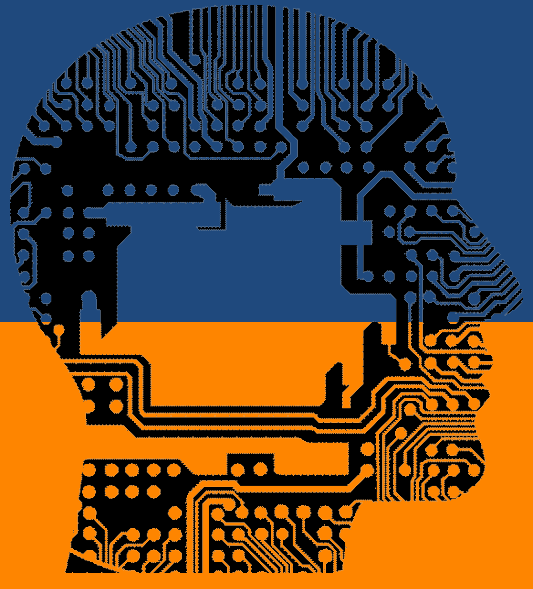
한국IT진흥부설
정보보호교육학원 아이섹

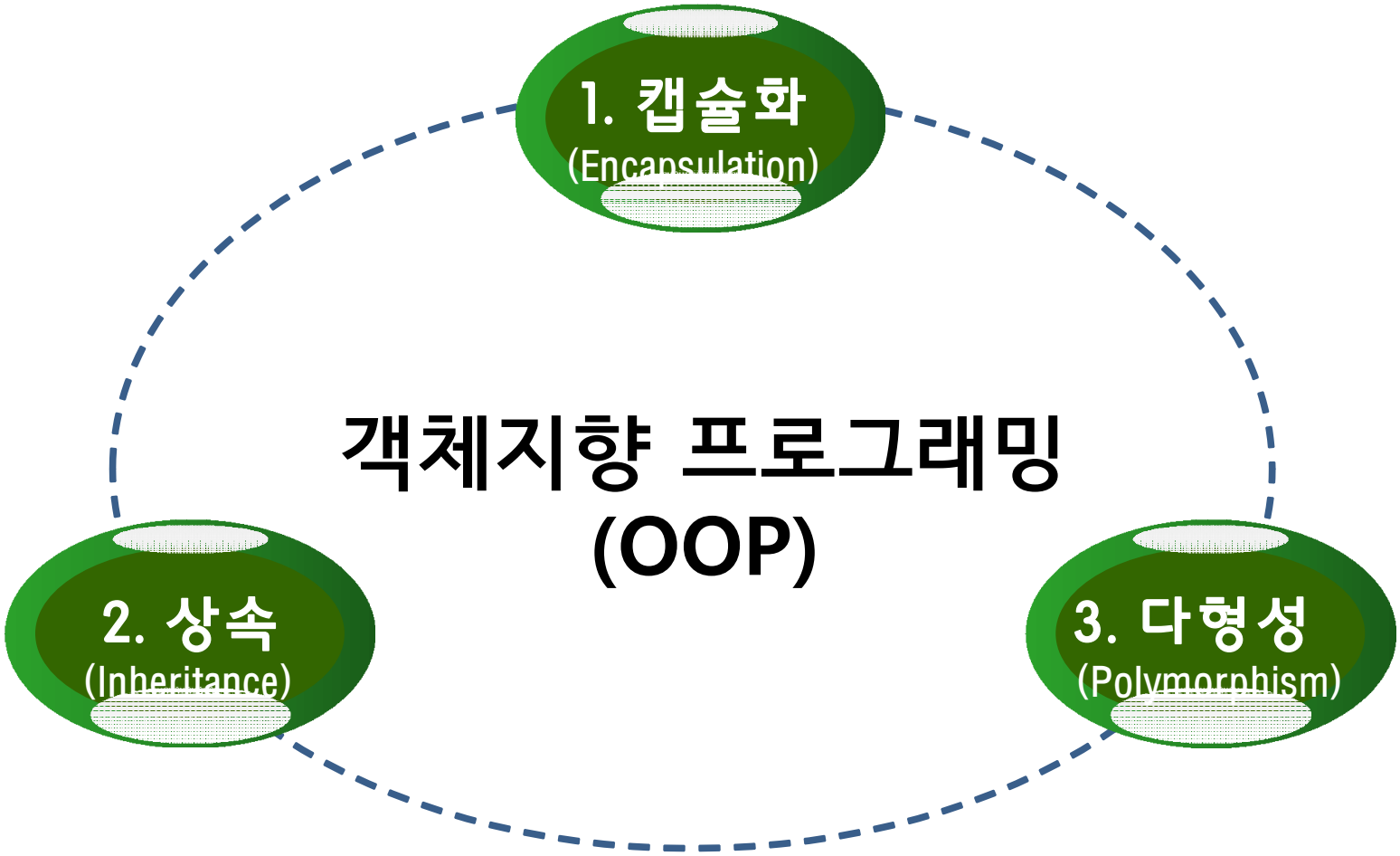


자바 프로그래밍 기초

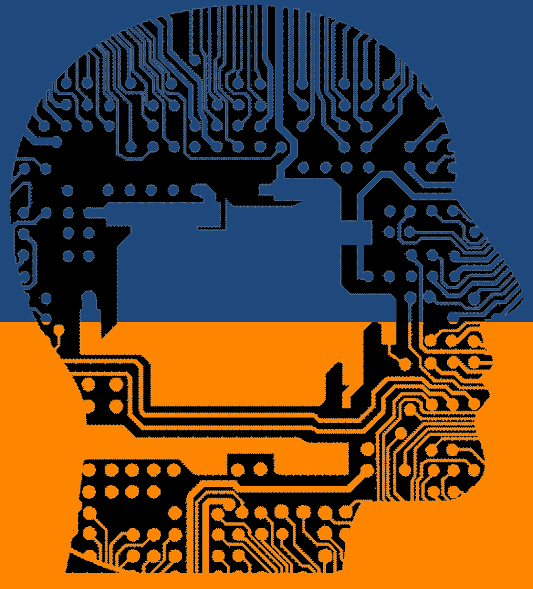
다형성(polymorphism)

By SoonGu Hong(Kokono)

