Chap 08. Class Design – Polymorphism

학습목표

1. Polymorphism
   - 개념
   - Virtual Method Invocation
   - Why Polymorphism
   - instanceof keyword
2. Object Class
3. Wrapper Class

OOP (Object-Oriented Programming) 특징

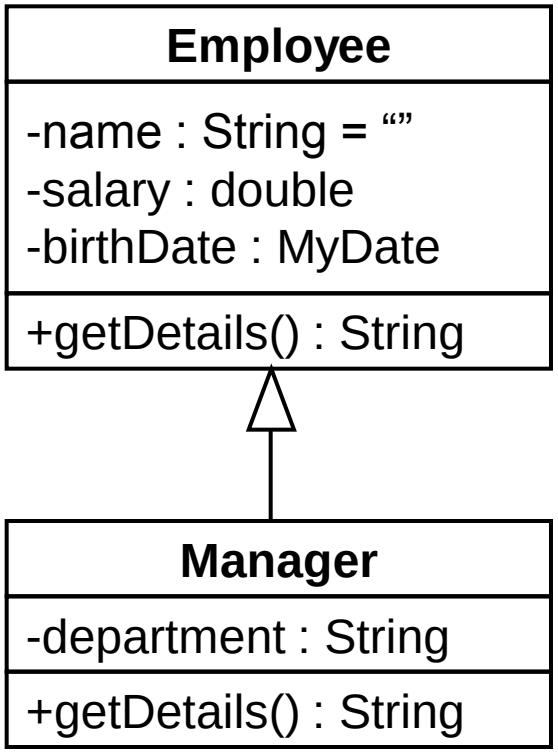
Encapsulation

Inheritance

Polymorphism

Polymorphism – 개념

- Polymorphism(다형성) : 다양한 형태를 가짐을 말한다.
- 자바에서는 한 reference변수가 다른 형태의 객체를 참조 할 수 있음을 말한다.
  (부모 renference 변수 → 자식 객체)
- 즉, reference 변수를 polymorphic 하다고 할 수 있다.

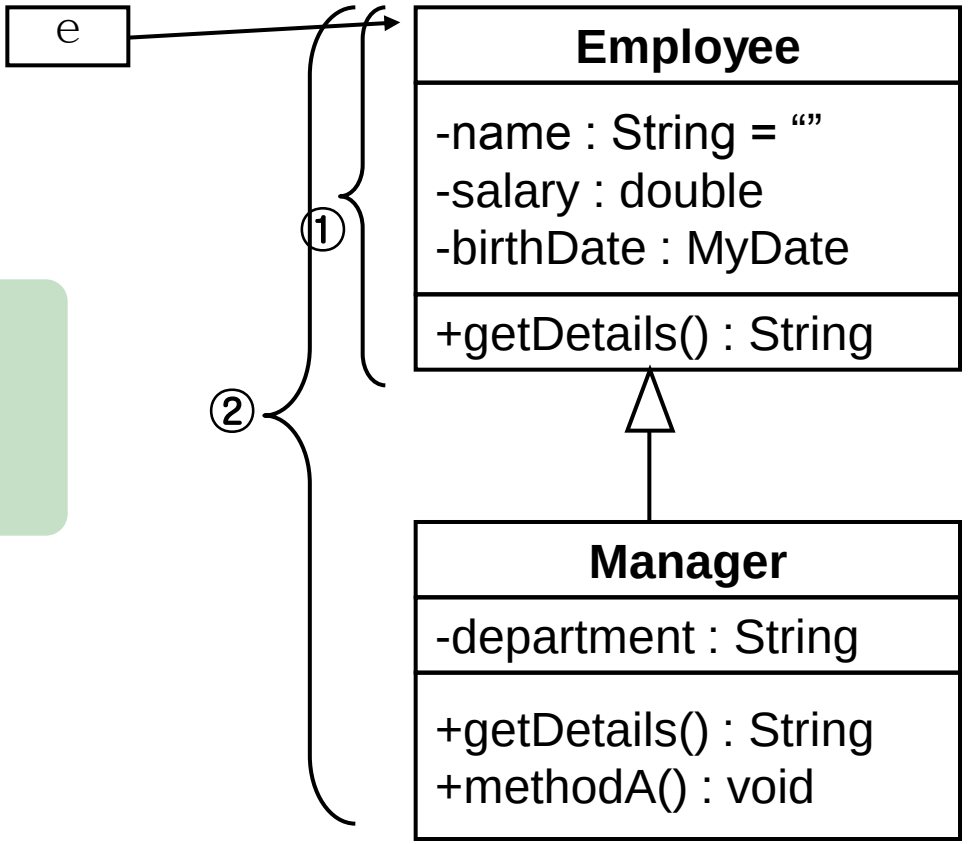



```
Employee e = new Manager(); //OK

Manager m = new Empolyee(); //error
```

