1️⃣ 생성자 대신 정적 팩토리 매서드를 고려하라

▼ 이름을 가질 수 있다. (동일한 시그니처의 생성자를 두개 가질 수 없다.)

- static 메소드이므로 같은 이름으로 오버라이딩 불가능(동일 이름이어도 새로 추가된 것으로 식별됨)

→ static 메소드는 컴파일 시 메모리에 올라감, 반면 오버라이딩된 메소드는 런타임 시 사용될 메소드를 결정함.

- 직관적인 기능 짐작이 가능해짐

▼ 호출될 때마다 인스턴트를 새로 생성하지 않아도 된다. (Boolean.valueOf)

> 👉 **싱글톤과의 차이??**
> 싱글톤 : getInstance(정적 팩토리 메서드)를 사용해 하나의 객체만 사용하게 만든다.
> 정적팩토리 메서드 : 필요에 따라 항상 새로운 객체를 생성해서 반환할 수도, 하나의 객체만 계속 반환할 수도 있다.
> 출처 : https://7942yongdae.tistory.com/147
> 코드 : https://primayy.tistory.com/60

- **인스턴스 생성의 통제**가 가능하다!

생성자로 new를 해서 생성하게 되면 계속 새로운 instance가 반환되는데, 이게 싫다면 정적 팩토리 메서드로 instance를 불러올때마다 같은 instance를 불러오게 통제가 가능(이것은 return 값을 뭘로 하느냐에 따라 같은 instance만 불러올지, 계속 새로운 instance를 불러올지 달라질 수 있다.)

- 플라이 웨이트 패턴

  폰트 같은 자주 사용하는 세팅값을 미리 넣어놓고 사용 - 똑같이 instance를 통제하는 방법

  java의 String constant pool , Boolean의 valueOf()가 이러하다.

  같은 객체가 자주 사용된다! → 객체를 재사용한다.

▼ 반환 타입의 하위 타입 객체를 반환할 수 있는 능력이 있다.

▼ 입력 매개변수가 따라 매번 다른 클래스의 객체를 반환할 수 있다. (EnumSet)

ENUM

- 특정 상수으로만 변수를 담을 수 있게 강제할 수 있다.(제한 → Type-safety)
- 싱글턴 패턴 구현시 사용되기도 함.
- EnumSet, EnumMap : 매우빠름

▼ 정적 팩터리 메서드를 작성하는 시점에는 반환할 객체의 클래스가 존재하지 않아도 된다. (서비스 제공자 프레임워크)

https://steady-coding.tistory.com/617

▼ 상속이 불가능하다.

만약 하나의 instance만 반환하는 정적 팩토리 메서드의 경우 생성자가 무조건 private 이어야 하기 때문에 상속이 불가능해진다.

이런 경우가 아니라면 상속이 가능하긴하다.

▼ 문서화에 좋지 않다.

```
mvn javadoc:javadoc      ---> 프로젝트의 java doc 생성 명령어
```

자동으로 제공하는 api 문서 생성 시 정적 팩토리 메서드는 구별이 어렵다.

따라서 of, getInstance, create 와 같은 약속된 네이밍을 써준다.

아니면 직접 클래스에서 주석으로 정의해준다. (@see 적극 사용)

리플렉션

JVM의 클래스로더가 클래스를 읽어들임 → 그 정보가 거울에 비친 나(클래스)

anntation 정보를 읽어들이는 것과 같음

클래스를 읽어올 수 있음. → 문자열만 가지고도 그 클래스의 메소드, 필드를 읽어오거나 해당 값을 바꾸는 것까지도 가능