Chapter 09.

접근제어 지시자와 정보은닉, 그리고 캡슐화

나는 정말 JAVA를 공부한 적이 없다

09-1. 정보은닉(Information Hideing)

■ 객체지향 관점에서 넌 빵점이야!

나 정말 JAVA를 공부한 적 없는데

```
class FruitSalesMain4
{
    public static void main(String[] args)
    {
        FruitSeller seller = new FruitSeller(0, 30, 1500);
        FruitBuyer buyer = new FruitBuyer(10000);
        seller.myMoney += 500;
        buyer.myMoney -= 500;
        seller.numOfApple -= 20;
        buyer.numOfApple += 20;
        System.out.println("과일 판매자의 현재 상황");
        seller.showSaleResult();
        System.out.println("과일 구매자의 현재 상황");
        buyer.showBuyResult();
    }
}
```

500원을 내고

그냥 사과 스무 개 가져가는 걸!

메소드를 통해서 정의한 판매의 규칙이 완전히 무너졌다!

■ 정보은닉이라? 인스턴스 변수의 private

나 정말 JAVA를 공부한 적 없는데

```
class FruitSeller
{
    private int numOfApple;
    private int myMoney;
    private final int APPLE_PRICE;
```

외부에서의 접근 금지!

```
class FruitBuyer
{
    private int myMoney;
    private int numOfApple;
```

외부에서의 접근 금지!

- private과 같은 키워드를 가리켜 접근제어 지시자라 한다.
- 인스턴스 변수의 private 선언으로 인해서 메소드가 유일한 접근수단

나는 정말 JAVA를 공부한 적이 없다

09-2. 접근제어 지시자

private과 public

나 정말 JAVA를 공부할 줄 몰라요

- private - 클래스 내부(메소드)에서만 접근 가능.
- public - 어디서든 접근 가능(접근을 제한하지 않는다).

```
class AAA
{
    private int num;
    public void setNum(int n) { num=n; }
    public int getNum() { return num; }
    . . . .
}
```

```
class BBB
{
    public accessAAA(AAA inst)
    {
        inst.num=20;
        inst.setNum(20);
        System.out.println(inst.getNum());
    }
    . . . .
}
```

num은 private 멤버이므로 컴파일 불가!

setNum, getNum은 public이므로 호출 가능!

접근제어 지시자를 선언하지 않는 경우(default)

나 정말 JAVA를 공부한 적 없는데

- 접근제어 지시자 선언하지 않는 경우 - default 선언
- default 선언은 동일 패키지 내에서의 접근 허용

```
package orange;

class AAA    // package orange로 묶인다.
{
    int num;
    . . . .
}

class BBB    // package orange로 묶인다.
{
    public init(AAA a) { a.num=20; }
    . . . .
}
```

default 선언!

BBB는 AAA와 동일 패키지로 선언되었으므로 접근 가능!

protected

나 정말 JAVA를 공부한 적 없는데

- protected - 상속관계에 놓여 있어도 접근을 허용
- default 선언으로 접근 가능한 영역 접근 가능,
- 그리고 상속관계에서도 접근 가능

```
class AAA
{
    protected int num;
    . . . .
}

class BBB extends AAA
{
    protected int num;          // 상속된 인스턴스 변수

    public init(int n) { num=n; }    // 상속된 변수 num의 접근!
    . . . .
}
```

상속을 의미함.

■ 접근제어 지시자의 관계

나 정말 JAVA를 공부한 적이 없다구요

지시자	클래스 내부	동일 패키지	상속받은 클래스	이외의 영역
private	●	×	×	×
default	●	●	×	×
protected	●	●	●	×
public	●	●	●	●

```
public > protected > default > private
```

나는 정말 JAVA를 공부한 적이 없다

09-3. public 클래스와 default 클래스

default 클래스

- default 클래스
  - 동일한 패키지 내에 정의된 클래스에 의해서만 인스턴스 생성이 가능

```
package apple;
class AAA    // default 클래스 선언
{
    . . . .
}
```

```
package peal;
class BBB    // default 클래스 선언
{
    public void make()
    {
        apple.AAA inst=new apple.AAA();
        . . . .
    }
    . . . .
}
```

인스턴스 생성 불가! AAA와 BBB의 패키지가 다르므로!

파일을 대표할 정도로 외부에 의미가 있는 클래스 파일을 public으로 선언한다.

public 클래스

나는 정말 JAVA를 고민할 것 없이 좋아해

- public 클래스
  - 동일한 패키지 내에 정의된 클래스에 의해서만 인스턴스 생성이 가능
  - 하나의 소스파일에 하나의 클래스만 public으로 선언 가능
  - public 클래스 이름과 소스파일 이름은 일치해야 한다.

```
package apple;
public class AAA     // public 클래스 선언
{
     . . . .
}
```

```
package peal;
public class BBB     // public 클래스 선언
{
     public void make()
     {
          apple.AAA inst=new apple.AAA();
          . . . .
     }
     . . . .
}
```

AAA는 public 클래스이므로 어디서든 인스턴스 생성 가능!

■ 생성자는 public인데, 클래스는 default?

나 정말 JAVA를 공부한 적이 없다구요

```
public class AAA
{
    AAA(){…}
    . . . .
}
```

클래스는 public으로 선언되어서 파일을 대표하는 상황! 그럼에도 불구하고 생성자가 default로 선언되어서 동일 패키지 내에서만 인스턴스 생성을 허용하는 상황!

```
class BBB
{
    public BBB(){…}
    . . . .
}
```

생성자가 public임에도 클래스가 default로 선언되어서 동일 패키니 내에서만 인스턴스 생성이 허용되는 상황!

