10. 메서드 재정의(overriding)

오버라이딩은 **상속 관계**에 있는 클래스들 사이에 같은 이름의 메서드를 재정의하는 것이다.
오버라이딩은 부모클래스와 자식클래스에 있는 메서드의 이름이 같은 경우이다.
오버라이딩에서는 **메서드 이름, 매개변수 개수 및 자료형, 반환형**이 같아야 한다.

```
class A
{
  int compute(int a, int b){ return a + b; }
}
class B extends A
{
  int compute(int a, int b){ return a * b; }    //메서드 재정의
}
class C extends B
{
  int compute(int a, int b){ return a - b; }    //메서드 재정의
}
class OverrideDemo
{
  public static void main(String args[]){
    A ride1 = new A();
    B ride2 = new B();
    C ride3 = new C();
                                                  [실행 결과]
    System.out.println(ride1.compute(2, 3));        5
    System.out.println(ride2.compute(2, 3));        6
    System.out.println(ride3.compute(2, 3));       -1
  }
}
```

[Tip] OOP 언어에서 오버로딩과 오버라이딩(다형성 구현)

- Overloading : 동일한 클래스 내에서 같은 이름의 메서드를 정의하여 다형성 지원
- Overriding : 상속 관계의 클래스에서 같은 이름의 메서드를 정의하여 다형성 지원

//탐구 - 자바에서 오버로딩이냐? 오버라이딩이냐?

```
class A{
  int p = 1;
  int q = 2;
  int abc(int k){ return k + 1; }
}
class B extends A{
  int x = 100;
  int y = 200;
  int abc(int r, int s){   //여기서, 메서드 abc()는 별도로 취급됨(재정의된 것이 아님)
      return r + s;
  }
}
public class Test{
    public static void main(String args[]){
      A a = new A();
      B b = new B();

      a = (A)b; //상속 관계에 있을 때만 가능     [실행결과]
      System.out.println(a.p);                  // 1
      System.out.println(a.q);                  // 2
      System.out.println(b.p);                  // 1
      System.out.println(b.q);                  // 2
      System.out.println(b.x);                  // 100
      System.out.println(b.y);                  // 200
      System.out.println(a.abc(1000));          // 1001
      System.out.println(b.abc(2000));          // 2001 → 상속받은 것이 처리됨
      System.out.println(b.abc(3000, 4000));    // 7000      (재정의 아님)
   }
}
```

◈ **자바에서 메서드 재정의(Overriding)**

① 메서드 이름이 같아도 인자목록과 반환형이 일치하지 않으면 재정의로 인정되지 않는다.
→ 새로운 메서드로 취급할 뿐이다.

② 메서드 이름과 인자목록은 같고 반환형이 다르면
두 메서드를 구별할 수 없어서 컴파일 오류가 발생된다.

③ 결론은 **"메서드 이름, 인자목록, 반환형"**이 같아야 재정의로 인정된다.

Tip	만약, 클래스 B에 다음 형태의 메서드가 있으면 재정의가 아니다.(오류 발생) `long abc(int k){ return k + 3; }` → **반환형이 다르므로 오류 발생**

기출문제 분석

1. 객체지향언어에서 클래스 A와 클래스 B는 상속관계에 있다. A는 부모클래스, B는 자식클래스라고 할 때 클래스 A에서 정의된 메서드(method)와 원형이 동일한 메서드를 클래스 B에서 기능을 추가하거나 변경하여 다시 정의하는 것을 무엇이라고 하는가? [2014년 국가 9급]

① 추상클래스(abstract class)
② 인터페이스(interface)
③ 오버로딩(overloading)
④ 오버라이딩(overriding)

♤ 오버라이딩(overriding, 재정의)

```
class A
{
   int don = 100;
   int get() { return don + 1; }
}

class B extends A                    //상속
{
   int don = 200;
   int get() { return don + 2;}      //재정의 - 다형성 지원 요인
}

class Test
{
   public static void main(String args[])
   {
     A p = new B();                  //명령어 new가 객체(p)를 생성한다.
     System.out.println(p.don);      //100 출력 - 속성 호출
     System.out.println(p.get());    //202 출력 - 재정의된 메서드 호출
   }
}
```

정답 : ④

2. 다음 설명과 가장 관련 있는 객체지향 기법의 원칙은? [2009년 국가 7급]

라인 그래프, 파이 차트, 히스토그램, 키비아트 다이어그램의 4가지 그래프를 그려야 하는 응용 프로그램이 있다고 하자. 이 프로그램을 설계하기 위해 일반적인 클래스로 graph라는 클래스를 정의하고 각각의 그래프를 나타내는 클래스들을 graph 클래스의 서브클래스로 정의한다. 그리고 graph 클래스와 각각의 서브클래스에 그래프를 그리는 draw라는 메서드를 정의한다.
그러면 객체는 이들 서브클래스들의 인스턴스인 객체를 어떤 것에도 draw라는 메시지를 보낼 수 있고 메시지를 받은 객체는 적절한 그래프를 생성하기 위해 자기 자신의 draw 메서드를 호출한다.

