상속과 다형성

- 상속이란
- this와 super
- Overriding과 Overliading
- 다형성이란
- 객체 캐스팅
- 접근 지정자
- static과 final

8-1

상속이란

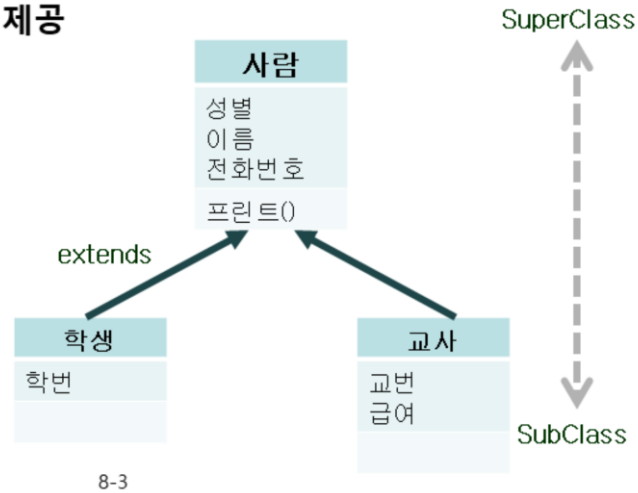
- 클래스간의 멤버들이 같은 것을 재사용한다.

학생	교사
학번 성별 이름 전화번호	교번 성별 이름 전화번호 급여
프린트()	프린트()

8-2

상속이란

- 공통 멤버를 갖는 클래스를 별도로 작성
- 기존 클래스에는 공통 멤버 추출
- 공통클래스를 기존 클래스가 상속
- 클래스의 단일 상속만 제공



8-3

상속이란

- 상속의 장점
  - 외부클래스의 멤버를 재사용
- 상속의 단점
  - 강한 결합도

```
[지정자] class 클래스명 [extends 부모_클래스명]
                [implements 부모_인터페이스명*] {
    //
}
```

8-4

상속이란

- 실습)

```
class SuperClass{
  public String name="SuperClass";
}
class SubClass extends SuperClass{ }

public class InheritanceTest_1 {
  public static void main(String[] args) {
  SuperClass s1=new SuperClass();
  SubClass s2=new SubClass();

  System.out.println(s1.name);
  System.out.println(s2.name);
  }
}
```

8-5

상속과 변수

- 부모로 통해 변수를 상속받아도 무조건 접근할 수 없다.
- 부모 클래스와 자식 클래스에서 같은 이름의 변수를 쓰면 우선 순위는 자신의 클래스의 멤버가 된다.
- 증복된 부모의 멤버 변수를 접근하기 위해서는 super 키워스를 사용한다.

8-6

• 상속과 변수

```
class SuperClass{
		private int num1=10;
		int num2=20;
		protected int num3=30;
		public int num4=40;
		public String name="SuperClass";
}

class SubClass extends SuperClass{
		public String name="SubClass";
		public String superClass=super.name;
}

public class InheritanceTest_2 {
		public static void main(String[] args) {
				SubClass s=new SubClass();

				//System.out.println(s.num1);  //접근불가
				System.out.println(s.num2);
				System.out.println(s.num3);
				System.out.println(s.num4);
				System.out.println(s.name);
				System.out.println(s.superClass);
		}
}
```

상속과 메서드

- 부모로 통해 메서드를 상속받아도 무조건 접근할 수 없다.
- 메서드 재정의(Overriding)
  - 부모 클래스의 메서드 명과 리턴 타입이 같다.
  - 매개인자 수와 자료형이 같다.
  - 예외발생은 같거나 작아야 한다.
  - 접근 지정자는 축소할 수 없다.(일반적으로 public)
- 증복된 부모의 멤버 메서드를 접근하기 위해서는 super 키워스를 사용한다.

8-8

• 상속과 메서드

```
class SuperClass{
  public void methodA(){
    System.out.println("SuperClass's methodA()");
  }
  public void methodB(){
    System.out.println("SuperClass's methodB()");
  }
}
class SubClass extends SuperClass{
  public void methodA(){
    System.out.println("SubClass's methodA()");
  }
  public void methodB(String name){
    System.out.println("SubClass's methodB()");
  }
  public void methodC(){
    System.out.println("SubClass's methodC()");
  }
}

public class InheritanceTest_3 {
  public static void main(String[] args) {
    SubClass s=new SubClass();
    s.methodA();
    s.methodB();
    s.methodC();
 }
}
```

상속과 생성자

- 자식 클래스는 생성자에서 부모의 super()를 호출한다.
- 명시적으로 호출할 때는 생성자에서 1번째 줄에 this() or super()는 1개만 와야 한다.
- 그래서 매개인자 많은 것부터 정의한다.

```
public SubClass(String name, int age){
    super(name);
    this.age=age;
}
public SubClass(String name){
    this(name, 0);
}
```