디폴트 생성자

나 정말 JAVA를 공부한 적이 없다구요

디폴트 생성자의 접근제어 지시자는 클래스의 선언형태에 따라서 결정된다.

```
public class AAA
{
    public AAA() {…}
     . . . .
}
```

public 클래스에 디폴트로 삽입되는 생성자

```
class BBB
{
    BBB() {…}
     . . . .
}
```

default 클래스에 디폴트로 삽입되는 생성자

나는 정말 JAVA를 공부한 적이 없다고

09-4. 어떤 클래스를 pubilc으로 선언할까요?

■ 파일당 하나의 외부 제공 클래스 정의하기

외부에서는 Calculaor 클래스의 존재만 알면 된다. Adder와 Substractor 클래스의 존재는 알 필요 없다.
그리고 이렇게 외부에 노출시킬 클래스를 public으로 선언한다.

```
public class Calculator
{
    private Adder adder;
    private Subtractor subtractor;

    public Calculator()
    {
        adder = new Adder();
        subtractor = new Subtractor();
    }
    public int addTwoNumber(int num1, int num2)
    {
        return adder.addTwoNumber(num1, num2);
    }
    public int subTwoNumber(int num1, int num2)
    {
        return subtractor.subTwoNumber(num1, num2);
    }
    public void showOperatingTimes()
    {
        System.out.println("덧셈 횟수 : " + adder.getCntAdd());
        System.out.println("뺄셈 횟수 : " + subtractor.getCntSub());
    }
}
```

```
class Adder
{
    private int cntAdd;

    Adder() { cntAdd=0; }
    int getCntAdd() { return cntAdd; }
    int addTwoNumber(int num1, int num2)
    {
        cntAdd++;
        return num1 + num2;
    }
}
```

```
class Subtractor
{
    private int cntSub;

    Subtractor() { cntSub=0; }
    int getCntSub() { return cntSub; }
    int subTwoNumber(int num1, int num2)
    {
        cntSub++;
        return num1 - num2;
    }
}
```

위의 세 클래스는 하나의 소스파일 Calculator.java에 정의되어 있다.

■ 클래스를 하나로 묶으면 안될까요?

나 정말 JAVA를 공부한 적 없는데요

계산기 기능의 완성을 위해서 Calculator 클래스 이외에 Adder, Substractor 클래스를 별도로 구분할 필요가 있는가?

질문에 대한 답변!

- 변경이 있을 때, 변경되는 클래스의 범위를 줄일 수 있다.
- 작은 크기의 클래스를 다른 클래스의 정의에 활용할 수 있다.

객체지향에서는 아주 큰 하나의 클래스보다, 아주 작은 열 개의 클래스가 더 큰 힘과 위력을 발휘한다!

나는 정말 JAVA를 공부한 적이 없다

09-5. 캡슐화(Encapsulation)

■ 콘택600에 숨겨져 있는 캡슐화

나는 정말 JAVA를 공부한 적이 없다

```
class SinivelCap    // 콧물 처치용 캡슐
{
    public void take(){  .  .  .  .  }
}
class SneezeCap     // 재채기 처치용 캡슐
{
    public void take() { .  .  .  . }
}
class SnuffleCap    // 코막힘 처치용 캡슐
{
    public void take() { .  .  .  . }
}
```

세 개의 클래스가 하나의 목적인 '콧물 감기의 치료'라는 일치된 목적을 갖고 있다. 그럼에도 불구하고 클래스가 나뉘어 있다.

SinivelCap, SneezeCap, SnuffleCap의 연관관계가 깊다면 캡슐화가 이뤄지지 않은 상태

```
public static void main(String[] args)
{
    ColdPatient sufferer = new ColdPatient();
    sufferer.takeSinivelCap(new SinivelCap());
    sufferer.takeSneezeCap(new SneezeCap());
    sufferer.takeSnuffleCap(new SnuffleCap());
}
```

콘택600은 콧물감기에 대한 적절한 캡슐화의 모델!

약의 복용순서가 정해져 있다면? 약을 복용하는 사람은 약의 복용과 관련해서 추가적인 지식이 필요하다. 캡슐화가 이뤄지지 않으면, 클래스의 사용을 위해서 알아야 할 것들이 많아진다.

■ 캡슐화가 이뤄져 있는 콘택600을 표현하면

나 정말 JAVA를 공부한 적이 없다면

```
class CONTAC600
{
    SinivelCap sin;
    SneezeCap sne;
    SnuffleCap snu;

    public CONTAC600()
    {
        sin=new SinivelCap();
        sne=new SneezeCap();
        snu=new SnuffleCap();
    }
    public void take()
    {
        sin.take();
        sne.take();
        snu.take();
    }
}
```

캡슐화는 관련 있는 모든 메소드와 변수를 하나의 클래스로 묶는 것!

둘 이상의 클래스를 묶어서 캡슐화를 완성할 수도 있다. 캡슐화는 메소드와 변수가 코드레벨에서 묶이는 것을 의미하지 않는다. 캡슐화는 개념적인 의미의 묶음을 의미한다.

take 메소드 내에 약의 복용순서가 그대로 기록되어 있다. 따라서 약의 복용을 위해 알아야 할 것이 take 메소드 하나이다! 이것이 캡슐화의 목적 및 장점이다.

캡슐화는 객체지향에서 실제 프로젝트에 적용하기 쉽지 않은 개념 중 하나이다. 그리고 캡슐화에는 정답이 없어서 더 어렵게 생각될 수 있다. 다음 질문에 답해 보겠는가?

그럼, 제대로 캡슐화하려면 기침, 몸살, 두통까지 넣어야 하지 않나요?

나는 정말 JAVA를 공부한 적이 없다