전통적인 for문보다는 for-each 문을 사용하라

최 혁

전통적인 for문을 사용한 실수

```
List<Card> deck = new ArrayList<>();
for (Iterator<Suit> i = suits.iterator(); i.hasNext(); ) {
    for (Iterator<Rank> j = ranks.iterator(); j.hasNext(); ) {
        deck.add(new Card(i.next(), j.next()));
    }
}
```

언뜻보면 문제가 없어 보이지만, iterator.next를 조금만 생각해보면 위 코드가 틀렸다는 사실을 알 수 있다.

i.next가 예상과 다르게 너무 많이 호출된다!

```
List<Card> deck = new ArrayList<>();
for (Iterator<Suit> i = suits.iterator(); i.hasNext(); ) {
    Suit suit = i.next();
    for (Iterator<Rank> j = ranks.iterator(); j.hasNext(); ) {
        deck.add(new Card(suit, j.next()));
    }
}
```

위처럼 변수에 담아서 j가 순회할 동안 한 번만 호출되도록 변경하면 문제를 해결할 수 있다.

for-each 문을 사용한다면?

```
List<Card> deck = new ArrayList<>();
for(Suit suit : suits)
    for(Rank rank : ranks)
        deck.add(new Card(suit, rank));
```

너무 쉽고 직관적인 방법으로 해결할 수 있다!

for-each문을 사용할 수 없는 상황

1. **파괴적인 필터링**: 컬렉션을 순회하면서 선택된 원소를 제거해야 할 때

```
for(String s : iter) {
    if (s.startsWith("A")) {
        iter.remove(); // 이런걸 할 수 없다.
    }
}

while(iter.hasNext()) {
    String s = iter.next();
    if (s.startsWith("A")) {
        iter.remove();
    }
}
```

2. **변형:** 리스트나 배열을 순회하면서 그 원소의 값 일부 혹은 전체를 교체해야 한다면 리스트의 반복자나 배열의 인덱스를 사용해야 한다.

```
String[] numbers = {"one", "two", "three"};
for(String number : numbers) {
    if(number.equals("two"))
        numbers[인덱스를 모르네..?] = "twoooooo";
}
```

3. **병렬 반복:** 여러 컬렉션을 병렬로 순회해야 한다면 각각의 반복자와 인덱스 변수를 사용해 엄격하고 명시적으로 제어해야 한다.

ex) 중첩 for문에서 외부와 내부를 조건에 따라 분기할 경우

정리

- **전통적인 for문은 반복자와 인덱스를 많이 사용하기에 혹시라도 잘못된 변수를 사용하면 오류가 발생할 수 있다.**
- **전통적인 for문과 비교했을 때 for-each(enhanced for statement)문은 명료하고, 유연하고, 버그를 예방해준다!**
- **for-each문을 사용해도 성능은 그대로다.** (for 문이 만들어내는 코드는 사람이 손으로 최적화한 것과 같다)
- **그러므로 for-each문을 사용하지 못하는 상황을 제외하곤, for-each문을 사용하자!**