8-10

• 상속과 생성자

```
class SuperClass{
		private String name;
		public SuperClass(String name){
			this.name=name;
		}
		public void print(){
			System.out.println("name : "+ name);
		}
}
class SubClass extends SuperClass{
		private int age;

		public SubClass(String name, int age){
			super(name);
			this.age=age;
		}
		public SubClass(String name){
			this(name, 0);
		}
		public void print(){
			System.out.println("age : "+ age);
			super.print();
		}
}

public class InheritanceTest_4 {
		public static void main(String[] args) {
			//
		}
}
```

this와 super

- this
  - 자신의 클래스에서 이름이 같은 멤버 호출
  - this.변수, this.메서드(), this()
- super
  - 부모의 클래스에서 이름이 같은 멤버 호출
  - super.변수, this.메서드(), super()

8-12

Overriding과 Overloading

- Overriding
  - 상속이 전제
  - 부모의 멤버 중 일부 구현부 변경
  - 단 부모의 인터페이스는 동일하게
- Overloading
  - 하나의 클래스 또는 상속받은 클래스의 멤버의 정의
  - 다양한 자료형에 따른 사용성을 높이기 위해서
  - 단 이름은 같되 매개인자의 수나 자료형을 다르게

8-13

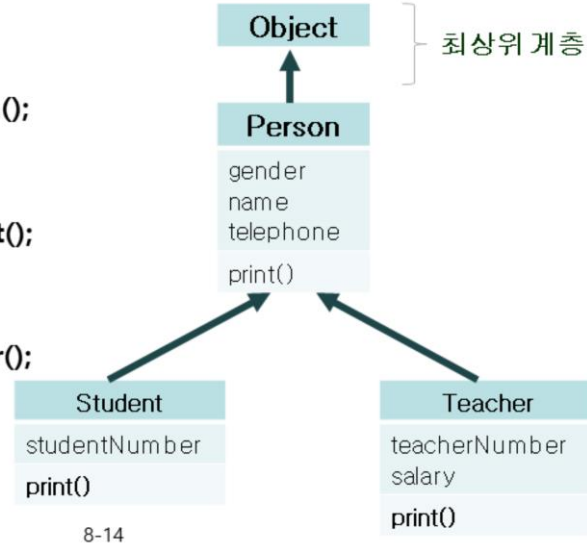
다형성이란

- 다형성(Polymorphism)
  - 이름이 같은 메서드 다양한 형태의 처리부를 제공할 수 있다.

```
Person p1=new Person();
p1.print();

Person p2=new Student();
p2.print();

Person p3=new Teacher();
p3.print();
```

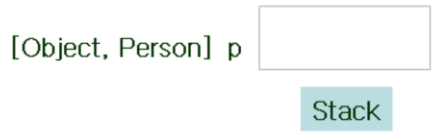


8-14

다형성이란

- 다형성 메모리 구조
  - 선언

    Person p;

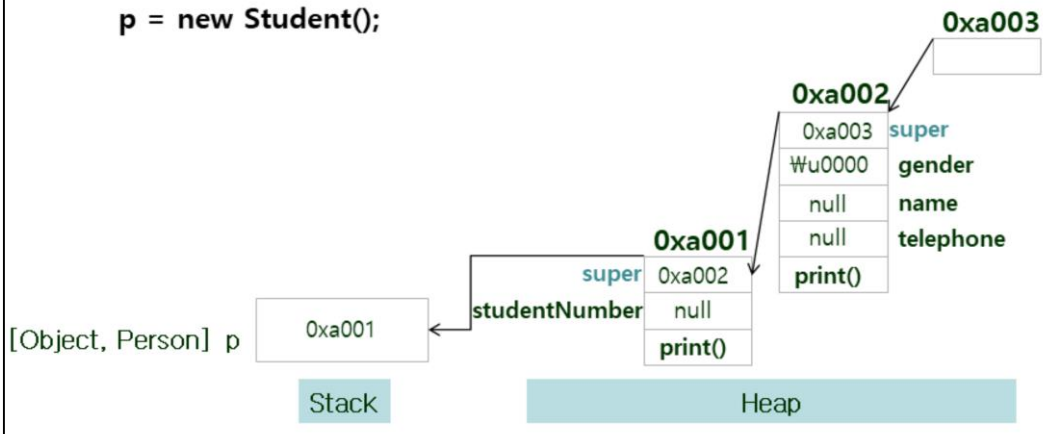


8-15

다형성이란

- 다형성 메모리 구조
  - 생성

    p = new Student();