* 객체지향 프로그래밍 특징



한국IT진흥부설
정보보호교육학원 아이섹

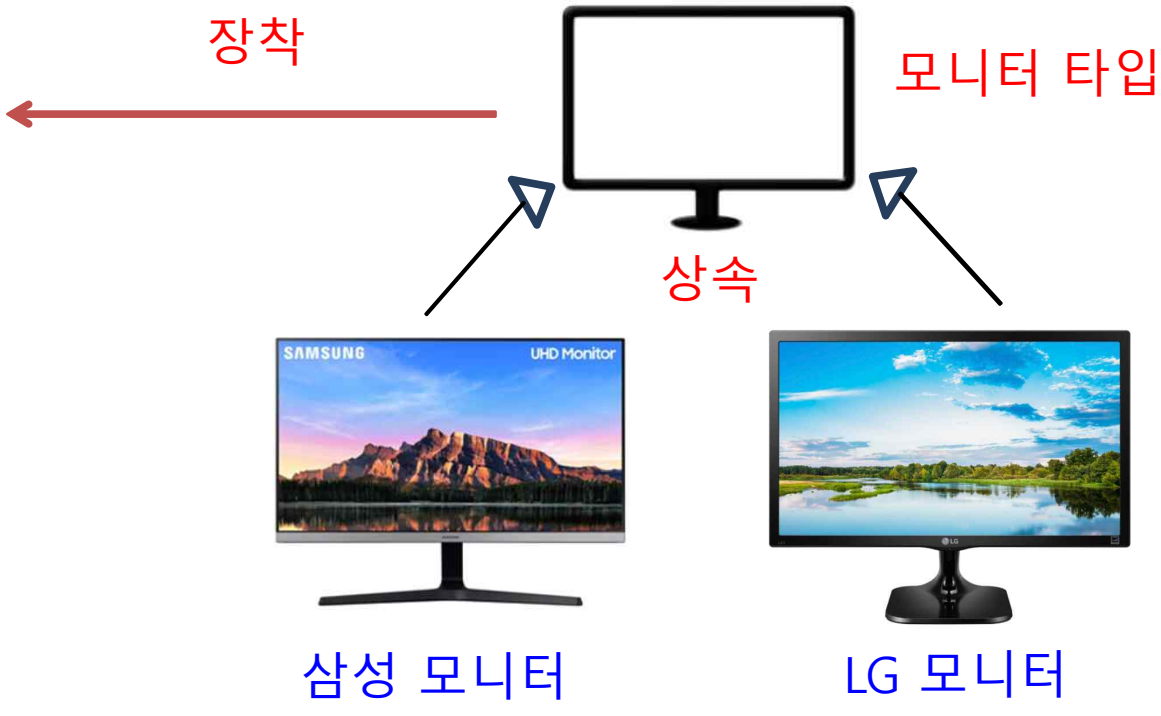


자바 프로그래밍 기초

1. 타입변환과 다형성

1-1. 다형성이란?

- 다형성이란 객체들이 여러가지 형태, 즉 타입을 이용할 수 있는 성질을 말합니다.
- 자바는 다형성을 위해 객체들의 부모타입변환을 허용하고 있습니다. 즉, 부모 타입에 모든 자식 객체를 대입할 수 있습니다.
- 다형성을 이용하면 객체의 부품화를 할 수 있습니다.



1-2. 다형성의 이점 - 객체의 교환성 증가

```
class Computer {
    LGMonitor monitor;
}

class LGMonitor {}
class HPMonitor {}
```

```
Computer com = new Computer();
com.monitor = new LGMonitor; // (O)
com.monitor = new HPMonitor(); // (X)
```

```
class Computer {
    Monitor monitor;
}

class Monitor {}
class LGMonitor
        extends Monitor {}
class HPMonitor
        extends Monitor {}
```

```
Computer com = new Computer();
com.monitor = new LGMonitor; // (O)
com.monitor = new HPMonitor(); // (O)
```

- 필드에 다형성을 적용한 결과 다양한 모니터객체를 교환할 수 있게 되었다!

1-3. 다형성의 이점 - 이종모음 배열 구성이 가능

```
class Album {}
class DVD {}
class Book {}
```

```
Album a = new Album();
DVD d = new DVD();
Book b = new Book();
```

- 배열을 배울 때 배열은 동종모음 구조라고 하였습니다. 즉, 같은 타입의 데이터만 배열에 저장할 수 있죠. 그러면 위의 3개의 객체를 하나의 배열에 담을 수 있을까요??

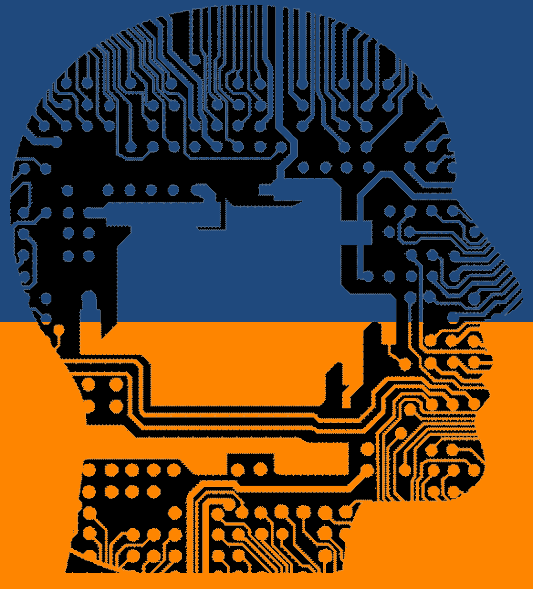
```
class Item {}

class Album
    extends Item {}
class DVD
    extends Item {}
class Book
    extends Item {}
```

```
Item a = new Album();
Item d = new DVD();
Item b = new Book();
```

- 다형성을 통해 타입을 부모타입인 Item으로 통일했습니다. 배열은 동종모음 구조라고 했던 것 기억나시나요? 이제 3개의 객체가 같은 타입이 되었으니 배열도 가능하겠죠?

```
Item[] items = {a, d, b};
```

한국IT진흥부설
정보보호교육학원 아이섹



자바 프로그래밍 기초

2. 메서드 오버로딩과 매개변수의 다형성

2-1. 오버로딩이란?

