1. 예외에 대한 설명 중 틀린 것은 무엇입니까? 답:4

➊ 예외는 사용자의 잘못된 조작, 개발자의 잘못된 코딩으로 인한 프로그램 오류를 말한다.

➋ RuntimeException의 하위 예외는 컴파일러가 예외 처리 코드를 체크하지 않는다.

➌ 예외는 try-catch 블록을 사용해서 처리된다.

➍ 자바 표준 예외만 프로그램에서 처리할 수 있다.

2. try-catch-finally 블록에 대한 설명 중 틀린 것은 무엇입니까? 답:3

➊ try { } 블록에는 예외가 발생할 수 있는 코드를 작성한다.

➋ catch { } 블록은 try { } 블록에서 발생한 예외를 처리하는 블록이다.

➌ try { } 블록에서 return 문을 사용하면 finally { } 블록은 실행되지 않는다.

➍ catch { } 블록은 예외의 종류별로 여러 개를 작성할 수 있다.

3. throws에 대한 설명으로 틀린 것은 무엇입니까? 답:4

➊ 생성자나 메소드의 선언 끝 부분에 사용되어 내부에서 발생된 예외를 떠넘긴다.

➋ throws 뒤에는 떠넘겨야 할 예외를 쉼표(,)로 구분해서 기술한다.

➌ 모든 예외를 떠넘기기 위해 간단하게 throws Exception으로 작성할 수 있다.

➍ 새로운 예외를 발생시키기 위해 사용된다.

4. throw에 대한 설명으로 틀린 것은 무엇입니까? 답:2

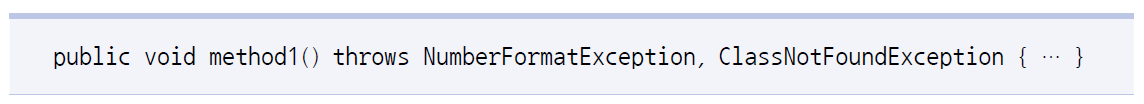
➊ 예외를 최초로 발생시키는 코드이다.

➋ 예외를 호출한 곳으로 떠넘기기 위해 메소드 선언부에 작성된다.

➌ throw로 발생된 예외는 일반적으로 생성자나 메소드 선언부에 throws로 떠넘겨진다.

➍ throw 키워드 뒤에는 예외 객체 생성 코드가 온다.

5. 메소드가 다음과 같이 선언되어 있습니다. 잘못된 예외 처리를 선택하세요. 답:3



➊ try { method1( ); } catch (Exception e) { }

➋ void method2( ) throws Exception { method1( ); }

➌ try { method1( ); }

catch (Exception e) { }

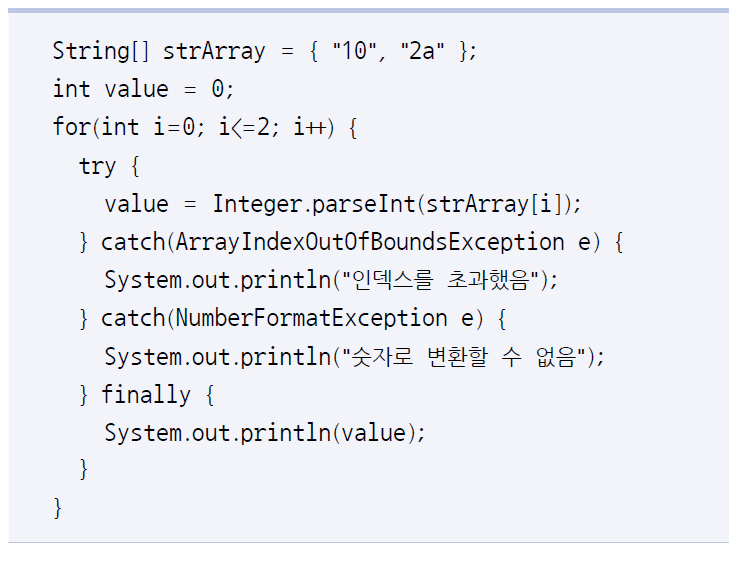
catch (ClassNotFoundException e) { }

➍ try { method1( ); }

catch (ClassNotFoundException e) { }

catch (NumberFormatException e) { }

6. 다음 코드가 실행되었을 때 출력 결과를 예상하고 작성해보세요.