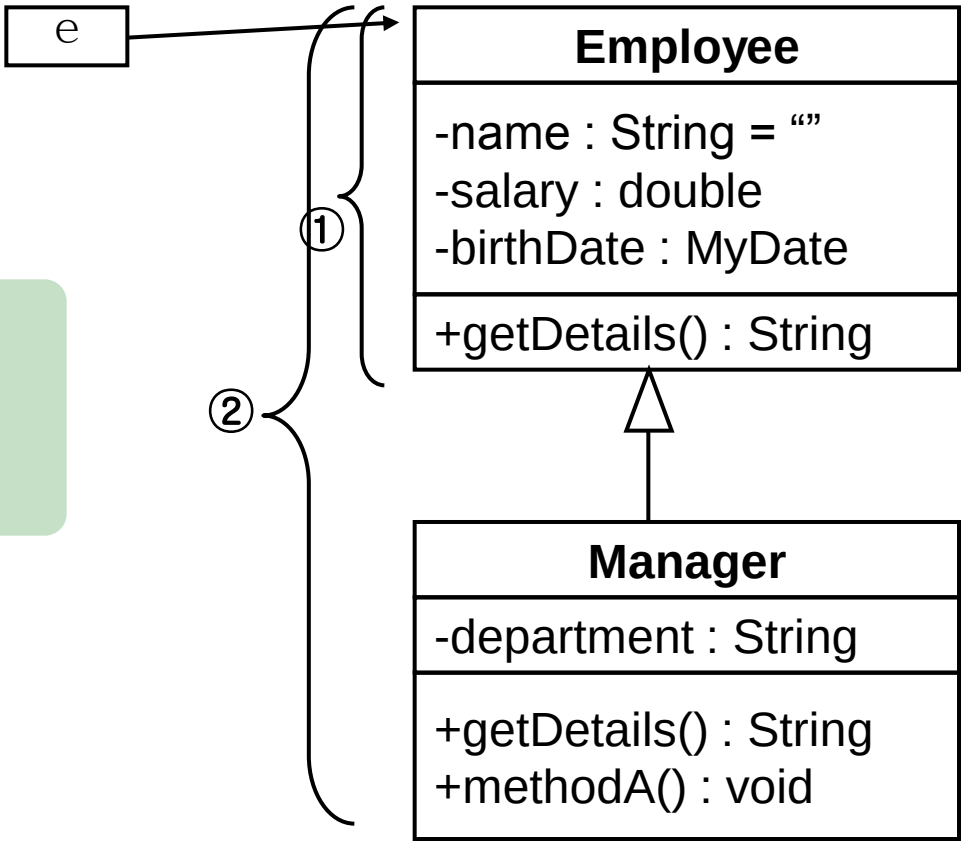
Polymorphism – Virtual Method Invocation


```
Employee e = new Manager();
e.methodA();     // 컴파일 결과는 ?
e.getDetails(); // Employee 의 메소드? Manager 의 메소드?
```



✓ Type-check
① Compile-time type
② Run-time type

- ❑ Compile 시에는 Reference type 만을 가지고, compile 여부 결정
- ❑ 실행 시에는 실제 new를 통해서 생성 되는 객체의 메소드 실행



