



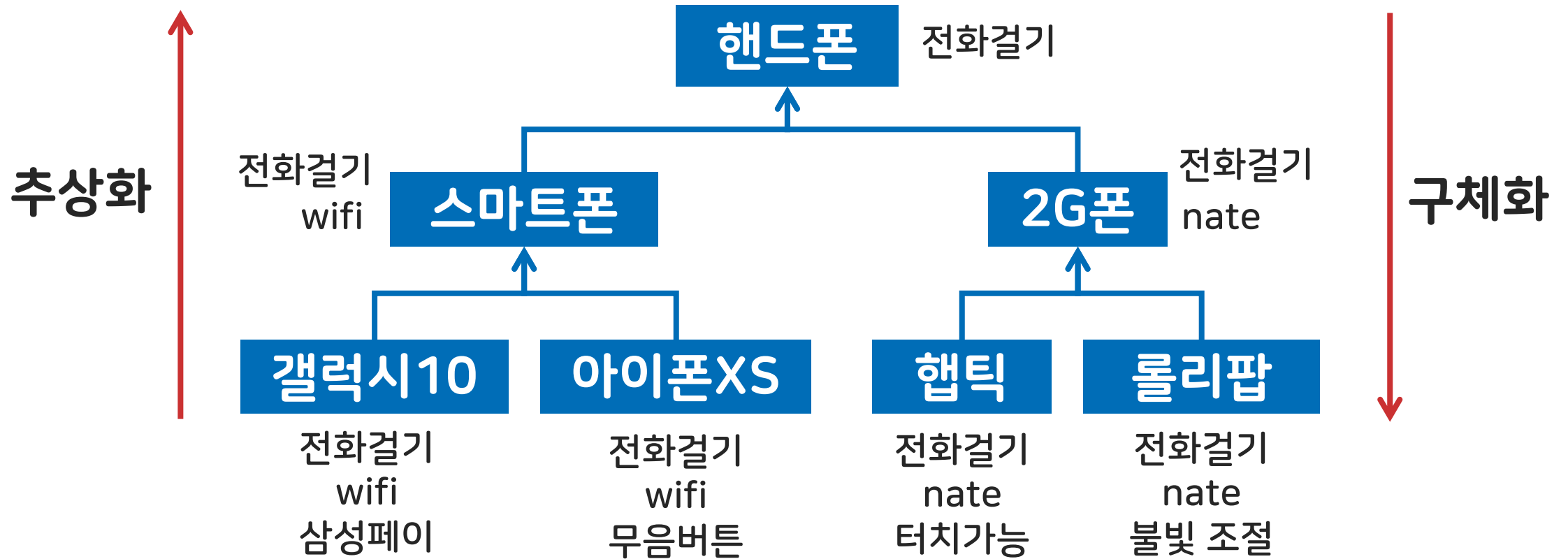
스마트인재개발원
Smart Human Resources Development

임 경 남 연구원



학습목표

1. 추상클래스에 대해 알아본다.
2. 추상클래스의 필요성에 대해 이해한다.
3. 접근제한자의 종류에 대해 알아본다.



추상화

- 클래스 간의 공통점을 찾아내서 공통의 조상을 만드는 작업
- 상속 계층도를 따라 올라갈수록 클래스의 추상화는 더욱 심화된다.

구체화

- 상속을 통해 클래스를 구현, 확장하는 작업
- 상속 계층도를 따라 내려올수록 클래스는 더 구체적이다.

추상 메소드

- 선언되어 있으나 구현되어 있지 않은 메소드(중괄호가 없는 메소드)
- `abstract` 키워드를 사용하여 선언

ex) **public abstract int** getValue();

- 추상 메소드는 서브 클래스(자식 클래스)에서 **오버라이딩 필수**

추상 클래스

- 추상 메소드를 하나라도 가진 클래스
- abstract 키워드를 사용하여 선언

ex) **public abstract class** Parent

Tip!

- 추상 클래스는 일반 메소드를 가질 수 있다.
- 일반 메소드만 가지고 있더라도 추상 클래스로 만들 수 있다.



go()

start()

front()

run()

keyUp()

up()

각각의 메소드가
어떤 기능인지
확인하는 시간이
너무 오래 걸림!


```
public abstract class KartRider{  
  
    //시작 위치 지정해주는 필드  
    final int position = 0;  
  
    // 앞으로 가는 메소드  
    public abstract void go(int now);  
  
    // 뒤로 가는 메소드  
    public abstract void back(int now);  
  
    // 드리프트 메소드  
    public abstract void drift(int now);  
  
}
```

```
public class UnbiKart extends KartRider {  
  
    int unbiPosition = position;  
  
    @Override  
    public void go(int now) {  
        unbiPosition += now;  
    }  
  
    @Override  
    public void back(int now) {  
        unbiPosition -= now;  
    }  
  
    @Override  
    public void drift(int now) {  
        unbiPosition *= now;  
    }  
  
}
```

1. 추상 클래스의 객체는 생성할 수 없다.