- 하나의 클래스 내에서 같은 이름의 메서드를 여러 개 선언하는 것을 메서드 오버로딩(overloading)이라고 합니다.

```
class A {

    리턴 타입   메서드이름 ( 매개변수, ... ) {}

    리턴 타입   메서드이름 ( 매개변수, ... ) {}

}
```

무관 / 동일 / 타입, 개수, 순서가 달라야함

2-2. 매개변수의 다형성

```
class Car {}
class Sonata
    extends Car {}
class Tucson
    extends Car {}
```

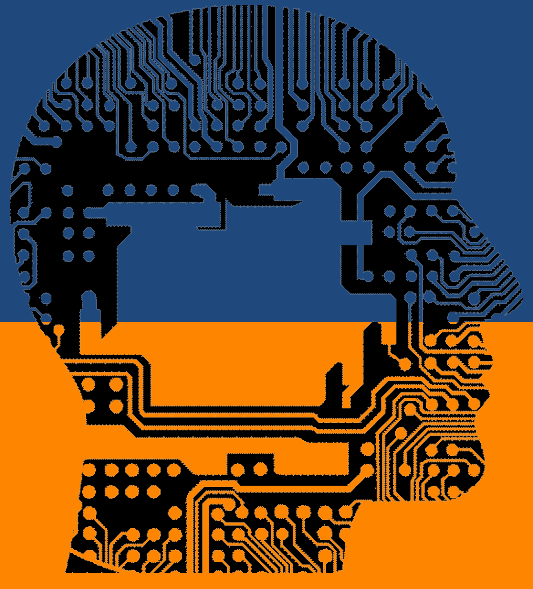
```
class Driver {
  public void drive(Sonata s) {}