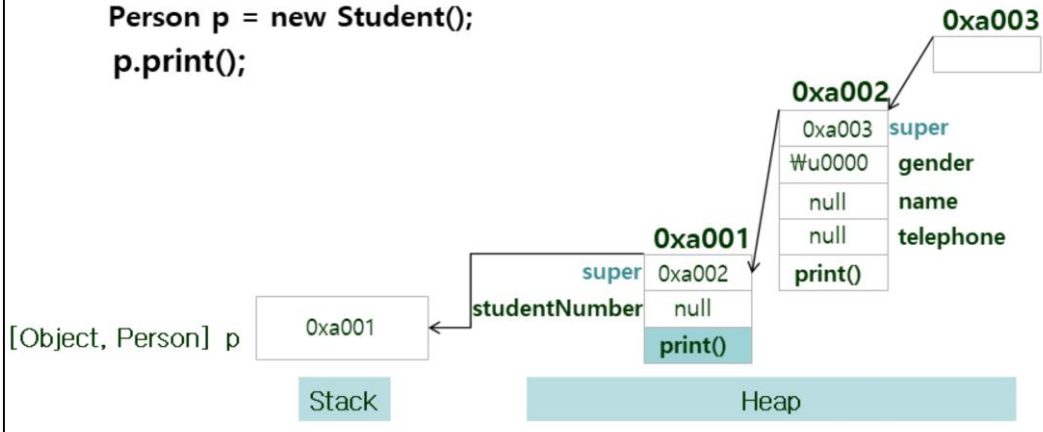


8-16

다형성이란

- 다형성 메모리 구조
  - 객체 호출

Person p = new Student();
p.print();

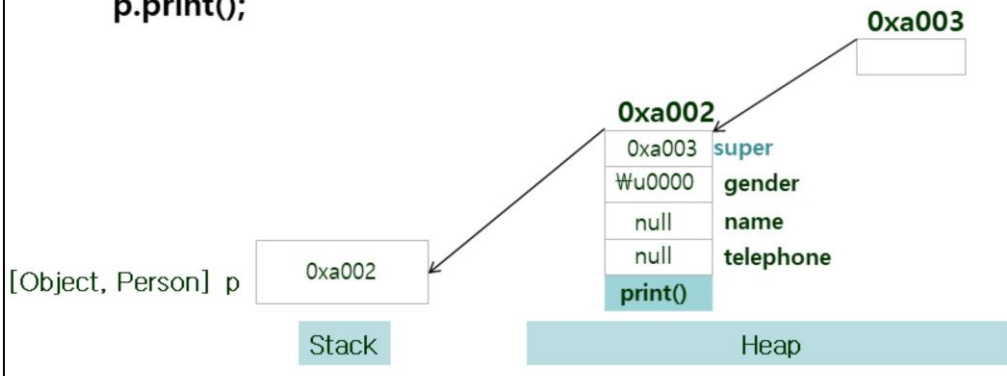


8-17

다형성이란

- 다형성 메모리 구조
  - 객체 호출

```
Person p = new Person();
p.print();
```

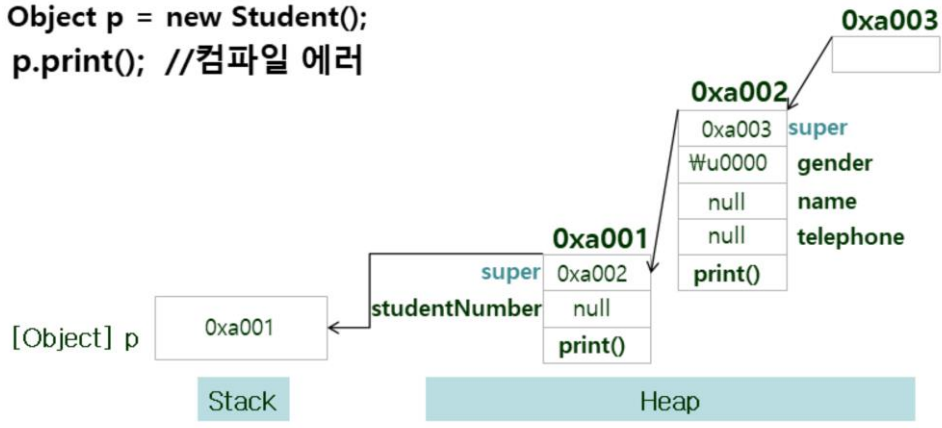


8-18

다형성이란

- 다형성 메모리 구조
  - 객체 호출

```
Object p = new Student();
p.print(); //컴파일 에러
```



8-19

다형성이란

- 다형성
  - 상속
  - 부모객체의 메서드 재정의
  - 재정의한 부모 객체로 선언하고 자식 객체로 생성
  - 선언한 부모객체는 다양한 구현부를 가진 동작을 호출할
  - 단 부모가 선언하지 않은 메서드를 호출하면 컴파일 오류 발생

