bird{  String name=”그냥 새”;  boolean fly=true;  int legs=2;  }  class Duck extends Bird{  String name=”오리”;  public Duck(String n){  name=n;  }  public static void main(String[] args){  name=”도날드”; (가)  Duck d=new Duck(); (나)  d.name=”도날드”; (다)  d.fly=false; (라)  }  }  (1) (가)에서 Duck 클래스의 name 멤버변수의 값이 도날드로 바뀔 것이다. x  (2) (나) 에서 오리의 인스턴스가 1개 생성되어 heap 영역에 올라갈 것이다. x  (3) (나)에서 에러가 나지 않는다고 가정했을 때 (다)에서 name 변수값은 도날드로 변경될 것이다. o  (4) (라)의 문장은 제대로 수행될 것이다. o |
| 8 | 다음 설명 중 올바르지 않은 것은?  (1) 상속이란 현실의 상,하위객체간의 관계를 OOP언어가 is a 관계로 표현한 기법이다. o  (2) 상속은 extends 키워드로 구현하며, extends 키워드를 is a 로 바꾸어 이해해도 된다. o  (3) 상속을 이용하면, 코드를 중복정의 하지 않아도 되며 개발시 재사용이 높아진다. o  (4) 부모클래스의 모든 것이 자식클래스에게 물려지지만, 생성자만은 자식 클래스에게 물려지지 않는다. o  (5) extends 키워드는 한 클래스내에서 여러 번 사용이 가능하며, 특별히 제한은 없다. x |
| 9 | 다음 설명 중 올바르지 않은 것은?  (1)자식(sub)클래스는 상속을 통하여 부모(super)클래스의 멤버 변수 및 메서드를 물려받을 수 있다. o  (2)자식클래스가 보유한 변수와 메서드가 이미 부모클래스에도 동일한 명칭으로 존재할 경우, 상속을  명시하는 자식이 보유한 변수와 메서드는 부모의 것으로 덮어써진다. x  (2) 상속을 이용하면, 객체가 일반적으로 갖는 공통성 있는 특징,기능들을 부모클래스에 정의하고,  자식 클래스에서는 동일한 코드를 중복정의 하지 않아도 되므로, 개발시 소스의 재사용성이 높아진다.o  (3) 현실에서도 다중 상속이 존재하지 않듯, OOP언어에서도 다중 상속은 허용되지 않는다. o  (4) 부모클래스에서 멤버변수에 private 으로 접근 제한을 둘 경우, 이 멤버변수 조차도 자식이 접근할 수 있다. x  (5) 부모의 생성자가 상속되지 않는 이유는 생성자의 목적이 해당 객체가 태어날 때 그 객체만이 갖는 개성과  스타일 등 해당 객체만이 가질 수 있는 유일한 초기화 작업을 지원하기 위함이므로, 생성자를 자식객체가  물려받는다 하더라도, 사용할 의미가 없다고 보아야한다. o |
| 10 | 아래 클래스에 대한 설명 중 맞는 것을 모두 고르면?  class Bird{  String name=”새”;  boolean fly;  public Bird(Boolean f){  fly=f;  }  }  class Duck extends Bird{  String sound=”quack”;  }  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. x  (2) (나)의 코드는 문제가 없이 실행될 것이다. x  (3) (다)의 코드는 Duck 클래스를 메모리에 올리는 과정이므로 아무런 문제가 없다.x  (4) 자식클래스는 부모클래스의 name,fly 변수를 상속받는다.o  (5) 자식클래스는 부모클래스의 생성자를 상속받는다. x  (6) 현실과 마찬가지로, 자식보다 부모의 존재가 시점상 우선해야 하므로, 상속 관계에 있는 자식 클래스를  메모리에 올리는 순간, 부모클래스의 인스턴스화 작업이 먼저 선행되어져야 한다. o |
| 11 | 현실에 존재하는 여러 사물들은 단독적이기보다 서로 관계를 맺고 존재하는 경우가 대부분이다.  특히 OOP언어가 주목하고 있는 객체간의 관계는 2가지가 있는데, 하나는 객체간의 상,하위 관계를  ( is a )관계로 보고 ( extends )키워드로 구현한다.  또 다른 하나는 객체가 특정 객체들을 부품으로 보유하는 관계를 ( has a )보고, 객체자료형을  ( 멤버 변수 )로 보유하여 구현한다. |