  public void drive(Tucson t) {}
}
```

- Driver객체가 운전을 수행하는 drive메서드가 오버로딩에 의해 운전할 수 있는 차량을 매개변수로 선언했을 때 만약 차종이 많다면 오버로딩의 개수가 늘어나 비효율적이겠죠??

```
class Driver {
  public void drive(Car c) {}
}
```

- 이런 경우 매개 변수에도 다형성을 적용하여 부모타입인 Car타입으로 자식 객체인 여러 자동차들을 처리할 수 있습니다.

한국IT진흥부설
정보보호교육학원 아이섹



자바 프로그래밍 기초

3. 강제 타입 변환

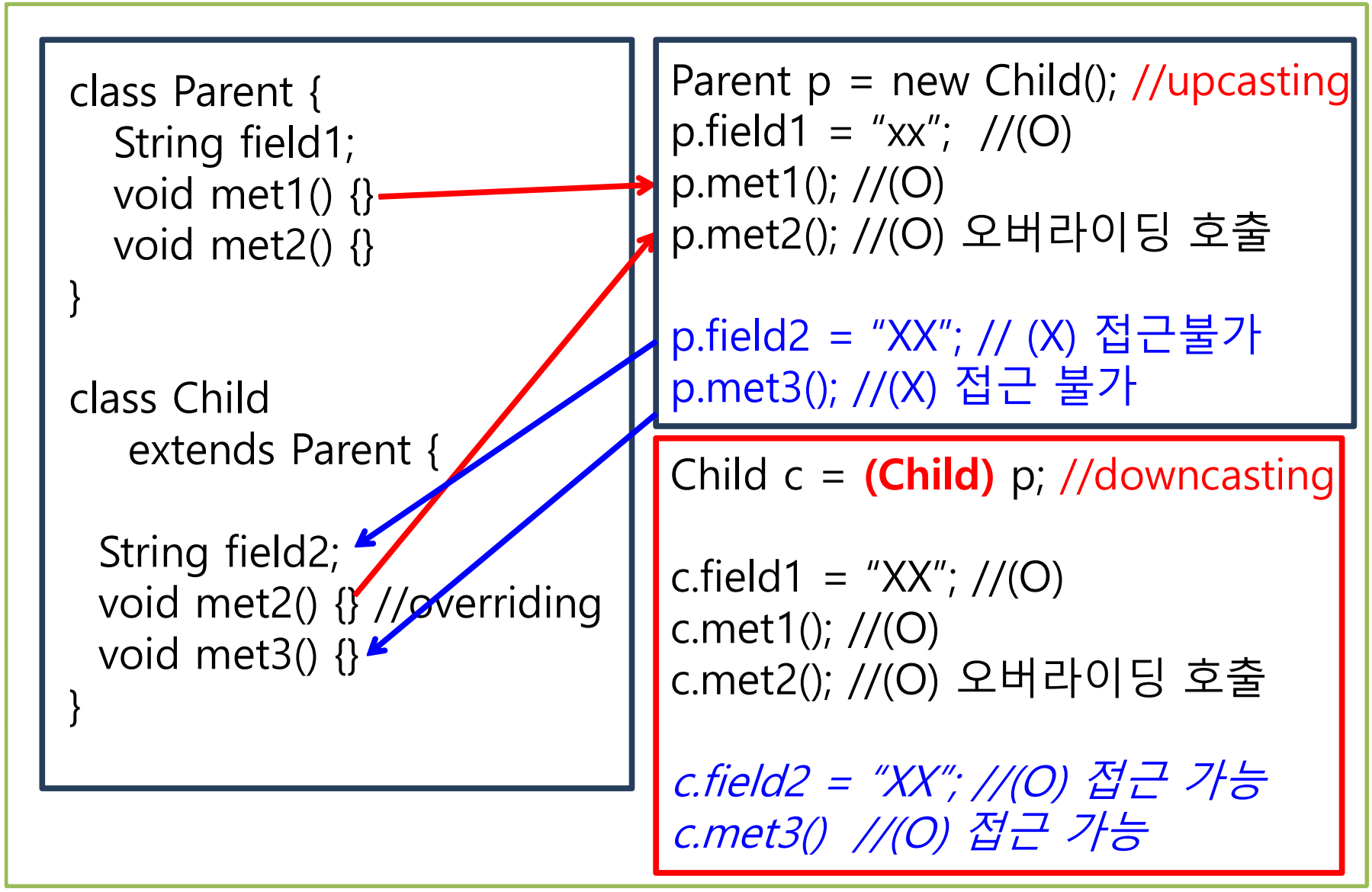
3-1. 강제 타입 변환(Casting)

