
JDK8로 배우는 자바 프로그래밍
어서와
Java는
처음이지!
천인국 지음
False
True
INFINITY BOOKS

TEST BANK

CHAPTER 6

1. 필드에 직접 접근하여서 사용하는 것보다 접근자 메소드와 설정자 메소드를 통해서 사용하는 것이 좋은 이유는 무엇인가?

2. 다음 프로그램의 실행 결과는 무엇일까? 이유를 설명하라.

```
public class Hello {

   public static void main(String[] args) {
        Object x = null;
        giveMeAString(x);
        System.out.println(x);

   }

   static void giveMeAString(Object y) {
        y = "This is a string";
   }
}
```

3. 다음 프로그램에서 잘못된 부분을 모두 지적하고 올바르게 수정하시오. 그리고 수정된 후의 출력 결과를 쓰시오.

```
class Television {
        private String model;
        void setModel(String b) {  // 설정자
             model = b;
        }
        void getModel() {          // 접근자
             return model;
        }
}
public class TelevisionTest {
        public static void main(String[] args){
             Television t = new Television;
             t.setModel("STV-101");
             String b = getModel();
        }
}
```

4. 다음은 영화를 나타내는 Movie 클래스이다. 질문에 답하라.

```
public class Movie
{
   private String title, director, actors;
}
```

(1) 각 필드에 대한 접근자와 설정자를 작성하여 보라.

(2) Movie 클래스를 UML로 그려보자. 메소드도 포함시킨다..

(3) Movie 클래스를 자바로 작성하여 보라.

(4) Movie 객체를 하나 생성하고, 생성된 객체의 title 속성을 “Transformer"로 변경하여 보자.

5. 다음 프로그램의 실행 결과는?

```
class A
{
    public A(int x){}
}
class B extends A { }
public class Test
{
    public static void main (String args [])
    {
        A a = new B();
        System.out.println("실행 완료");
    }
}
```

6. 다음 프로그램의 실행 결과는?

```
public class Test
{
    void sub()
    {
         int [] a1 = {3,4,5};
         int [] a2 = modify(a1);
        System.out.println(a1[0]+" "+a1[1]+" "+a1[2]);
        System.out.println(a2[0]+" "+a2[1]+" "+a2[2]);
    }
    int [] modify(int [] a3)
    {
        a3[1] = 10;
        return a3;
    }
```

```
    public static void main(String [] args)
    {
         Test p = new Test();
        p.sub();
    }

}
```

7. 아래 클래스의 기본 생성자는 어떻게 선언될까?

```
public class Test { }
```

8. 다음 프로그램의 출력은?

```
public class Test
{
    public static void main(String args[])
    {
        class MyClass
        {
            public int i = 3;
        }
        Object o = (Object)new MyClass();
        MyClass obj = (MyClass)o;
        System.out.println("i = " + obj.i);
    }
}
```

9. 다음 프로그램의 출력은?

```
class Parent
{
   Parent()
    {
        System.out.print("Parent");
    }
}
public class Child  extends Parent
{
    public static void main(String[] args)
    {
        new Child();
        new Parent();
    }
}
```

10. 다음 프로그램의 출력은?

```
class Test
{
    public static void main(String [] args)
    {
        Test p = new Test();
        p.sub();
    }

    void sub()
    {
        boolean b1 = false;
        set(b1);
        System.out.println(b1);
    }

    void set(boolean b1)
    {
        b1 = true;
    }
}
```

11. 다음 프로그램의 출력은?

```
class Point { int x; int y;}
public class Test
{
    public static void main(String [] args)
    {
        Test obj = new Test();
        obj.sub();
    }

    void sub()
    {
        Point p = new Point();
        p.x = 10;
        p.y = 20;
        set(p);
        System.out.println(p.x+","+p.y);
    }

    void set(Point p)
    {
```

```
        p.x = 30;
        p.y = 40;
    }
}
```

12. 다음 프로그램의 출력은?

```
public class Test
{
    int s;
    public static void main(String [] args)
    {
        Test p = new Test();
        p.sub();
    }

    void sub()
    {
        int x = 5;
        setDouble(x);
        System.out.print(x + " ");
        System.out.println(s);
    }

    void setDouble(int x)
    {
        x = x*2;
        s = x;
    }
}
```

13. 아래 코드의 빈칸에 Inner 클래스의 객체를 생성하는 문장을 넣어보자.

```
class Outer
{
    class Inner { }
}
class Test
{
    public static void main (String [] args)
    {
        Outer f = new Outer();
        _____;
    }
}
```

14. 아래 코드는 오류를 가지고 있다. 올바르게 수정하라.

```
public class Test
{
    public static void main (String [] args)
    {
        Object obj = new Object()
       {
          public boolean equals(Object obj)
          {
             return true;
          }
       }

       System.out.println(obj.equals("Hello"));
    }
}
```

15. 다음 프로그램의 출력은?

```
public class Test
{
    void Test()
    {
       System.out.println("Class Test");
    }
    public static void main(String[] args)
    {
       new Test();
    }
}
```

16. 다음의 은행 업무를 기술한 문장을 읽고 클래스의 후보를 생각하여 보자.