8-20

객체 캐스팅

- 객체 캐스팅(Object casting)
  - 학생과 교사에서 각각 재정의된 멤버를 호출하고자 할 때
  - 학생과 교사 객체의 부모 클래스 중 Person or Object 객체로 선언
  - 단 각각에서 정의된 메서드를 호출하기 위해서는 선언 형 변경
  - instanceof로 객체의 종류 확인

8-21

객체 캐스팅

- 객체 캐스팅(Object casting)
  - 학생과 교사에서 각각 재정의된 멤버를 호출하고자 할때

```
public class Person extends Object{}
public class Student extends Person{}
public class Teacher extends Person{}
```

```
public class SchoolBiz{
   public void print(Object obj){
      if(obj instanceof Teacher){
         Teacher t=(Teacher)obj;
      }else if(obj instanceof Student){
         Student s=(Student)obj;
      }else {
         //
      }
   }
}
```

8-22

접근 지정자

- 접근 지정자(Access Modifier)

접근 범위	멤버 클래스	패키지	상속	모두접근
public	O	O	O	O
protected	O	O	O	
[생략]	O	O		
private	O			

8-23

static과 final

- static 지정자(Modifier)
  - new를 사용하지 않고 바로 접근 가능
    Math.random()
  - 공유
  - static한 멤버는 static 하지 않는 멤버를 접근할 수 없다.

	static 사용여부	기타
class	X	
variable	O	private static int counter;
method	O	public static void print(){}
constructor	X	

8-24

static과 final

- **final 지정자(Modifier)**
  - **final 로 정의한 상수는 일반적으로 대문자로 선언**
  - **고정값을 정의**
  - **일반적으로 상수는 static과 같이 사용**

	final 사용여부	기타
class	O	자식 클래스를 가질 수 없다. public final class FinalClass{}
variable	O	사용자 정의 상수 일반적으로 대문자 표기 final static byte DEFAULT_VALUE=100;
method	O	상속받은 메서드중 재정의를 불가 public final void finalMethod(){}
constructor	X	

8-25

Object

- **Object 클래스**
  - **java.lang.Object**
  - **모든 클래스의 최상위 클래스**

```
class Person{
        char gender='W';
        String name;

        public  void print(){
                System.out.println("name  : "+ name);
                System.out.println("gender  : "+ gender);
        }
}
```

```
Person p=new Person();
p.print();
System.out.println(p);
```

```
>java ObjectTest
name  : null
gender  : W
kr.zeroand.java.inheritance.Person@dd20f6
```

8-26

Object

- **Object 클래스**
  - **toString() 재정의**
  - **System.out.println()에서 자동 호출**

```
class Person{
...
    @Override
    public String toString() {
        return "name : "+ name + ", gender : "+ gender;
    }
}
```

```
Person p=new Person();
p.print();
System.out.println(p);
System.out.println(p.toString());
```

```
>java ObjectTest
name : null
gender : W
name : null, gender : W
name : null, gender : W
```

8-27

Object

- **Object 클래스**
  - **equals(), hashCode() 재정의**
  - **System.out.println()에서 자동 호출**

```
class Person{
…
    @Override
    public int hashCode() { … }
    @Override
    public boolean equals(Object obj) { … }
}
```

```
Person p1=new Person("김민성", 'M');
Person p2=new Person("김민성", 'M');
System.out.println(p1 ==p2 );
System.out.println(p1.equals(p2));
```

```
>java ObjectTest
false
true
```

8-28

• Object

• Person.java

```
class Person{
    private char gender='W';
    private String name;
    public Person(){}
    public Person(String name, char gender){
        this.name=name;
        this.gender=gender;
    }
    public void print(){
        System.out.println("name : "+ name);
        System.out.println("gender : "+ gender);
    }
    @Override
    public String toString() {
        return "name : "+ name + ", gender : "+ gender;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + gender;
        result = prime*result+((name == null) ? 0 : name.hashCode());
        return result;
    }
```

• Object

```
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass())  return false;

        Person other = (Person) obj;
        if (gender != other.gender)return false;
        if (name == null) {
            if (other.name != null)  return false;
        } else if (!name.equals(other.name))
            return false;
        return true;
    }
}//Person class end
```

•ObjectTest.java

```
public class ObjectTest {
    public static void main(String[] args) {
        Person p=new Person();
        p.print();
        System.out.println(p);
        System.out.println(p.toString());

        Person p1=new Person("김민성", 'M');
        Person p2=new Person("김민성", 'M');
        System.out.println(p1 ==p2 );
        System.out.println(p1.equals(p2));
    }
}
```

LAB

8-31