7_29_2022__java_Extends, AbstractClass, Interface

Step08_gitWithJava, Extends

~Step10_Interface

Git

- 다른 프로젝트를 이미 깃에 저장된 곳에 함께 넣고 싶다면
  - Team → Share Project
- gitignore
  - 설정파일들이 들어있다. 함께 commit 해서 좋을게 없다.
  - 업로드 하지 않을 설정들을 여기에 입력하면됨

Extends

- java의 모든 data는 object type 이다. 기본 data type들도 참조 데이터화 되어 들어간다.
- HandPhone, Phone을 상속받은 SmartPhone 클래스는 Object까지 총 4개의 타입을 가지고 있다.
- super( )로 부모 클래스 참조

Abstract Class(추상 클래스[미완성 클래스])

- 미완성 클래스 - 기반 시설을 만들어 놓을테니 완성해서 사용하라!
- 미완성인 메소드를 추상메소드라고 하며 이 추상 메소드를 하나이상 가지고 있는 클래스를 추상 클래스라고 한다.
- 추상 메소드와 추상 클래스를 만들기 위해서는 abstract 예약어가 필요하다.
- 미완성 메소드 사용시 중괄호를 사용하지 않고 바로 세미콜른으로 닫음

```
package test.mypac;
public abstract class Weapon {
  //무기 작동을 준비하는 메소드
  public void prepare() {
    System.out.println("무기 작동을 준비합니다.");
  }
  //공격하는 메소드
  public abstract void attack();
```

- 추상메소드 이기는 하지만 모양 자체는 완벽함
  - call 했을 때 무엇이 할지 정해지지는 않음
  - 인자가 필요하면 선언 가능
  - data type 의 역할은 가능하다. 하지만 객체생성은 불가능하다.
- 객체로 사용하기 위해서는 다른 클래스로 추상 클래스를 상속해서 오버라이드를 하면 된다.
  - 에러가 나는데
    - 상속 받은 클래스를 추상 클래스로 바꿔주거나
    - 강제 오버라이드 하면 된다.

```
package test.main;
import test.mypac.MyWeapon;
import test.mypac.Weapon;

public class MainClass01 {
  public static void main(String[] args) {
    Weapon w1 = new MyWeapon();
    w1.prepare();
    w1.attack();
  }
}
```

- 번외 - Static, nonStatic 사용법
  - 같은 클래스안에서 만들어진 메소드를 호출 하는방법

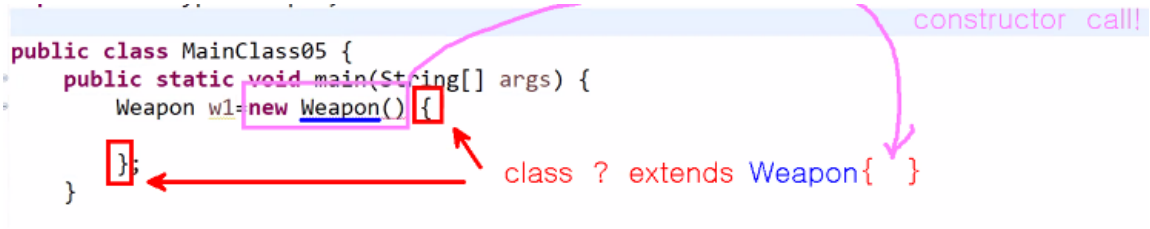
```
package test.main;
public class mainClass02 {
  public static void main(String[] args) {
    }
  public void useWeapon() {
```

```
  }
}
```

```
package test.main;
public class mainClass02 {
  public static void main(String[] args) {
  useWeapon();
  }
  public static void useWeapon() {
  }
}
```

  - 패키지 명까지 적으면 import 할 필요 없다.

- local inner class 지역 이너클래스메소드 안에도 클래스를 정의 할 수 있다. 해당 메소드 안에서만 사용 가능하다
  - Weapon 추상 클래스를 상속받은 inner class
  - main() 메소드는 static 메소드 이기 때문에 main() 메소드에서 사용하려면
  - inner class 도 static 영역에 올라가 있어야 하기 때문에 static 예약어가 필요하다



{ } == Weapon 추상 클래스를 상속받은 클래스다

new Weapon() 는 생성자를 호출하는 것이다.

