Java 기초

상속

상속(Abstract)

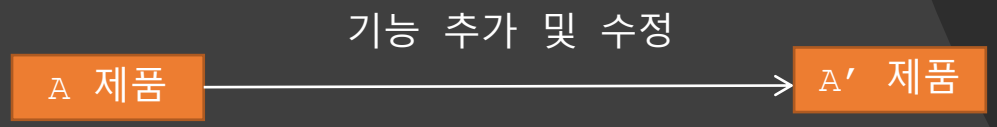
2002

2005

2009

상속(Abstract)

- 상속(Abstract)
  - 기존의 클래스를 재사용하여 새로운 클래스를 작성하는 것



A'제품은 A제품을 상속받았다

  - 여기서 상속하는 쪽을 부모클래스, 상속받는 쪽을 자식 클래스라고 한다.
  - 상속 시 부모의 형질을 자식클래스가 그대로 물려받는다.
  - 자식의 맴버개수는 조상보다 작을 수 없다.

상속(Abstract)

- 상속 방법

public class [상속할 클래스명] extends [상속받을 클래스명]

```
public class AClass {
    int x;
    int y;
    public void insert(){}
}
```

상속

```
public class BClass extends AClass{
    int z;
}
```

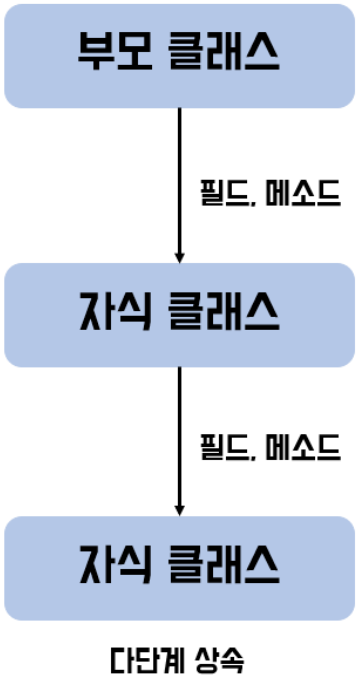
상속(Abstract)

- 상속 시 유의점
  - 생성자는 상속되지 않는다.
  - 클래스는 반드시 단 하나의 클래스만 상속이 가능하며 상속받은 클래스를 다른 클래스가 상속이 가능하다

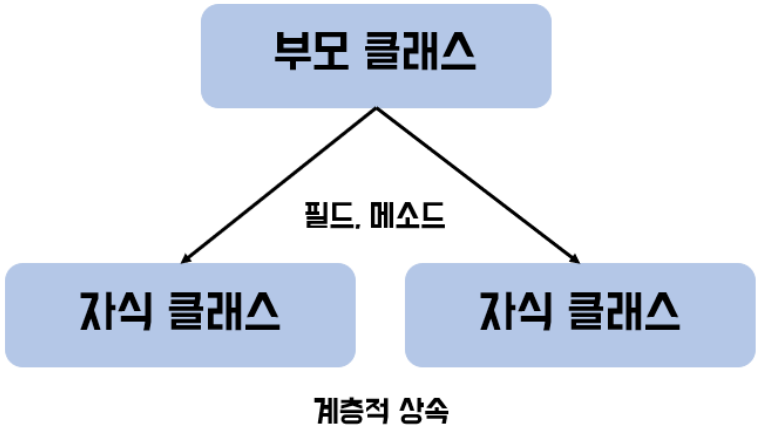
- 상속의 장점
  - 적은양의 코드로 새로운 클래스를 작성할 수 있다
  - 코드의 추가 변경이 용이하다.

상속(Abstract)

상속의 종류



다단계 상속



계층적 상속

상속(Abstract)

- super
  - this가 자기자신을 가리켰듯이 super는 부모를 가리키는 키워드
  - super 클래스를 통해 부모의 속성과 메서드에 접근이 가능하지만 일반적으로 생략이 가능하다.
  - 단 뒤에 나올 오버라이딩된 메서드의 경우 super를 써야 상위의 메서드에 접근이 되므로 유의할 것
  - 쓰는 방법은 this와 동일하다.

super.method() or super.field

상속(Abstract)

- super 예제

```java
public class ParentClass {
    int num = 2;

    public void parentMethod1() {
        System.out.println("이것은 parentMethod 입니다.");
    }
}
public class SuperClassEx01 extends ParentClass{
    public void childMethod1() {
        // 실제 부모 메서드에 접근했을 때의 모습
        super.parentMethod1();
        // 실제 부모 필드에 접근했을 때의 모습
        System.out.println(super.num);
    }

    public static void main(String[] args) {
        new SuperClassEx01().childMethod1();
    }
}
```

```
이것은 parentMethod 입니다.
2
```

상속(Abstract)

