객체간 형변환

|  |  |
| --- | --- |
| 1 | 다음 설명 중 맞는 것을 모두 고르면?  (1)자바의 기본 자료형간의 형변환은 같은 종류에 상관없이 모두 가능하다. x  숫자 : 정수 byte1 < short2 < int4 < long8  실수 float 4 < double8    문자,  논리값  (2)작은 자료형이 큰 자료형으로 형변환 되어질때는 데이터의 손실이 발생한다.x  (3)큰 자료형이 작은 자료형으로 형변환 되어질때는 데이터의 손실이 발생하므로 프로그래머가 강제형  변환을 명시해야 한다.o  (4)작은자료형에서 큰자료형으로의 형변환을 promotion 이라 하고, 그 반대의 경우를 demotion이라 한다 o |
| 2 | 다음 설명 중 틀린 것을 모두 고르면?  (1) 자바의 자료형은 기본자료형 이외에 객체자료형이 존재한다. o  (2) 자바의 기본 자료형은 같은 종류별로 분류가 가능하지만(예 숫자형간) 객체자료형은 같은 종류라는  개념이 존재할 수 없기 때문에, 형변환이란 기법은 객체자료형에는 적용되지 않는다.x  (3) 상속관계에 있는 클래스들은 서로 같은 종류의 데이터 타입으로 간주되어 상,하위 객체간 Casting  이 가능하다. o  (4) 자식 자료형에서 부모자료형으로의 형변환을 UPCasting이라 한다. o  (5) 부모 자료형에서 자식 자료형으로의 형변환을 UPCasting이라 한다. x  (6) 부모자료형에서 자식자료형으로 형변환시 데이터 손실이 발생한다. x |
| 3 | 1) 옆집강아지, 앞집강아지, 옆동네 강아지, 친구네 강아지를 한마디로 가리킬 수 있는 말 ( 강아지 )  2) 전선줄에 앉아 있는 여러 종류의 새를 가리킬 수 있는 말 ( 새 )  일상 생활에서 수많은 대상을 가장 편안하고 폭넓게 가리킬 수 있는 방법은 그 객체의 상위객체로  가리키면 되는데, 이와 같이 상위에 존재하는 하나의 데이터 타입으로 여러 객체들을 가리킬 수 있는  기법을 OOP에서는 ( 다형성 ) 즉 polymorphism 이라 한다. |
| 4 | 다음 설명 중 맞는 것은?  class Bird{  String name="난 그냥새";  }  class Duck{  String name="난 오리";  }  class Test{  public static void main(String[] args){  Bird b1=new Bird(); (A)  Duck dk=new Duck(); (B)  Bird bird=dk; (C)  Duck d=(Duck)bird; (D)  System.out.println(d.name);(E)  }  }  (1) (A)에서 메모리에 올라가는 인스턴스는 Duck 과 Bird 이다. x  (2) (B)에서 Duck 클래스가 올라가기 전에 Bird 클래스가 먼저 메모리에 올라가게 되어 있다. o  (3) (C)에서 dk 를 레퍼런스할 수 있는 자료형은 Duck 형임에도 불구하고 Bird 형으로 레퍼런스하고  있으므로 오류가 날 것이다. x  (4) (D)는 부모자료형인 bird를 자식자료형인 Duck으로 강제 형변환을 시도한 것이며 이러한 형변환을  업케스팅이라 한다.x  (5) (E)에서의 출력결과는 "난 그냥새"가 출력된다. x |
| 5 | 상속에 대한 설명 중 틀린 것은 ?  (1) 상속관계에 있는 클래스들간에는 같은 종류의 자료형으로 본다. o  (2) 자바의 기본자료형간 발생하는 형변환에서는 데이터의 손실이 발생할 수도 있지만, 자바의 객체  자료형간 발생하는 형변환은 레퍼런스가 가리킬 수 있는 범위의 문제이므로,데이터의 기본자료형  간의 형변환과는 차이가 있다.o  (3) 부모클래스 Bird에 name이란 멤버변수가 존재하고 자식클래스인 Duck 에도 name 이란 멤버변수가  존재할때 Duck의 name 변수는 Bird의 name 변수로 대체되어 버린다.x  (4) 상속은 is a 관계로 표현할 수 있다. o |
| 6 | 다음 설명 중 틀린 것은 ?  (1) 하나의 클래스내에 메서드명을 중복정의 할 수 있는 기법을 오버로딩이라 한다. o  (2) 생성자도 메서드이므로 생성자 오버로딩이 가능하다. o  (3) 부모클래스에 이미 정의된 메서드명을 자식클래스에서 중복 정의하면 에러가 발생한다. x  (4) 부모클래스의 메서드를 오버라이딩 하려면 메서드명 뿐만 아니라 인수의 갯수 자료형까지 똑같이  해야 한다. o |
| 7 | 다음 설명 중 맞는 것은?  class Animal{  String name="난동물";  public Animal(String n){ (C)  name=n;  }  public Animal(){ (D)  }  }  class Tiger extends Animal{  String name="난호랑이";  public Tiger(){  }  public Tiger(String n){  name=n;  }  }  class UseAnimal{  public static void main(String[] args){  Tiger t1=new Animal(); (A)  Animal a1=new Tiger(); (B)  System.out.println(a1.name); (E)  }  }  (1) (A)는 다형성을 이용한 코드이므로 정상적인 컴파일이 될 것이다. x  (2) (B)에서 실질적으로 메모리에 올라간 인스턴스는 Tiger 뿐이다. x  (3) (B)에서 new 연산자뒤의 생성자가 잘못된 방법으로 호출되고 있다.x  (4) (B)에 의해 호출되는 부모의 생성자는 (C) 생성자이다. X  (5) (E)에서 출력되는 결과는 "난 동물" 일 것이다.O |
| 8 | 다음 설명 중 맞는 것은?  class Plant{  String name="난 식물";  public void grow(){  System.out.println("식물이 자랍니다.");  }  }  class Flower extends Plant{  String name="난 꽃";  public void grow(){  System.out.println("꽃이 자랍니다.");  }  }  class UsePlant{  public static void main(String[] args){  Plant p1=new Plant(); (A)  System.out.println(p1.name); (B)    Plant p2=new Flower(); (C)  System.out.println(p2.name); (D)    p2.grow(); (F)  }  }  (1) (A)라인에 의해 Plant의 인스턴스뿐 아니라 Flower의 인스턴스도 메모리에 올라간다. x  (2) (B)에서 출력되는 결과는 "난 꽃" 이다. x  (3) (C)라인에 의해 메모리에 올라가는 인스턴스는 Flower 뿐이다. x  (4) (D)에서 출력되는 결과는 "난 꽃" 이다. x  (5) (F)의 메서드 호출에 출력되는 결과는 "꽃이 자랍니다." 이다. o |
| 9 | 다음 설명 중 틀린 것은?  (1) 상속관계에 있는 클래스가 메모리에 올라갔을 경우, 부모자료형과 자식자료형간의 형변환을 허용  하는 기법을 케스팅이라 한다. o  (2) 현실에서 상위개념으로 여러 개별적 객체들을 가리키는 기법을 다형성이라 하는데, 자바언어는  OOP 언어이므로 현실의 다형성의 표현이 가능하다. o  (3)  (4) (D)에서 출력되는 결과는 "난 꽃" 이다. o  (5) (F)의 메서드 호출에 출력되는 결과는 "꽃이 자랍니다." 이다. o |