상속 (Inheritance)

기존의 클래스를 재사용하여 새로운 클래스를 작성하는 것.

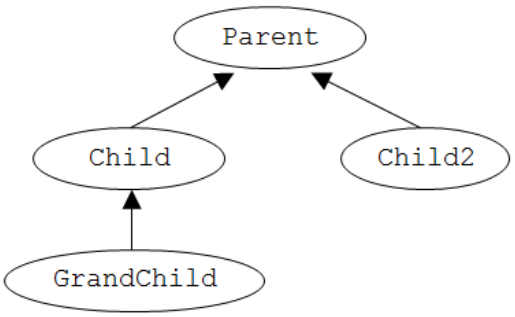
두 클래스의 관계를 부모와 자식으로 맺어 주는 것.

자식은 부모의 모든 멤버를 상속 받는다. (생성자, 초기화 블록 제외)

자식의 멤버 개수는 부모보다 적을 수 없다. (같거나 많다)

```
class Child    extends Parent {
    ....
}
```

```
class Parent {}
class Child extends Parent {}
class Child2 extends Parent {}
class GrandChild extends Child {}
```



자바는 단일 상속만 허용한다. (C++는 다중 상속을 허용한다)

```
class TVCR extends TV, VCR {      // 이와 같은 표현은 허용하지 않는다.
    //...
}
```

실습예제1

Person과 Student 클래스 간의 상속관계를 이해하면서 실습해 보세요

```
class Person {
        int age;                        // Student 클래스에서 접근 가능
        public String name; // Student 클래스에서 접근 가능
        protected int height;// Student 클래스에서 접근 가능
        private int weight;    // Student 클래스에서 접근 불가
        public void setWeight(int weight) {
                this.weight = weight;
        }
        public int getWeight() {
                return weight;
```

```
	}

	void show() {
		System.out.println("이름: " + name);
		System.out.println("몸무게: " + weight);
		System.out.println("키: " + height);
	}
}

public class Student extends Person {
	void set() {
		age = 30;
		name = "홍길동";
		height = 175;
		setWeight(99);

	}
	public static void main(String[] args) {
		Student s = new Student();
		s.set();
		 s.show();
	}
}
```

```
이름: 홍길동
몸무게: 99
키: 175
```

super()

자식클래스의 생성자에서 부모클래스의 생성자의 호출을 명시적으로 선택하는 키워드.

아래코드에서 super()를 사용했는데 출력 결과를 반드시 이해해야 합니다. 실습해 보세요

```
class A {
	public A() {
		System.out.println("생성자A");
	}
	public A(int x) {
		System.out.println("매개변수생성자A" + x);
	}
}

class B extends A {
	public B() {
		System.out.println("생성자B");
	}
	public B(int x) {
		super(x);
		System.out.println("매개변수생성자B" + x);
	}
}

public class CostructorEx {
	public static void main(String[] args) {
		B b = new B(5);
	}
}
```

```
매개변수생성자A5
매개변수생성자B5
```

업캐스팅과 다운캐스팅

업캐스팅

자식클래스는 부모클래스의 모든 특성을 상속받는다. 자식 클래스 객체가 부모 클래스 타입으로 변환 되는 것. 아래 코드 실습하세요

```
class Person {
	String name;
	String id;

	public Person(String name) {
		this.name = name;
	}
}

class Student extends Person {
	String grade;
	String department;

	public Student(String name) {
		super(name);
	}
}

public class UpcastingEx {
	public static void main(String[] args) {
		Person p;
		Student s = new Student("홍길동");
		p = s;   // 업캐스팅 발생

		System.out.println(p.name);

		// p.grade = "A";   // 컴파일 오류 발생
		// p.department = "Computer"; // 컴파일 오류 발생
	}
}
```

```
홍길동
```

업캐스팅은 굳이 아래와 같이 명시적 타입 형변환을 하지 않아도 된다. 왜냐하면 Student 객체는 Person 타입이기도 하기 때문이다.

p = (Person)s;   // p=s; 와 동일

다운캐스팅

업캐스팅된 것을 다시 원래대로 되돌리는 것을 다운캐스팅이라고 한다.

다운캐스팅은 업캐스팅과 달리 명시적으로 타입을 지정해야 한다.