- super 생성자
  - super 사용 시 주의해야 할 점은 생성자 접근이다.
  - 생성자는 상속이 되지 않고 참조가 된다.
  - 단 부모 클래스에서 명시적으로 생성자를 선언하였을 경우 자식 객체에서는 강제 참조하는 과정에서 파라미터의 미스매치로 에러가 발생하는 경우가 생길 수 있다.
  - 이럴 경우 부모클래스에서 생성한 생성자를 자식 클래스의 생성자로 호출해야 하는데 이럴 때 super를 이용해서 바로 위의 생성자를 호출해야 한다.
  - 상위의 생성자를 호출하는 방법은 다음과 같다.
  - 만약 생성자를 선언했는데 생성자에 파라미터가 호출되지 않았다면 선언을 하지 않더라도 묵시적으로 선언이 된다.

```
super([...args])
```

상속(Abstract)

- super 생성자 예제 1

```
public class SuperConstructorEx01 {
    public SuperConstructorEx01() {
        System.out.println("SuperConstructor 생성자");
    }
}
public class SuperConstructMainEx01 extends SuperConstructorEx01{
    public SuperConstructMainEx01() {
        // 아무 것도 선언하지 않을 경우 매개변수가 없는 부모 생성자를 그대로 참조한다.
        // 디폴트로 이 안에는 SuperConstructorEx01의 생성자가 참조 되어진다.
    }

    public static void main(String[] args) {
        new SuperConstructMainEx01();
    }
}
```

상속(Abstract)

- super 생성자 예제 2

```java
public class SuperConstructorEx02 {
    public SuperConstructorEx02(int a, String b) {
        // 명시적으로 생성자에 들어가는 매개변수를 선언한 경우
        System.out.println("a : "+a);
        System.out.println("b : "+b);
    }
}
public class SuperConstructMainEx02 extends SuperConstructorEx02{

    public SuperConstructMainEx02(int a, String b) {
        // 상속받은 클래스에 매개변수가 존재하는 생성자가 있을 경우 반드시 명시적으로
        // 생성자를 선언해 주어야 하며 반드시 로직 안에는 super(args)라고
        // 부모 객체의 생성자를 참조하는 로직이 있어야 한다.
        super(a, b);
    }

    public static void main(String[] args) {
        new SuperConstructMainEx02(5, "명시적 생성자");
    }
}
```

```
a : 5
b : 명시적 생성자
```

오버라이딩(Overriding)

- 메소드 오버라이딩(Method Overriding)
  - 부모에게서 받은 메서드의 내용을 자식 클래스에서 변경하는 것
  - 자손 클래스에서 오버라이딩하는 메서드는 조상 클래스의 메서드와 이름, 매개변수, 리턴 타입이 모두 같아야 한다.
  - 접근 제어자는 조상 클래스의 메서드보다 좁은 범위로 변경 할 수 없다.
  - 조상 클래스의 메서드보다 많은 수의 예외를 선언할 수 없다.
  - 인스턴트메서드를 static이나 혹은 그 반대로 선언이 불가능하다

오버라이딩(Overriding)

- 메소드 오버라이딩(Method Overriding) 예제

```java
public class OverridingParent01 {
    public void parentMethod1() {
        System.out.println("OverridingParent01 의 메서드");
    }
}
```

```java
public class OverridingMain01 extends OverridingParent01{

    // 오버라이딩 메서드는 반드시 이름을 부모에서 선언한 타겟 메서드와
    // 동일한 이름으로 작성해야 하며 오버라이딩된 메서드는 여러 툴에서
    // Override란 어노테이션을 제공하지만 없아더 크게 상관은 없다.
    @Override
    public void parentMethod1() {
        System.out.println("OverridingMain에서 덮어씌운 로직");
    }

    public static void main(String[] args) {
        new OverridingMain01().parentMethod1();
    }
}
```

```
OverridingMain에서 덮어씌운 로직
```

Object

- Object
  - 모든 클래스의 최상위 클래스.
  - 모든 클래스는 Object를 상속받으며 Object가 제공한 메서드를 사용 가능하다.

메소드	설 명
boolean equals(Object obj)	두 개의 객체가 같은지 비교하여 같으면 true를, 같지 않으면 false를 반환한다.
String toString()	현재 객체의 문자열을 반환한다.
protected Object clone()	객체를 복사한다.
protected void finalize()	가비지 컬렉션 직전에 객체의 리소스를 정리할 때 호출한다.
Class getClass()	객체의 클래스형을 반환한다.
int hashCode()	객체의 코드값을 반환한다.
void notify()	wait된 스레드 실행을 재개할 때 호출한다.
void notifyAll()	wait된 모든 스레드 실행을 재개할 때 호출한다.
void wait()	스레드를 일시적으로 중지할 때 호출한다.
void wait(long timeout)	주어진 시간만큼 스레드를 일시적으로 중지할 때 호출한다.
void wait(long timeout, int nanos)	주어진 시간만큼 스레드를 일시적으로 중지할 때 호출한다.