클래스 불변속성(class invariant)

클래스의 모든 객체들이 일생 동안 만족하는(해야하는) 유효한 상태
생성자는 객체를 유효한 상태로 만들어야 한다 (특별한 이유가 없다면 반드시 그래야 한다)

```
public interface MyIterator {
    int getNext();
    boolean isEnd();
    void remove();
}
```

생성된 iterator 객체는 유효한 상태이어야 하므로, 생성한 즉시 getNext, isEnd 메소드를 호출할 수 있어야 한다.

객체가 특정 상태에서 호출할 수 없는 메소드가 있는지 반드시 따져 봐야 한다.
객체의 상태가 여럿이라면, 상태 다이어그램을 그려서 분석해 보는 것이 바람직하다.

예를 들어 iterator 의 remove 메소드는, 직전에 getNext 메소드가 리턴한 항목을 삭제하는 것이기 때문에,
아직 getNext 메소드를 호출하기 전인 생성한 직후에는 remove 메소드를 호출할 수 없다.

```
public interface MyIterator {
    void setFirst();
    int getNext();
    boolean isEnd();
    void remove();
}
```

생성된 iterator 객체는 유효한 상태이어야 하므로, setFirst 메소드를 호출하지 않고도
getNext, isEnd 메소드를 호출할 수 있도록 구현하는 것이 바람직하다.
즉 생성하자 마자, setFirst 메소드를 호출하기 전에도, 목록의 선두 원소를 리턴할 수 있는 유효한 상태가 되도록 구현하는 것이 좋다.
그렇다면 setFirst 메소드의 용도는, 다시 탐색을 하기 위해서, 선두로 돌아가기 위한 용도이어야 할 것이다.

만약 생성자가 iterator 객체를 유효한 상태를 만들지 않고,
setFirst 메소드를 호출해야 유효한 상태가 되는 것이라면,
왜 생성자에서 유효한 상태를 만들지 않았는지 따져 물어보지 않을 수 없다.

```
class FileInputStream {
    FileInputStream(String path);  // 생성자에서 파일을 오픈할 것인가?
    close();
}
```

```
class FileInputStream {
    open(String path);  // 생성한 후 open 메소드를 호출해서 파일을 열도록 할 것인가?
    close();
```

}
FileInputStream 객체는 파일의 내용을 읽기 위한 객체이다.
파일이 열려있지 않은 상태에서 FileInputStream 객체는 아무 쓸모가 없으니,
파일이 열려잇지 않은 상태의 FileInputStream 객체는 유효한 상태라고 말하기 힘들다.
생성자에서 파일을 오픈하지 않을 이유가 없다.

```
class Connection {
    Connection(String address);  // 생성자에서 통신 연결을 할 것인가?
    close();
}
```

```
class Connection {
    connect(String address); // 생성한 후 connection 메소드를 호출해서 연결하도록 할 것인가?
    close();
}
```

통신 연결 연결이 쉽지 않아서 여러차례 연결 시도를 해야 할 수도 있고,
통신 도중 갑자기 끊여서 다시 연결해야할 필요도 있다.
따라서 connect 메소드가 필요하다.

통신 연결은 시간이 오래 걸리고, 연결 실패 확률도 높다.
생성자에서는 성공확률이 매우 높은 초기화만 구현하는 것이 바람직하다.

통신 연결되지 않은 상태의 Connection 객체도 유효한 상태일 수 있다.
(예: 상대편의 연결 요청을 기다리는 상태)