10

숫자로 변환할 수 없음

10

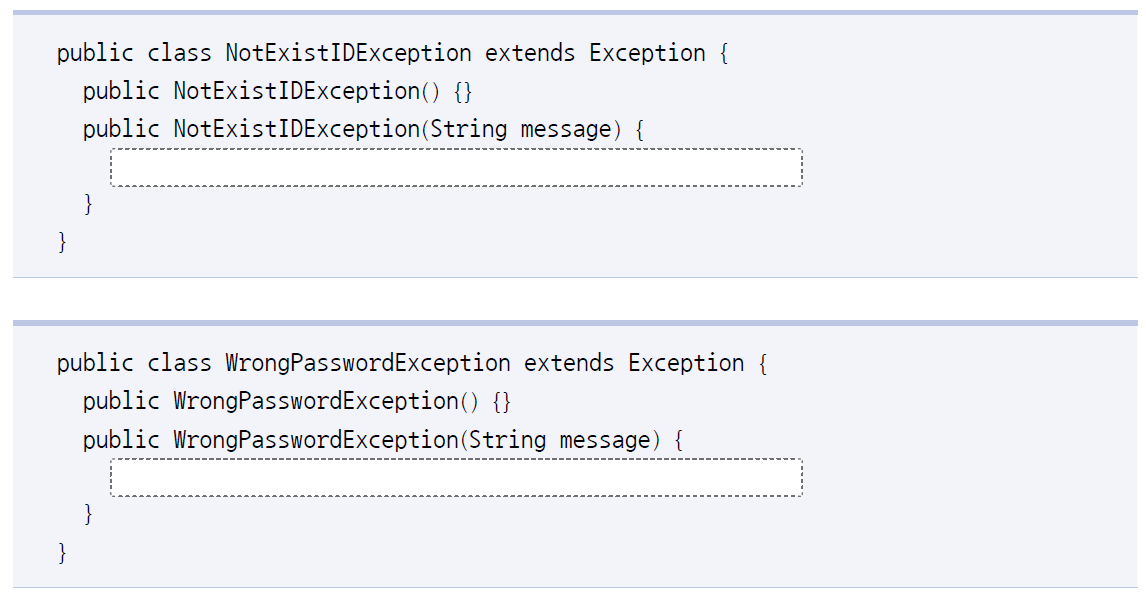
인덱스를 초과했음

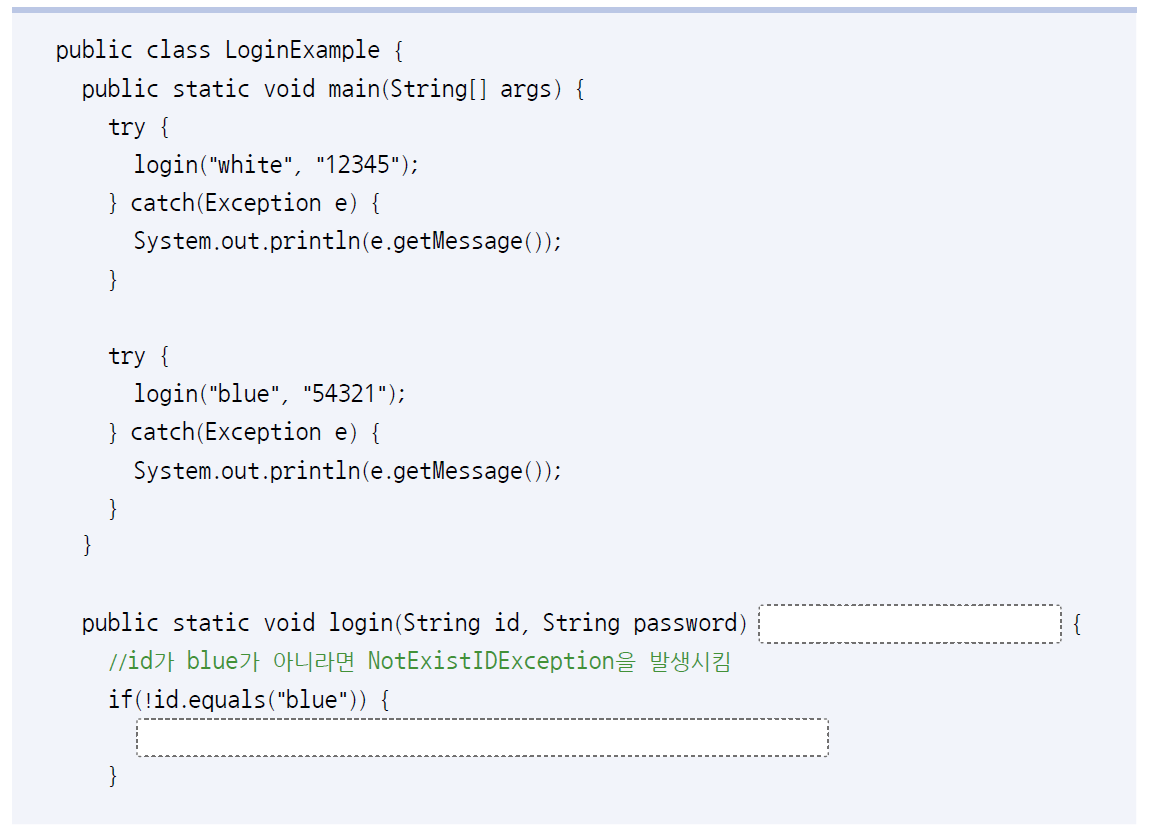
10

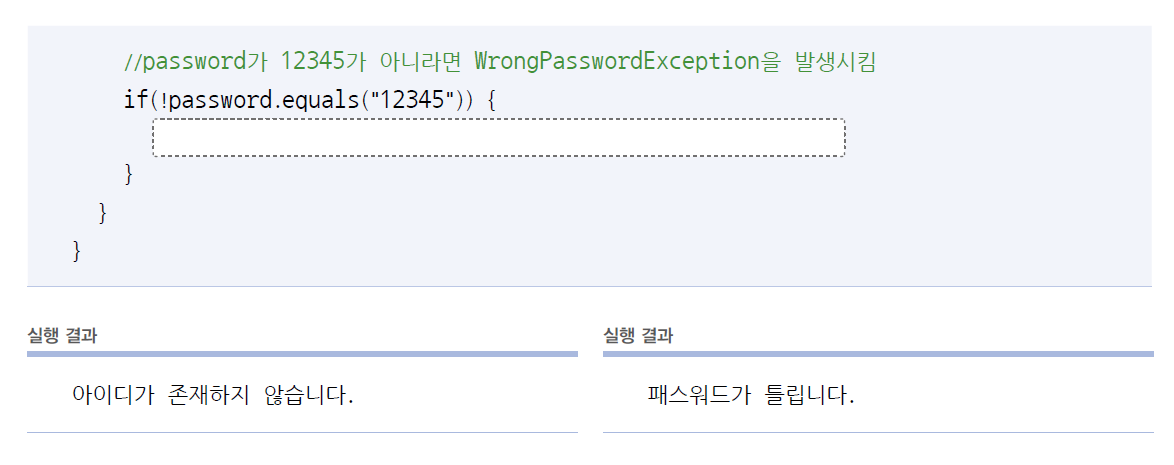
7. login( ) 메소드에서 존재하지 않는 ID를 입력하면 NotExistIDException을 발생시키고, 잘못된

패스워드를 입력하면 WrongPasswordException을 발생시키려고 합니다. 다음 LoginExample