Student s = (Student)p; // 다운캐스팅, 강제 타입 변환

위의 Person, Student 클래스를 재사용하여 실습하시오

```
class Person {
	String name;
```

```
        String id;

        public Person(String name) {
                this.name = name;
        }
}

class Student extends Person {
        String grade;
        String department;

        public Student(String name) {
                super(name);
        }
}

public class UpcastingEx {
        public static void main(String[] args) {
                Person p = new Student("홍길동"); // 업캐스팅 발생
                Student s;

                s = (Student)p; // 다운캐스팅

                System.out.println(s.name); // 오류 없음
                s.grade = "A";
        }
}
```

```
홍길동
```

instanceof 연산자

인스턴스 변수가 가리키는 실제 객체가 어떤 클래스 타입인지 구분하기 위해 자바에서는 instanceof 연산자를 사용한다.

```
Person jee = new Student();
Person kim = new Professor();
Person lee = new Researcher();
if (jee instanceof Person)      // jee는 Person 타입이므로 true
if (jee instanceof Student)     // jee는 Student 타입이므로 true
if (kim instanceof Student)     // kim은 Student 타입이 아니므로 false
```

메서드 오버로딩

부모클래스의 메서드를 자식클래스에서 재정의 하는 것이다. 한마디로, 부모클래스의 메서드를 자식클래스에서 덮어쓴다는 의미이다.

```
class Point {
     int x;
     int y;

     String getLocation() {
          return "x :" + x + ", y :"+ y;
     }
}

class Point3D extends Point {
     int z;
     String getLocation() {      // 오버라이딩
          return "x :" + x + ", y :"+ y + ", z :" + z;
     }
}
```

오버로딩과 오버라이딩의 구분

오버로딩(overloading) - 기존에 없는 새로운 메서드를 정의하는 것(new)
오버라이딩(overriding) - 상속받은 메서드의 내용을 변경하는 것(change, modify)

```
class Parent {
    void parentMethod() {}
}

class Child extends Parent {
    void parentMethod() {}          // 오버라이딩
    void parentMethod(int i) {}     // 오버로딩

    void childMethod() {}
    void childMethod(int i) {}      // 오버로딩
    void childMethod() {}           // 에러!!! 중복정의임
}
```

실습예제2

Person을 상속받는 Professor라는 새로운 클래스를 만들고, Professor 클래스에서 getPhone() 메서드를 오버라이딩 하시오.

```
class Person {
        String phone;

        public void setPhone(String phone) {
                this.phone = phone;
        }
        public String getPhone() {
                return phone;
        }
}

class Professor extends Person {
        public String getPhone() { // Person의 getPhone()을 오버라이딩
                return "Professor : " + super.getPhone(); // Person의 getPhone() 호출
```

```
        }
}

public class Overriding {
        public static void main(String[] args) {
                Professor a = new Professor();
                a.setPhone("011-123-1234"); // Professor의 getPhone() 호출
                System.out.println(a.getPhone());

                Person p = a;
                System.out.println(p.getPhone()); // 동적 바인딩에 의해
                                                  // Professor의 getPhone() 호출
        }
}
```

```
Professor : 011-123-1234
Professor : 011-123-1234
```

실습예제3

3개의 클래스(.java)를 구성하고 상속관계를 형성해서 객체를 만들어 보세요

```
< Student.java >
public class Student {
        private String studentNo;
        private String name;
        private int age;

        public Student() { } // Default Constructor

        public Student(String studentNo, String name, int age) {
                this.studentNo = studentNo;
                this.name = name;
                this.age = age;
        }

        public String getStudentNo() {
                return studentNo;
        }

        public void setStudentNo(String studentNo) {
                this.studentNo = studentNo;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public int getAge() {
                return age;
        }

        public void setAge(int age) {
                if (age > 0) {
                        this.age = age;
                }
        }

        public void displayInformation() {
```

```
                System.out.println("학번 : " + studentNo);
                System.out.println("이름 : " + name);
                System.out.println("나이 : " + (age > 0 ? age : "미정"));
                System.out.println("----------");
        }
}
```

< UniversityStudent.java >

```
public class UniversityStudent extends Student {
        private String major;

        public UniversityStudent() { }
        public UniversityStudent(String studentNo, String name, int age, String major) {
                super(studentNo, name, age);
                this.major = major;
        }

        public String getMajor() {
                return major;
        }

        public void setMajor(String major) {
                this.major = major;
        }

        public void displayInformation() {
                System.out.println("학번 : " + getStudentNo());
                System.out.println("이름 : " + getName());
                System.out.println("나이 : " + (getAge() > 0 ? getAge() : "미정"));
                System.out.println("전공 : " + getMajor());
                System.out.println("----------");
        }
}
```

< Driver1.java >

```
public class Driver1 {
        public static void main(String[] args) {
                UniversityStudent u1 = new UniversityStudent("U001", "나대학", 25, "Computer Science");
                Student s2 = new Student("S002", "김삿갓", 30);

                u1.displayInformation();
                s2.displayInformation();
        }
}
```

```
학번 : U001
이름 : 나대학
나이 : 25
전공 : Computer Science
----------
학번 : S002
이름 : 김삿갓
나이 : 30
----------
```

수고했습니다. 이제 상속을 이용하여 아래 문제를 코딩해 보세요. 클래스를 3개 만들어야 합니다. 즉, 부모클래스, 자식클래스, main ()를 가지는 클래스.

공통: 1042, 1046, 1602, 1604, 1120, 1530, 1701, 1702, 1712