- 강제 타입 변환은 부모 타입을 자식 타입으로 변환하는 것을 말합니다. 그러나 상속관계에서 모든 부모 타입을 자식 타입으로 바꿀 수 있는 것은 아닙니다.
- 반드시 **자식타입이 부모타입으로 한번 변환된 객체**에 대해서만 강제 타입 변환을 사용할 수 있습니다.

이걸 왜 하는건데???

- 다형성 적용을 위해 자식객체가 부모타입을 갖게 되면 부모클래스에서 선언된 필드와 메서드, 그리고 자식클래스가 오버라이딩한 메서드만 사용할 수 있는 **제약사항**이 생깁니다.
- 이 문제를 해결하기 위해선 다시 자식타입으로 돌려놓는 강제 타입 변환이 꼭 필요합니다.

3-2. 강제 타입 변환 예시


class Parent {
String field1;
void met1() {}
void met2() {}
}
class Child
extends Parent {
String field2;
void met2() {} //overriding
void met3() {}
}
Parent p = new Child(); //upcasting
p.field1 = "xx"; //(O)
p.met1(); //(O)
p.met2(); //(O) 오버라이딩 호출
p.field2 = "XX"; // (X) 접근불가
p.met3(); //(X) 접근 불가
Child c = (Child) p; //downcasting
c.field1 = "XX"; //(O)
c.met1(); //(O)
c.met2(); //(O) 오버라이딩 호출
c.field2 = "XX"; //(O) 접근 가능
c.met3() //(O) 접근 가능

3-3. 객체 타입 체크 연산자 - instanceof

- 강제 타입 변환의 전제조건은 자식 타입이 부모 타입으로 변환된 경우에만 가능하다는 것인데, 이 조건을 만족하지 않은 상태로 강제 타입변환을 시도하면 에러가 발생합니다

```
Parent p = new Parent();
Child c = (Child) p;  // 에러 발생!
```

- 그러면 부모타입을 가지고 있는 객체가 자식 객체인지 확인하는 방법은 없을까요? 이 때 바로 instanceof 연산자를 사용합니다.
- instanceof연산자는 좌항에 객체를 우항에 타입을 지정하고 좌항의 객체가 우항의 타입인지를 체크하여 맞으면 true, 틀리면 false를 리턴합니다.

```
boolean result = p instanceof Child;
System.out.println(result); => false
```

감사합니다
THANK YOU