---

① 캡슐화(encapsulation)
② 추상화(abstraction)
③ 다형성(polymorphism)
④ 정보은닉(information hiding)

♤ 다형성 – 재정의(overriding)

• 주어진 내용을 Java로 구현하면 다음과 같다.

```
class Graph { void draw(){ System.out.println("무슨 그래프?"); } }

class Line extends Graph{ void draw(){ System.out.println("라인"); } }          //재정의
class Pie extends Graph { void draw(){ System.out.println("파이"); } }           //재정의
class Histo extends Graph { void draw(){ System.out.println("히스토그램"); } }  //재정의
class Kebee extends Graph { void draw(){ System.out.println("키비아트"); } }    //재정의

class Test
{
   public static void main(String args[]){
     Graph g1 = new Line();    g1.draw();    //라인
     Graph g2 = new Pie();     g2.draw();    //파이
     Graph g3 = new Histo();   g3.draw();    //히스토그램
     Graph g4 = new Kebee();   g4.draw();    //키비아트
  }
}
```

정답 : ③

3. 다음 자바코드에 나타난 것과 같이 동일한 이름의 메시지로 다른 구현을 호출할 수 있는 객체 지향 개념은? [2021년 국가 7급]

```
Animal a;
a = new Dog( );
a.makeSound( );     // "멍멍" 출력함
a = new Cat( );
a.makeSound( );     // "야옹" 출력함
```

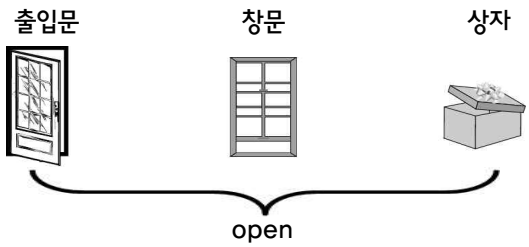
① 다형성 ② 정보은닉 ③ 정적바인딩 ④ 캡슐화

♤ 객체지향언어에서 오버로딩과 오버라이딩(다형성 구현)

- Overloading : 동일한 클래스 내에서 같은 이름의 메서드를 정의하여 다형성 지원
- Overriding : 상속 관계의 클래스에서 같은 이름의 메서드를 정의하여 다형성 지원

정답 : ①

4. <보기>에서 설명하는 객체지향 개념은? [2012년 계리] [2015년 국가 9급]



- 그림에서 'open'이라는 오퍼레이션(operation)은 객체마다 다르게 기능한다.
- Java 언어에서 오버로딩(overloading), 오버라이딩(overriding)으로 구현되는 개념이다.

① 캡슐화(encapsulation) ② 인스턴스(instance)
③ 다형성(polymorphism) ④ 상속(inheritance)

♤ 다형성(polymorphism) ≠ 단형성(monomorphism)

- 다형성은 메서드, 연산자, 상수, 변수 등이 다양한 형태에 적용되는 성질이다.

출입문을 열어라 / 창문을 열어라 / 상자를 열어라

정답 : ③

5. 객체지향에서 다형성(polymorphism)의 주요 설명으로 옳지 않은 것은? [2020년 국가 7급]

① 속성과 관련된 오퍼레이션을 클래스 안에 묶어서 하나로 취급한다.
② 하나의 인터페이스에 메서드를 데이터 타입(data type) 및 파라미터 수를 변경하여 재정의가 가능하다.
③ 하나의 인터페이스를 일관성 있게 사용 중심에서 제공할 수 있다.
④ 오버로딩(overloading), 오버라이딩(overriding)을 이용한 재사용성을 높일 수 있다.

♤ **다형성(polymorphism)**

- 속성과 관련된 오퍼레이션을 클래스 안에 묶어서 하나로 취급한다.**(×) → 캡슐화**
- 다형성의 사전적 의미는 '여러 개의 형태를 가진다'라는 뜻이다.
- 여러 형태를 가지므로 사용자는 사용편의성을 취할 수 있다.

// OOP 언어에서 오버로딩과 오버라이딩(다형성 구현)

- Overloading : 동일한 클래스 내에서 같은 이름의 메서드를 정의하여 다형성 지원
- Overriding : 상속 관계의 클래스에서 같은 이름의 메서드를 정의하여 다형성 지원

정답 : ①

6. 객체지향 프로그래밍의 특징 중 상속 관계에서 상위클래스에 정의된 메서드(method) 호출에 대해 각 하위클래스가 가지고 있는 고유한 방법으로 응답할 수 있도록 유연성을 제공하는 것은? [2015년 국가 9급]

① 재사용성(reusability)
② 추상화(abstraction)
③ 다형성(polymorphism)
④ 캡슐화(encapsulation)

♤ **객체지향에서 다형성**

- 다형성은 오버로딩(overloading)이나 오버라이딩(overriding)으로 구현할 수 있다.
- Overloading : 동일한 클래스 내에서 같은 이름의 메서드를 정의하여 다형성 지원
- Overriding : 상속 관계의 클래스에서 같은 이름의 메서드를 정의하여 다형성 지원

정답 : ③