2. 추상 클래스 필요성

- 상속관계에서 서브클래스가 반드시 구현 해야 함을 알릴 때(**강제성**)

- 설계와 구현 분리

- ✓ 슈퍼 클래스에서는 개념적 특징 정의
- ✓ 서브 클래스에서 구체적 행위 구현

월급 계산 프로그램 만들기

EmpNo(사번)	Name	Employee	Pay (일당/연봉)	Bonus	WorkDay	비고(월급 계산)
SMHRD001	채수민	RegularEmployee	4000	400		$(\text{Pay} + \text{Bonus}) / 12$
SMHRD002	임경남	TempEmployee	3000			$\text{Pay} / 12$
SMHRD003	김운비	PartTimeEmployee	15		8	$\text{Pay} * \text{WorkDay}$

RegularEmployee 월급 계산 프로그램 만들기

RegularEmployee클래스의 필드		
타입	변수명	설명
String	empno	사번
String	name	이름
int	pay	연봉
int	bonus	보너스

RegularEmployee클래스의 메소드			
이름	리턴타입	매개변수	설명
Regular Employee	X	String 변수이름, String 변수이름, int 변수이름, int 변수이름	4개의 매개변수를 가진 생성자로서 객체 생성 시 empno, name, pay, bonus를 초기화
getMoneyPay	int	-	월 급여를 계산 후 리턴 (pay+bonus)/12
print	String	-	사번:이름:연봉 리턴

Main 클래스 만들기

- RegularEmployee 생성자를 이용해 객체 regular을 만드세요.

EmpNo(사번)	Name	Pay (일당/연봉)	Bonus
SMHRD001	홍0동	4000	400

- regular 객체를 이용하여 아래와 같이 출력하세요.

```
Console
<terminated> Main (10) [Java Application] C:WP
SMHRD001 : 홍0동 : 4000
```

- regular 객체를 이용하여 월 급여를 구하여 아래와 같이 출력하세요.

```
Console
<terminated> Main (10) [Java Application] C:WP
SMHRD001 : 홍0동 : 4000
366만원
```

TempEmployee 월급 계산 프로그램 만들기

TempEmployee클래스의 필드		
타입	변수명	설명
String	empno	사번
String	name	이름
int	pay	연봉

TempEmployee클래스의 메소드			
이름	리턴타입	매개변수	설명
Temp Employee	X	String 변수이름, String 변수이름, int 변수이름	3개의 매개변수를 가진 생성자로서 객체 생성 시 empno, name, pay를 초기화
getMoneyPay	int	-	월 급여를 계산 후 리턴 pay/12
print	String	-	사번:이름:연봉 리턴

Main 클래스 만들기

- TempEmployee 생성자를 이용해 객체 temp를 만드세요.

EmpNo(사번)	Name	Pay (일당/연봉)
SMHRD002	박O수	3000

- temp 객체를 이용하여 아래와 같이 출력하세요.

```
Console
<terminated> Main (10) [Java Application] C:\WP
SMHRD002 : 박O수 : 3000
```

- temp 객체를 이용하여 월 급여를 구하여 아래와 같이 출력하세요.

```
Console
<terminated> Main (10) [Java Application] C:\WP
SMHRD002 : 박O수 : 3000
250만원
```

PartTimeEmployee 월급 계산 프로그램 만들기

PartTimeEmployee클래스의 필드		
타입	변수명	설명
String	empno	사번
String	name	이름
int	pay	일당
int	workDay	일수

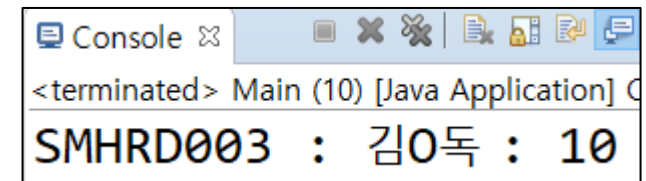
PartTimeEmployee클래스의 메소드			
이름	리턴타입	매개변수	설명
PartTime Employee	X	String 변수이름, String 변수이름, int 변수이름, int 변수이름	4개의 매개변수를 가진 생성자로서 객체 생성 시 empno, name, pay, workDay를 초기화
getMoneyPay	int	-	월 급여를 계산 후 리턴 pay*workDay
print	String	-	사번:이름:일당 리턴

Main 클래스 만들기

- PartTimeEmployee 생성자를 이용해 객체 partTime을 만드세요.

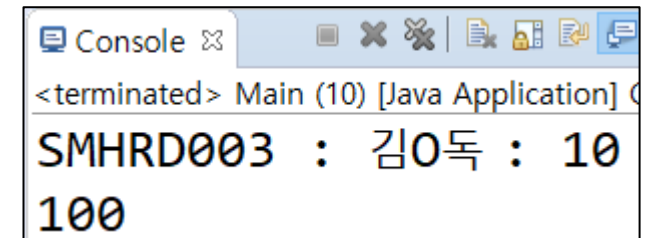
EmpNo(사번)	Name	Pay (일당/연봉)	workDay
SMHRD001	김O독	10	10

- partTime 객체를 이용하여 아래와 같이 출력하세요.

