Chapter 14-1

다형성

다형성

다형성(Polymorphism)

- 다형성이란 "객체가 여러 형태를 가진다"라는 의미로 해석되며, 하나의 객체가 여러가지 유형으로 사용되는 것을 의미합니다.

- 다형성은 상속을 전제조건으로 합니다.

- 다형성을 위해 자바는 자식클래스가 부모클래스의 타입을 가질 수 있도록 허용합니다. 즉, 부모 타입에 모든 자식객체가 대입될 수 있습니다.

부모클래스 변수 = new 자식클래스()

이때.
부모클래스에 있는 기능만 사용할 수 있지만,
오버라이딩 된 메서드는 먼저 실행됩니다

다형성 basic

polymorphism/PolymorphismExample.java

```
1:  package polymorphism;
2:
3:  public class PolymorphismExample {
4:     public static void main(String[] args) {
5:        Person p;
6:
7:        p = new Student("허현수", 17, "20160001");
8:        System.out.println(p.getDetails());
9:
10:       p = new Teacher("허현준", 22, "Java Programming");
11:       System.out.println(p.getDetails());
12:
13:       p = new Employee("허현정", 23, "교무처");
14:       System.out.println(p.getDetails());
15:    }
16: }
```

Student, Teacher, Employee(자식클래스)로 생성해서 Person(부모) 타입에 저장 할 수 있다

```
Problems  Javadoc  Declaration  Console
<terminated> PolymorphismExample [Java Application] C:₩Program Files₩Ja
이름: 허현수        나이: 17 학번: 20160001
이름: 허현준        나이: 22 과목: Java Programming
이름: 허현정        나이: 23 부서: 교무처
```

이 때, 부모에 있는 메서드만 실행가능 하며 overriding메서드는 우선 실행된다

클래스 Casting

강제 타입 변환(Type Casting)

- 강제 타입 변환은 부모 타입을 자식 타입으로 변환하는 것을 말합니다.

- 객체에서 타입 캐스팅을 사용하려면 우선 먼저 Promotion이 일어나야 합니다. 즉, **부모 타입으로 한번 형 변환이 된 자식 객체만 강제 타입 변환을 사용**할 수 있습니다.

- Promotion이 일어나면 자식 클래스가 가지고 있는 **재정의되지 않은 메서드를 사용할 수 없다는 단점**이 있었습니다. 이 단점을 극복하기 위해 강제 타입 변환을 사용하여 자식 메서드를 호출하는 방법을 사용합니다.

```
public class MainClass {

	public static void main(String[] args) {

	Parent p = new Child();

	Child c = (Child)new Parent();
	c.method3();

	}
}
```

클래스 타입의 형 변환이 가능하다
다시 Child 타입의 기능 사용이 가능해 진다.

이종 모음 도 다형성의 원리이다

이종모음(Heterogeneous Collection)

- 이종모음은 배열에 다형성을 적용시키는 원리입니다. 배열은 원래 동종모음 구조입니다.

- 예를 들어 int[] iArr = new int[10]; 이런 구조의 배열이 있다면 iArr배열에는 int형 정수 데이터만 저장할 수 있습니다.
- 하지만 다형성을 이용하면 이종모음 구조의 객체 배열이 생성 가능합니다.

polymorphism/HeteroCollectionExample.java

```
1:  package polymorphism;
2:
3:  public class HeteroCollectionExample {
4:     public static void main(String[] args) {
5:        Person[] pArr = new Person[4];
6:
7:        pArr[0] = new Person("홍길동", 20);
8:        pArr[1] = new Student("허현수", 17, "20160001");
9:        pArr[2] = new Teacher("허현준", 22, "Java Programming");
10:       pArr[3] = new Employee("허현정", 23, "교무처");
11:
12:       for(int i=0; i<pArr.length i++) {
13:          System.out.println( pArr[i].getDetails() );
14:       }
15:    }
16: }
```

부모 타입 배열에 자식 객체가 저장 될 수 있다.

```
Console
<terminated> HeteroCollectionExample [Java Application] C:₩Program Files₩
이름: 홍길동        나이: 20
이름: 허현수        나이: 17 학번: 20160001
이름: 허현준        나이: 22 과목: Java Programming
이름: 허현정        나이: 23 과목: 교무처
```

매개변수의 다형성

매개 변수의 다형성

- promotion은 멤버 변수의 값을 대입할 때도 발생하지만, 메서드를 호출할 때 사용하는 매개 변수에도 발생할 수 있습니다.

- 보통 메서드를 호출할 때는 메서드 선언부에서 지정한 데이터 타입과 일치하는 매개값을 전달하여 호출하지만, 매개 변수에 다형성을 적용하면 자식 객체를 전달할 수도 있습니다.

polymorphism/PolyArgumentExample.java

```
1:  package polymorphism;
2:
3:  public class PolyArgumentExample {
4:     public static void main(String[] args) {
5:        Student s = new Student("허현수", 17, "20001234");
6:        printPersonInfo(s);
7:
8:        Teacher t = new Teacher("허현준", 22, "Java Programming");
9:        printPersonInfo(t);
10:
11:       Employee e = new Employee("허현정", 23, "교무처");
12:       printPersonInfo(e);
13:    }
14:
15:    public static void printPersonInfo(Person p) {
16:       System.out.println("*************** Person Info ***************");
17:       System.out.println(p);
18:       System.out.println(p.getDetails());
19:       System.out.println("*******************************************\n");
20:    }
21: }
```

매개변수에 객체를 전달하려면
타입을 클래스 타입을 적어주면 된다

이 때 Person의 자식 클래스는
전부 전달 될 수 있다

클래스 타입 확인! instanceof 키워드

instanceof

- 자바의 키워드 중에서 instanceof 는 객체가 지정한 클래스의 인스턴스인지 아닌지 검사할 때 사용하는 연산자입니다.

- instanceof 연산자의 왼쪽 항의 객체가 오른쪽 항 클래스의 인스턴스 즉, 오른쪽 항의 객체가 생성되었다면 true를 리턴하고, 그렇지 않으면 false를 리턴합니다.

polymorphism/InstanceofExample.java

```
1:  package polymorphism;
2:
3:  public class InstanceofExample {
4:      public static void main(String[] args) {
5:          Student s = new Student("허현수", 17, "20001234");
6:          printPersonInfo(s);
7:
8:          Teacher t = new Teacher("허현준", 22, "Java Programming");
9:          printPersonInfo(t);
10:
11:         Employee e = new Employee("허현정", 23, "교무처");
12:         printPersonInfo(e);
13:     }
14:
15:     public static void printPersonInfo(Person p) {
16:         if(p instanceof Student) {
17:             System.out.println("*************** Student Info **************");
18:         }else if(p instanceof Teacher) {
19:             System.out.println("*************** Teacher Info **************");
20:         }else if(p instanceof Employee) {
21:             System.out.println("*************** Employee Info **************");
22:         }else {
23:             System.out.println("*************** Person Info ***************");
24:         }
25:         System.out.println(p);
26:         System.out.println(p.getDetails());
27:         System.out.println("*****************************************\n");
28:     }
29: }
```

- ex) Person p = new Student();

  p instanceof Student -> true

Chapter 14

수고하셨습니다