**생성자**

|  |  |
| --- | --- |
| 1 | 아래의 클래스에 대한 설명으로 틀린 것은?  **class** Rose{  **int** leaf=0;  String color="red";    **public** **void** fall(){  leaf=0;  }    }  **class** UseRose{  **public** **static** **void** main(){  Rose r1=**new** Rose(); // (가)  r1.fall(); //(나)  r1.bloom(); //(다)  }  }  (1) (다)에서의 bloom() 은 Rose 클래스에 존재하지 않는 메서드이므로 컴파일 에러가 발생한다.O  (2) (나)에서의 fall()은 Rose 클래스의 멤버 메서드를 호출하고 있는 것이다.O  (3) (가)에서 장미 인스턴스를 생성하기 위한 new 연산자의 뒤에 오는 Rose()는 Rose 클래내에 작성  한 적이 없으므로 컴파일 에러가 발생 할 것이다.X  (4) (가)에서 new 연산자 뒤에 오는 Rose() 메서드는 생성자 메서드이고, 개발자가 정의하지 않으면  컴파일러에 의해 존재하게 된다. O |
| 2 | 생성자 메서드에 대한 설명 중 틀린 것은?  (1) 객체가 new로 생성되어 질때 무조건적으로 호출이 강제되는 메서드이다.O  (2)생성자 메서드의 이름은 반드시 클래스명과 일치해야 한다.O  (3)생성자에는 반환형을 두어서는 안된다.O  (4)생성자는 개발자가 정의하지 않으면 객체 생성시 존재하지 않는 메서드로 인한 에러가 발생하게  되므로, 컴파일러에 의해 자동으로 정의되어 진다. O  (5)개발자가 생성자를 정의해도 디폴트 생성자는 컴파일러에 의해 계속 존재하게 된다.X  (6) Sun사가 객체의 생성시, 생성자 메서드를 무조건 적으로 강제하고 있는 이유는 개발자로 하여금  클래스로부터 만들어지는 모든 객체에 대해 초기화 작업의 기회를 주기 위해서이다.O |
| 3 | 생성자 메서드와 일반메서드의 차이점에 대한 설명 중 틀린 것은?  (1)생성자가 아니더라도 객체에 개성을 부여하는 방법은 일반 메서드로도 가능하다.O  (2)생성자도 메서드이다. O  (3) 실행 시기적 관점에서 본다면 생성자와 일반 메서드는 단지 작업의 착수 시점이 다를 뿐이다.  특히 생성자의 호출을 무조건적으로 강제한 이유는 개발자로 하여금 객체 생성 초기에 원하는  초기화 작업을 할 수 있도록 배려한 것이다.O  (4) 위(3)번의 배려에도 불구하고 개발자가 생성자를 정의하지 않으면, 생성자가 존재하지 않게 되어  new 연산자에서 호출시 에러가 발생하게 되므로, sun에서는 개발자가 정의하지 않을 경우 최소  한의 관여만 하여 디폴트 생성자를 두게끔 처리했다O.  (5) 생성자에 반환형을 둘 수 있다.X  (6) 현실에 비유하자면 생성자는 사물 제작 초기부터 해야 할 작업에, 일반메서드는 사물을 제작한  이후에 해야 하는 작업에 비유 할 수 있다O. |
| 4 | 생성자 메서드에 대한 설명 중 틀린 것은?  (1)현실에서 사물을 만들 때 고려할 사항이 있고, 만들어진 이후에 고려해야 할 사항이 있듯, 생성자      메서드는 객체지향 프로그래머가 사물을 생성할 때, 생성 초기에 해야 할 작업을 위해 기회를      준 것이다.O  (2)Sun에서 생성자 호출을 new 연산자 바로 뒤에 존재시키는 이유는 객체가 생성되어질때 똑같은     모습인 아닌 개성있는 모습으로 차별화 될수 있는 기회를 주기 위함이다.O      new Car(“blue”, “v8”);  (3)생성자는 메서드가 아니다.X  따라서 메서드 오버로딩 마저 적용됨  (4)생성자는 개발자가 정의할 수 없다X |
| 5 | 아래의 클래스를 생산 시점부터 빨간색인 하이힐을 만들수 있는 클래스로 완성하시오  **class** HighHeel{  String color="None Color";  public HighHeel(String c){  color=c;  }  }  HighHeel h = new HighHeel("red"); |
| 6 | 아래의 클래스를 생산 시점부터 검정색으로 만들 수 있는 기능 및 생산한 이후에 언제든 색상을  빨간색으로 변경 할 수 있는 클래스로 완성하시오  **class** SportsCar{  String color="None Color";  } |