📖 Java Study Skill Up Team Review

Inheritance

- 상속(Inheritance[인헤러턴스])은 프로그램에서도 부모 클래스의 필드와 메서드를 자식 클래스에 물려줄 수 있는 것이다.
- 하나의 java 파일에서 상속 이해를 위한 소스를 작성해 보았다.

```
package com.chapter.ch07.example.ex01;

public class Inheritance {
  public class ParentClass {
    String parentField;

    void parentMethod() {
      System.out.println("parent method");
    }
  }

  public class ChildrenClass extends ParentClass {
    String childrenField;

    void childrenMethod() {
      System.out.println("childrenMethod");
    }
  }

  public static void main(String[] args) {
    ParentClass parentClass = new ParentClass(); // ❌ 오류 발생
  }
}
```

[오류 발생]

- new ParentClass(); 소스에서 cannot be referenced from a static context 오류가 발생했다.
  - 해석: 정적 컨텍스트(static context)에서는 참조할 수 없다.
  - 즉 main 메서드가 static 으로 정의되어 있는데 ParentClass 와 ChildrenClass 는 Inheritance 클래스의 내부 클래스로 정의되어 있기 때문이다.
  - 그래서 내부 클래스는 외부 클래스의 인스턴스를 필요로 하며 정적 메서드인 main 에서 직접 내부 클래스를 인스턴스화 할 수 없다.

[오류 분석]

- static 관련 참조 오류는 상속과 관계있는 것은 아니다.
- 자바에서 정적(static) 메서드는 클래스 수준에서 실행되며 클래스의 인스턴스 생성 없이 호출할 수 있다.
- 그러나 내부 클래스(ChildrenClass)의 인스턴스를 생성하려면 먼저 외부 클래스의 인스턴스가 생성되어야 한다.
- 내부 클래스는 외부 클래스의 인스턴스에 종속적이기 때문이다.

- 따라서 외부 클래스의 인스턴스 생성 없이 내부 클래스의 인스턴스를 생서하려고 시도하면 오류가 발생한다.

👉 결론적으로 내부 클래스의 인스턴스를 생성하려면 먼저 외부 클래스의 인스턴스를 생성해야 한다.

[해결 방법]

- 방법 1
  - ParentClass 와 ChildrenClass 를 static 으로 변경하거나 정적 내부 클래스로 정의한다.

```
package com.chapter.ch07.example.ex01;

public class Inheritance {
  public static class ParentClass {
    String parentField;

    void parentMethod() {
      System.out.println("parent method");
    }
  }

  public static class ChildrenClass extends ParentClass {
    String childrenField;

    void childrenMethod() {
      System.out.println("childrenMethod");
    }
  }

  public static void main(String[] args) {
    ChildrenClass childrenClass = new ChildrenClass(); // ⭕ 방법 1
    childrenClass.parentMethod();
  }
}
```

- 방법 2
  - main 메서드 내에서 Inheritance 클래스의 인스턴스를 생성한 후 그 인스턴스를 사용해서 내부 클래스를 인스턴스화 한다.

```
package com.chapter.ch07.example.ex01;

public class Inheritance {
    public class ParentClass {
        String parentField;

        void parentMethod() {
            System.out.println("parent method");
        }
    }
```

```
    public class ChildrenClass extends ParentClass {
        String childrenField;

        void childrenMethod() {
            System.out.println("childrenMethod");
        }
    }

    public static void main(String[] args) {
        // ParentClass parentClass = new ParentClass(); // ❌ 오류 발생
        Inheritance inheritance = new Inheritance();    // ⭕ 방법 2
        Inheritance.ChildrenClass childrenClass = inheritance.new ChildrenClass();
        childrenClass.parentMethod();
    }
}
```

- 방법 3
- static void 에서 static 을 제거한다.

```
package com.chapter.ch07.example.ex01;

public class Inheritance {
    public class ParentClass {
        String parentField;

        void parentMethod() {
            System.out.println("parent method");
        }
    }

    public class ChildrenClass extends ParentClass {
        String childrenField;

        void childrenMethod() {
            System.out.println("childrenMethod");
        }
    }

    public void main(String[] args) {   // ⭕ 방법 3
        ChildrenClass childrenClass = new ChildrenClass();
        childrenClass.childrenMethod();
    }
}
```

[결론]

- static void 에 대해서 알게되었고 예시가 잘못된 거 같아서 정석 느낌으로 상속에 대한 예제를 다시 작성해 보았다.

다시 돌아가 상속이란?