> 은행은 정기 예금 계좌와 보통 예금 계좌를 제공한다. 고객들은 자신의 계좌에 돈을 입금할 수 있으며 계좌에서 돈을 인출할 수 있다. 그리고 각 계좌는 기간에 따라 이자를 지급한다. 계좌마다 이자는 달라진다.

(1) 이 문제 영역 기술 문서를 읽고 잠재적인 클래스 후보를 찾아보라. 참고로 업무 기술서의 문장에 등장하는 명사가 클래스의 후보가 된다.

(2) 이 문제를 해결하는 데 필요한 클래스만을 선별하여 보자.

(3) 각 클래스에 필요한 필드와 메소드를 생각하여 보자.

17. 다음 질문에 간단히 답하라.

(1) 생성자는 어떤 용도로 쓰이는가?

(2) 중복 정의된 메소드들은 어떻게 구별되어서 호출되는가?

(3) 키워드 this는 무엇을 가리키는가?

(4) 정적 변수와 인스턴스 변수의 차이점은 무엇인가?

(5) 객체가 매개 변수로 전달될 때는 어떤 값이 전달되는가?

(6) 왜 정적 메소드는 인스턴스 변수를 참조할 수 없는가?

18. 다음 코드에서 오류를 찾으시오.

(1)
```
public class Point {
   private int x, y;
   public void Point(int x, int y) {
         x = x;
         y = y;
   }
}
```

(2)
```
public class MyMath {
   public int getRandom() {
         return (int)Math.random();
   }
   public double getRandom() {
         return Math.random();
   }
}
```

(3)
```
public class MyClass {
   private String getName() {
         return "MyClass";
   }
   public static String getClassName() {
         return getName();
   }
}
```

19. 정육면체를 나타내는 클래스 Cube가 다음과 같이 정의되어 있다.

```
public class Cube {
   private double side;    // 정육면체의 한변
   public double getSide() {
         return side;
   }
   public double getVolume() {
         return side*side*side;
   }
}
```

(1) 매개 변수가 없는 생성자를 작성하라. 이 생성자는 side를 0으로 할당한다.

(2) 또 하나의 생성자를 중복 정의하라. 이 생성자는 매개 변수를 통하여 전달된 값을 side에 할당한다.

20. MyMetric이라는 클래스를 작성하고 여기에 킬로미터를 마일로 변환하는 정적 메소드인 kiloToMile()을 작성하라. 또 반대로 마일을 킬로미터로 변환하는 정적 메소드 mileToKilo()로 작성하라. MyMetricTest 클래스에서 이들 정적 메소드를 호출하여 테스트하여 보자.

21. 객체 중에는 전체 시스템을 통 털어서 딱 하나만 존재하여야 하는 것들이 있다. 다음의 프로그램을 분석하여서 왜 객체가 하나만 생성되는지를 설명하라. 단 이러한 객체를 생성할 때는 new를 사용하지 않고 정적 메소드 getInstance()를 호출하여야 한다. 이것은 싱글톤 디자인 패턴(singleton design pattern)으로 불린다.

```
class Single {
    private static Single s_instance;
    public static Single getInstance() {
        if (s_instance == null) {
            s_instance = new Single();
        }
        return s_instance;
    }
}
public class SingleTest {
    public static void main(String[] args) {
        Single obj1 = Single.getInstance();
        Single obj2 = Single.getInstance();
    }
}
```

22. 원을 나타내는 Circle이라고 이름 붙여진 클래스를 설계하여 보자. Circle은 반지름 r과 중심의 좌표 cx, cy를 필드로 가진다. 또한 원의 넓이를 계산하여서 반환하는 area()를 메소드로 가진다. 각 필드에 대한 접근자 메소드와 설정자 메소드도 정의한다. 먼저 Circle 클래스를 UML로 그린다. Circle 클래스를 작성하고 객체를 생성하여서 테스트하라.

23. 책을 나타내는 Book 클래스를 정의하여 보자. Book 클래스는 제목(title)과 저자(author)를 나타내는 필드를 가진다. 필드는 모두 private로 선언한다. 각 필드에 대하여 접근자와 설정자 메소드를 정의하고 이것을 통하여 제목과 저자를 설정하여 보자.