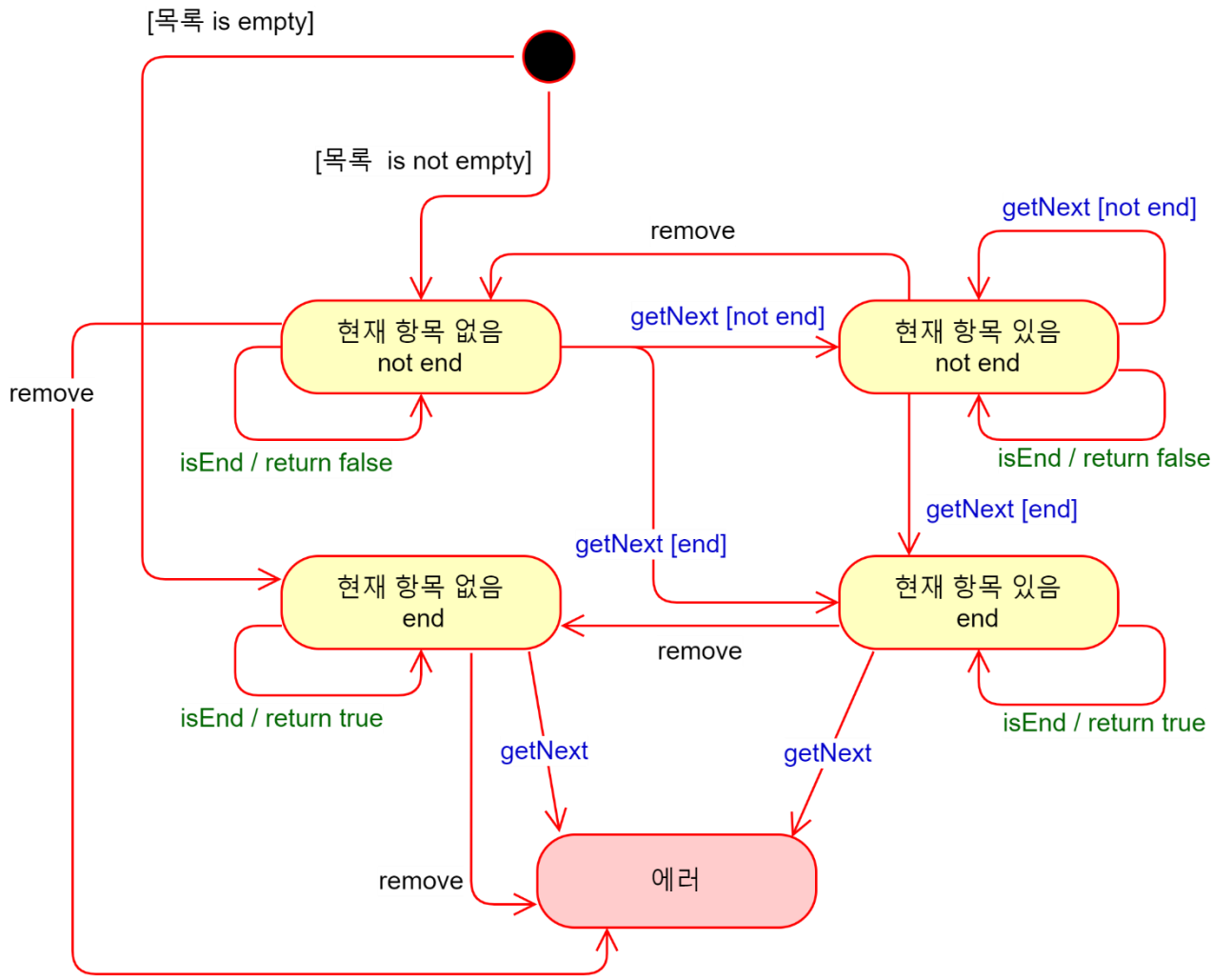
전제 조건 (precondition)

```
public interface MyIterator {
     int getNext();
     boolean isEnd();
     void remove();
}
```

iterator 의 remove 메소드는, 직전에 getNext 메소드가 리턴한 항목을 삭제하는 것이기 때문에,
  아직 getNext 메소드를 호출하기 전인, iterator를 생성한 직후에는 remove 메소드를 호출할 수 없다.
  그리고 remove 메소드를 이미 호출한 후 getNext 메소드를 호출하기 전에도
  remove 메소드를 호출할 수 없다.

remove 메소드 전제조건 정리
  getNext를 호출한 후 remove 메소드를 호출하지 않았는가?

상태 다이어그램


[목록 is empty]
[목록 is not empty]
remove
getNext [not end]
현재 항목 없음
not end
getNext [not end]
현재 항목 있음
not end
remove
isEnd / return false
isEnd / return false
getNext [end]
getNext [end]
현재 항목 없음
end
현재 항목 있음
end
remove
isEnd / return true
isEnd / return true
getNext
getNext
remove
에러