extends Object And final Class, Method

Object Class

- 최상위 Object Class는 모든 Class가 상속받는다.

```
class Myclass {

}
```

↓

```
class MyClass extends Object {
     // extend Object라고 선언을 하지 않더라도 자동으로 선언한다.
}
```

그런데 Object를 상속 받았으면, 다른 Class를 상속받는데 문제가 되지 않나?

→ 모든 Class는 Objec Class를 상속 받았으므로, class MyClass extends AAA { }라고 하더라도,
AAA도 Object Class를 상속 받았으므로, MyClass는 간접적으로 Object Class를 상속받는다.

그러므로 다형성에 의해서, 모든 Class는 Object형 참조변수로 참조가 가능하다.

```
Object obj1 = new MyClass();
Object obj2 = new int[]; // 배열도 Instance 이므로
Object obj3 = "String"; // 문자열도 String Instance 이므로
```

왜? Object Class를 상속받게 했을까? - String Class와 Object Class

```
class Friend {
     String myName;
     public Friend(String name){ myName = name; }

     Public String toString(){ Return "제 이름은 "+myName+"입니다."; }
}

class MainClass{
     public static void main(String[] args){
          Friend fren1 = new Friend("폴길버트");
          Friend fren2 = new Friend("잉베이 맘스틴");
          System.out.println(fren1);
          System.out.println(fren2);
     }
}
```

실행결과
제 이름은 폴길버트입니다.
제 이름은 잉베이 맘스틴입니다.

의문점?
Println Method는 어떻게 Instance를 받을 수 있을까?
→ Friend Class는 Object Class를 상속받았다.
그리고, Java Doc에서 여러 println의 정의 중 한가지를 살펴보면

- println(Object x)
- Prints an Object and then terminate the line.

매개변수형이 Object인데, 다형성에 의해서 모든 Instance를 인자로 전달할 수 있다!

그리고, MyClass의 toString Method는 결국 Object의 toString Method를 Overriding한 것이다.

---

final Class And final Method

```
final class Myclass {
    // Myclass를 누군가가 상속받는 것을 허용하지 않겠다!
}

class YourClass {
    final void yourFunc(int n) {...};
    // 이 Method를 Overriding 하는 것을 허용하지 않겠다!
    // 이 Method가 가려지는 것을 허용하지 않겠다!
}
```