- 상속(Inheritance[인헤러턴스])은 프로그램에서도 부모 클래스의 필드와 메서드를 자식 클래스에 물려줄 수 있는 것이다.
- 예제로 부모 클래스, 자식 클래스, 실행 클래스를 생성하였다.

✍ ParentClass.java

```
package com.chapter.ch07.example.ex01;

public class ParentClass {
    String parentClassField;

    public void parentClassMethod() {
        System.out.println("parent class method");
    }
}
```

✍ ChildrenClass.java

```
package com.chapter.ch07.example.ex01;

public class ChildrenClass extends ParentClass {
    String childrenClassField;

    void childrenClassMethod () {
        System.out.println("children class method");
    }
}
```

✍ Main.java

```
package com.chapter.ch07.example.ex01;

public class Main {
    public static void main(String[] args) {
        System.out.println("ChildrenClass 클래스를 인스턴스화 하여 childrenClass 인
스턴스를 생성한다.");
        ChildrenClass childrenClass = new ChildrenClass();
        System.out.println("childrenClass 객체를 통해 부모 클래스에 정의된
parentClassMethod를 호출한다.");
        childrenClass.parentClassMethod();
    }
}
```

- 이처럼 상속은 간단하게 말하자면 이미 잘 개발된 클래스를 재사용 할 수 있다.
- 그렇기에 중복 코드를 줄여 개발 시간을 단축시켜 준다.

- 또 추가 작성해야 하는 필드와 메서드가 상속받은 클래스에 존재한다면 추가 작성할 필요가 없어 효율적이다.
- 부모 클래스를 수정하면 모든 자식 클래스에 수정되기 때문에 클래스의 수정을 최소화할 수 있다.

✋ 여러 개의 부모 클래스를 상속 받을 수 있나요?

❌ 정답은 '없다'이다.

- 다중 상속을 허용하는 언어는 존재하지만 자바에서는 다중 상속을 허용하지 않는다.
- 그 이유는 다중 상속을 받은 부모 클래스에서 동일한 명칭의 메서드가 존재할 수 있다.
- 이 경우 어떤 부모 클래스에 접근할지에 대해 모호해지기 때문에 다주 상속을 허용하지 않는다.

[생성자]

- 모든 객체는 생성자를 호출해야 생성이 된다.
- 그걸 떠나서 나는 부모 생성자를 생성하지도, ParentClass() 와 같이 호출하지도 않았는데.. 위 소스는 어떻게 동작이 되는 걸까?
- 해답은 '자식 생성자'에 있다.
- 부모 생성자는 자식 생성자의 제일 첫 줄에 숨겨져 있는 super()에 의해 호출된다.
- super()는 컴파일 과정에서 자동으로 추가되면서 부모의 기본 생성자를 호출하는 것이다.
- 여기서 말하는 기본 생성자는 부모생성자() 로써 매개변수가 존재하지 않는 것을 뜻한다.
- 만약 부모 생성자에 매개변수가 존재한다면 super(매개변수...)와 같이 일치하도록 맞춰 주어야 한다.

✍ ParentClass.java

```
package com.chapter.ch07.example.ex01;

public class ParentClass {
    String parentClassField;

    // 부모 기본 생성자 선언
    public ParentClass() {
      System.out.println("ParentClass 기본 생성자");
    }

    public void parentClassMethod() {
        System.out.println("parent class method");
    }
}
```

✍ ChildrenClass.java

```
package com.chapter.ch07.example.ex01;

public class ChildrenClass extends ParentClass {
    String childrenClassField;

    // 자식 기본 생성자 선언
    public ChildrenClass(String parentClassField) {
```

```
        super();    // 생략 가능 (컴파일 시 자동으로 추가됨)
        System.out.println("childrenClass 생성자 선언");
    }

    void childrenClassMethod () {
        System.out.println("children class method");
    }
}
```

- 이처럼 생성자를 호출해 주어야 객체가 생성된다.
- 하지만 코드를 입력하지도 않았는데 잘 동작했던 이유는
- 생략 시에는 컴파일을 할 때 알아서 기본 생성자를 생성해 주기 때문이다.
- 하지만 기본 생성자에 매개 변수에 있는 경우에는 super 를 꼭 맞춰서 넣어줘야 한다.

✍ ParentClass.java

```
package com.chapter.ch07.example.ex01;

public class ParentClass {
    String parentClassField;

    public ParentClass(String parentClassField) {
        parentClassField = this.parentClassField;
    }

    public void parentClassMethod() {
        System.out.println("parent class method");
    }
}
```

- 즉 이처럼 매개변수를 가지는 경우에는 생략해도 되는 super(); 를 사용하면 안되고 인자 개수를 맞춰 주어야 한다.