✓ Type-check
① Compile-time type
② Run-time type

❑ Homogenous collection

동일한 Class의 객체로 이루어진 집합

```
MyDate[] dates = new MyDate[2];
dates[0] = new MyDate( 22, 12, 1964 );
dates[1] = new MyDate( 22, 7, 1964);
```

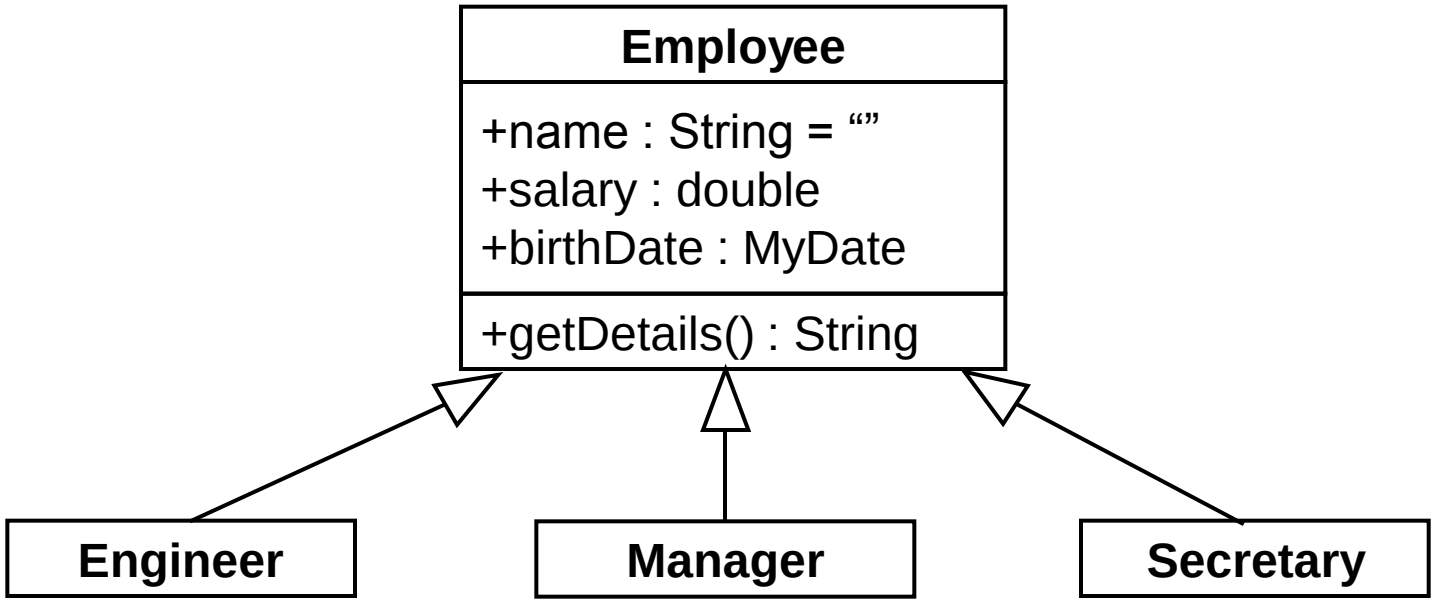
❑ Heterogeneous Collection

다른 Class의 객체로 이루어진 집합

```
Employee[] staff = new Employee[1024];
staff[0] = new Manager();
staff[1] = new Employee();
staff[2] = new Engineer();
```

❑ 사원에 대한 세금을 계산필요 → 메소드로 구현

직군(관리자, 엔지니어, 비서) 에 따라 세금이 달라진다.