24. 주사위를 나타내는 클래스인 Dice를 작성하여 보자. Dice 클래스에 필요한 필드와 메소드를 생각하여 보자. 메소드에는 주사위를 굴리는 메소드인 roll()을 포함하라. roll() 메소드를 작성할 때 난수를 얻는 다음 문장을 참조하라.

```
face = (int) (Math.random() * 6) + 1;
```

Dice 클래스를 테스트하기 위한 별도의 클래스를 작성하여 테스트하라.

25. 2차원 공간에서 하나의 점을 나타내는 Point 클래스를 작성하여 보자. Point 클래스는 x좌표와 y좌표를 나타내는 필드를 가진다. 또한 좌표를 설정하는 set(int x, int y) 메소드와 좌표의 값을 화면에 출력하는 print() 메소드를 가진다. Point의 객체를 생성하여서 테스트하여 보자.

26. 직원을 나타내는 Employee 클래스를 작성하여 보자. 직원은 이름, 전화 번호, 연봉을 필드로 가지고 있다. 각 필드에 대하여 접근자와 설정자를 작성하라. EmployeeTest 클래스를 작성하여서 Employee 객체를 생성하고 테스트하라.

27. 은행 계좌를 나타내는 BankAccount 클래스에 다음과 같은 기능을 하는 메소드를 추가하고 테스트하라. 필요한 필드는 추가하라.

```
// 현재 객체의 잔액에서 amount만큼을 otherAccount 계좌로 송금한다.
public int transfer(int amount, BankAccount otherAccount)
{
   ...
}
```

28. Average 클래스 안에 다음과 같이 getAverage()를 중복 정의하고 테스트하라.

(1) 두 개의 정수를 받아서 평균을 구하는 메소드 getAverage(int a, int b)를 정의하여 보자.

(2) 세 개의 정수를 받아서 평균을 구하는 메소드 getAverage(int a, int b, int c)를 중복 정의하여 보자.

29. 강아지를 나타내는 Dog이라는 이름의 클래스를 설계한다. Dog 클래스는 다음과 같은 필드를 가져야 한다.

- name: 강아지의 이름, 전용 멤버
- breed: 강아지의 종류, 예를 들면 “요크셔테리어”, 공용 멤버
- age: 강아지의 나이, 전용 멤버

Dog 클래스는 다음과 같은 생성자와 메소드를 가져야 한다. 초기화되지 않은 필드를 null이나 0으로 초기화하라.

- Dog(String name, int age): 강아지의 이름과 나이를 초기화
- Dog(String name, String breed, int age): 강아지의 이름, 종류, 나이를 초기화

30. 비행기를 나타내는 Plane라는 이름의 클래스를 설계하라. Plane 클래스는 제작사(예를 들어서 에어버스), 모델(A380), 최대 승객수(500)를 필드로 가지고 있다.

(1) 필드를 정의하라. 모든 필드는 전용 멤버로 하라.
(2) 모든 필드에 대한 접근자와 설정자 메소드를 작성한다.
(3) Plane 클래스의 생성자를 몇 개를 중복 정의하라. 생성자는 모든 데이터를 받을 수도 있고 아니면 하나도 받지 않을 수 있다.
(5) PlaneTest라는 이름의 테스트 클래스를 만드는데 main()에서 Plane 객체 여러 개를 생성하고 접근자와 설정자를 호출하여 보라.
(6) Plane 클래스에 지금까지 생성된 비행기의 개수를 나타내는 정적 변수인 planes를 추가하고 생성자에서 증가시켜 보자.
(7) Plane 클래스에 정적 변수 planes의 값을 반환하는 정적 메소드인 getPlanes()를 추가하고 main()에서 호출하여 보라.

31. 상자를 나타내는 Box라는 이름의 클래스를 설계하라. Box 클래스는 상자의 높이, 너비, 높이를 필드로 가지고 있다. 박스가 비어 있는지 그렇지 않은지를 나타내는 empty라고 하는 필드도 추가한다. Box 클래스의 생성자를 중복 정의하라. 생성자는 모든 데이터를 받을 수도 있고 아니면 하나도 받지 않을 수 있다. 새로 생성된 Box는 비어 있다고 가정한다.

32. 영화를 나타내는 Movie라는 이름의 클래스를 설계하라. 제목, 감독, 제작사를 나타내는 필드를 가진다. Movie 클래스의 생성자를 중복 정의하라. 생성자는 모든 데이터를 받을 수도 있고 아니면 하나도 받지 않을 수 있다.

33. 은행 계좌를 나타내는 BankAccount 라는 이름의 클래스를 설계하라. BankAccount 클래스는 이름, 계좌 번호, 잔액, 이자율을 나타내는 필드를 가진다. BankAccount 클래스의 생성자를 중복 정의하라. 생성자는 모든 데이터를 받을 수도 있고 아니면 하나도 받지 않을 수 있다.