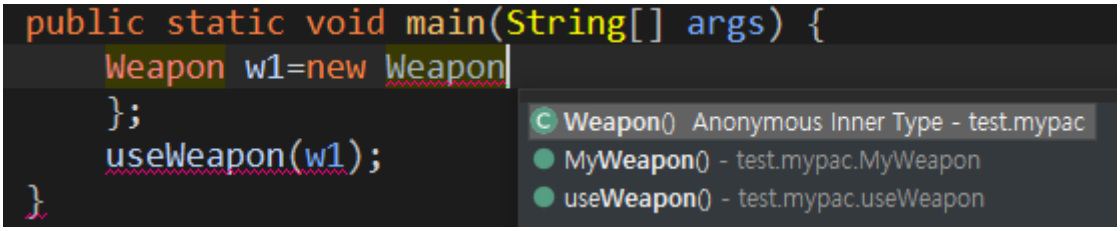
```
package test.main;

import test.mypac.Weapon;

public class MainClass05 {
  public static void main(String[] args) {
    Weapon w1=new Weapon(){
      @Override
      public void attack() {
        System.out.println("난 사실 평화주의자");
      }
    };
    useWeapon(w1);
  }
  public static void useWeapon(Weapon w) {
```

```
    w.prepare();
    w.attack();
  }
}
```

이렇게하면 추상클래스를 간단하게 익명 클래스를 사용해 사용할 수 있다.



인터페이스(Interface)

> 💡 인터페이스는 추상 메소드만 사용할 수 있음

- 생성자가 없다 (단독 객체 생성 불가)
- 필드는 static final 상수만 가질수 있다.
- data type 의 역활을 할수 있다.
- interface type 의 참조값이 필요하면 구현(implements) 클래스를 만들어서
- 객체를 생성해야 한다.
- 클래스 상속은 단일 상속이지만, 인터페이스는 다중 구현이 가능하다

```
package test.mypac;

public interface RemoteController {
  public void up();
  public void down();
}
```

데이터 타입으로 사용할 수 있음

- 인터페이스는 상속(extend)이 아니고 구현(implements)
  - 구현한 class에서 override
  - 다중구현이 가능

- 예약어 final

```
final int num=10;
final String name="Dubu"
final RemoteController r2=new MyRemote();
```

  - 필드나 지역변수에 final 이라는 예약어를 붙이면 변수가 아니고 상수가 된다.
  - 상수는 값이 변경되지 않는다.
  - 따라서 아래의 num, name, r2에 다른 값을 다시 대입할 수 없다.
  - js의 const 와 같다고 생각하면 된다.
  - 관례상 final 지역변수나 필드는 모두 대문자로 표기한다.

```
final double PI=3.141592;
final String GREET_KOR="안녕하세요!";
final String GREET_ENG="hello";
final String GREET_JPN="곤니찌와";
final String GREET_CHN="니하오";
```

  - 숫자를 상수화 시키면 복잡한 숫자에 이름을 부여하는 효과를 준다.
  - 문자열을 상수화 시키면 혼돈하기 쉬운 문자열에 이름을 부여하는 효과를 준다
  - 프로그래머가 복잡한 숫자나 혼돈하기 쉬운 문자열을 쉽게 불러다 쓸 수 있도록 도움을 준다.
- 메소드가 하나만 있다면 화살표 함수처럼 줄여 쓸 수 있다.

```
package test.main;

import test.mypac.Drill;
```

```
public class MainClass04 {
  public static void main(String[] args) {
    useDrill(new Drill() {
      @Override
      public void hole() {
        System.out.println("벽에 구멍을 뚫어요!");
      }
    });
    useDrill(() -> {
      System.out.println("구멍에 벽을 뚫어요!");
    });
  }

  public static void useDrill(Drill d) {
    d.hole();

  }
}
```

```
Drill d2=()->{};

//위는 아래의 줄임 표현이다

Drill d3=new Drill() {
@Override
public void hole() {
  }
};
```