✓ 방법 1 : 메소드 오버로딩을 이용하여 각각의 직군에 따른 세금계산 메소드 작성

```
Employee[] staff = new Employee[1024];
```

```
public int getTax( Employee e ) {
    return 100;
}

public int getTax( Manager m ) {
    return 300;
}

public int getTax( Engineer e ) {
    return 200;
}
```

✓ 방법2 : Polymorphism 이용 (Polymorphic Arguments)

```
Employee[] staff = new Employee[1024];
```

```
public double getTax( Employee e ) {

}
```

❑ 문제점 : 부모 reference로 입력 받은 객체가 실제로 어떠한 객체인지 판별 필요

✓ 방법2 : Polymorphism 이용 (Polymorphic Arguments)

```
Employee[] staff = new Employee[1024];
```

```
public double getTax( Employee e ) {
    if ( e instanceof Manager )
    ....
}
```

❑ A instanceof B

- A가 B의 자식이거나 같은 class 타입이면 true
- A가 B의 부모이면 false
- 부모/자식 관계에 속하지 않는 것끼리 instanceof 하면 컴파일 에러

✓ 주의

```
public double getTax( Employee e ) {
    if ( e instanceof Employee )
    ....
}
```

❑ Casting Operator

```
public double getTax( Employee e ) {
    double tax = 0.0;
    if ( e instanceof Manager ) {
        Manager m = (Manager)e;
        tax = m.getSalary() * 0.30;
    } else if ( e instanceof Engineer ) {
        Engineer eng = (Engineer)e;
        tax = eng.getSalary() * 0.20;
    } else {
        tax = e.getSalary() * 0.10;
    }

    return tax;
}
```

❑ Upward Casting : Type Casting 불필요

```
Employee e = new Manager();
```

❑ Downward Casting : Compile Ok, Runtime Exception 발생

```
Employee e = new Employee();
Manager m = (Manager)e;
```

❑ 이렇게 하자...

```
Employee e = new Manager();
Manager m = (Manager)e;
```

Object 클래스

► 모든 클래스의 부모클래스이다.

► extends 키워드가 안 쓰인 클래스는 extends Object를 한 것이다.

❑ Object 클래스의 메소드들

1) equals

- ref.변수를 '==' 연산자로 비교한다는 것은, 주소 값을 비교하는 것이다.
- equals() 의 내용은 비교되는 두 reference의 값을 '==' 연산자로 비교한다.
- 주소비교가 아닌 객체 내용을 비교하려는 경우에는 equals() 를 override 한다.

2) toString()

- 객체를 String 으로 변환한다.
- String Concatenation시 사용된다.
- toString() 내용은 '클래스이름@hashcode값' 이다.
- 필요하면 적절하게 override하여 사용한다.
- Primitive 타입인 경우 String으로 변환하기 위해, Wrapper 클래스의 toString()을 이용한다.

Object 클래스 – equals( ) 메소드

Object 클래스

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

String 클래스

```
public boolean equals(Object anObject) {
    if (this == anObject) {
        return true;
    }
    if (anObject instanceof String) {
        String anotherString = (String)anObject;
        int n = count;
        if (n == anotherString.count) {
            char v1[] = value;
            char v2[] = anotherString.value;
            int i = offset;
            int j = anotherString.offset;
            while (n-- != 0) {
                if (v1[i++] != v2[j++])
                    return false;
            }
            return true;
        }
    }
    return false;
}
```

Object 클래스 – toString( ) 메소드

Object 클래스

```
public String toString() {
   return getClass().getName() +
    "@" + Integer.toHexString(hashCode());
}
```

String 클래스

```
public String toString() {
   return this;
}
```

Wrapper 클래스

❑ Pimitive Data Type을 객체화 해 주는 클래스이다.

Primitive Data Type	Wrapper Class
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

❑ Primitive type을 객체화 하기

```
int pInt = 500;
Integer wInt = new Integer( pInt );
int p2 = wInt.intValue();
```

```
int pInt = 500;
Integer wInt = pInt;    //Boxing
int p2 = wInt;          //Unboxing
```

❑ String 값을 숫자로 바꾸기

```
1) int x = Integer.valueOf( "2" ).intValue();
2) int x = Integer.parseInt( "2" );
```