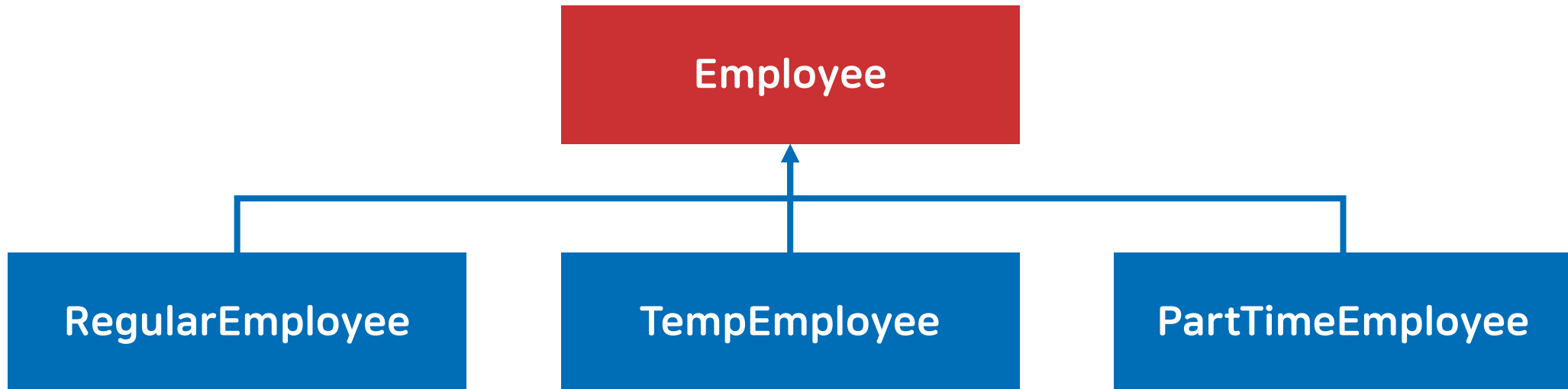


```
Console
<terminated> Main (10) [Java Application] C
SMHRD003 : 김O독 : 10
```

- partTime 객체를 이용하여 월 급여를 구하여 아래와 같이 출력하세요.



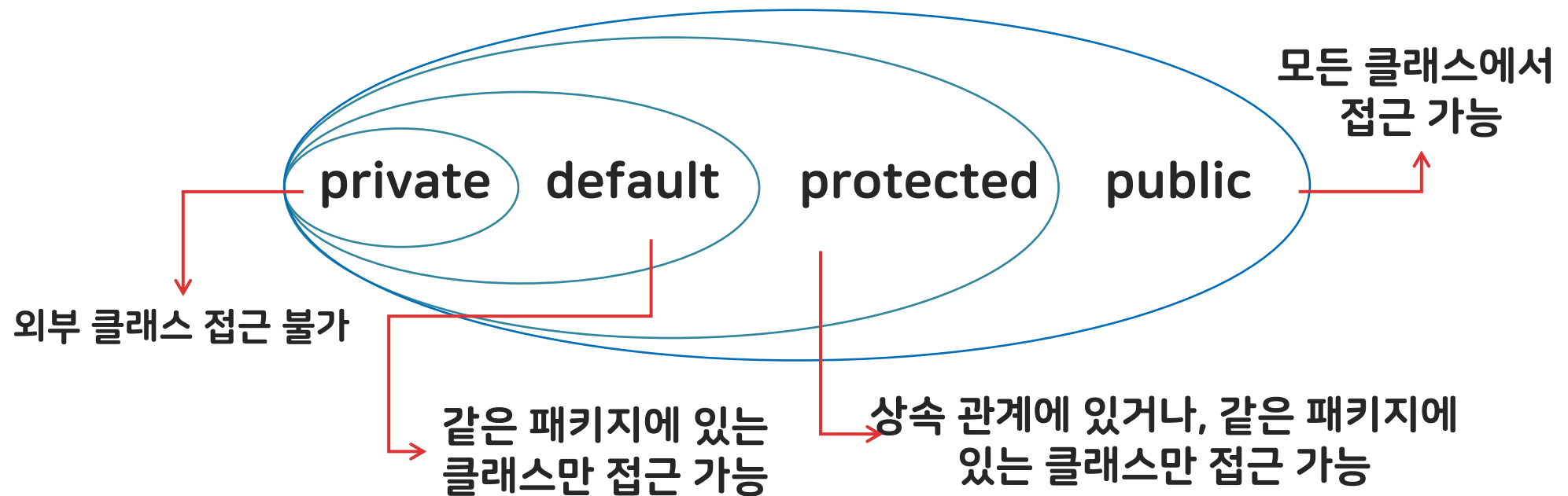
```
Console
<terminated> Main (10) [Java Application] C
SMHRD003 : 김O독 : 10
100
```



- 각각의 클래스를 대표할 수 있는 Employee 추상클래스를 설계하세요.

접근제한자(지정자)

클래스 변수와 메소드를 외부(다른 클래스)에서 접근할 수 있는 범위를 지정





다음시간에 배울 내용

인터페이스