의 실행 결과를 보고 빈칸을 채워보세요







위에서 부터

super(message);

super(message);

throws NotExistIDException, WrongPasswordException

throw new NotExistIDException("아이디가 존재하지 않습니다.");

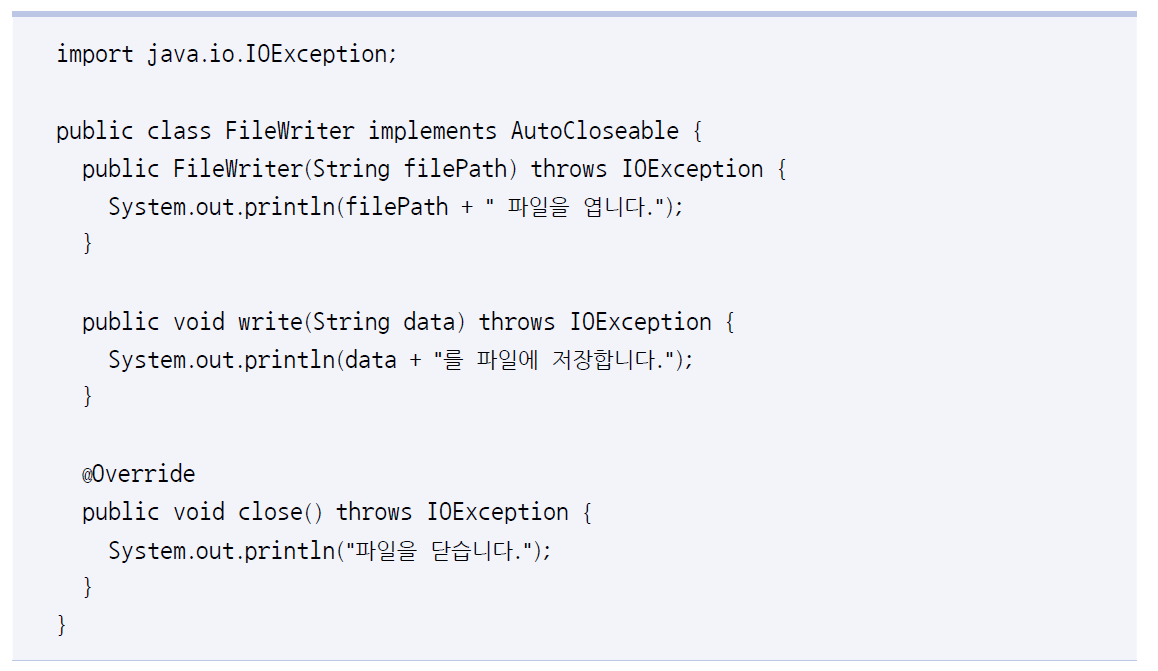
throw new WrongPasswordException("패스워드가 틀립니다.");

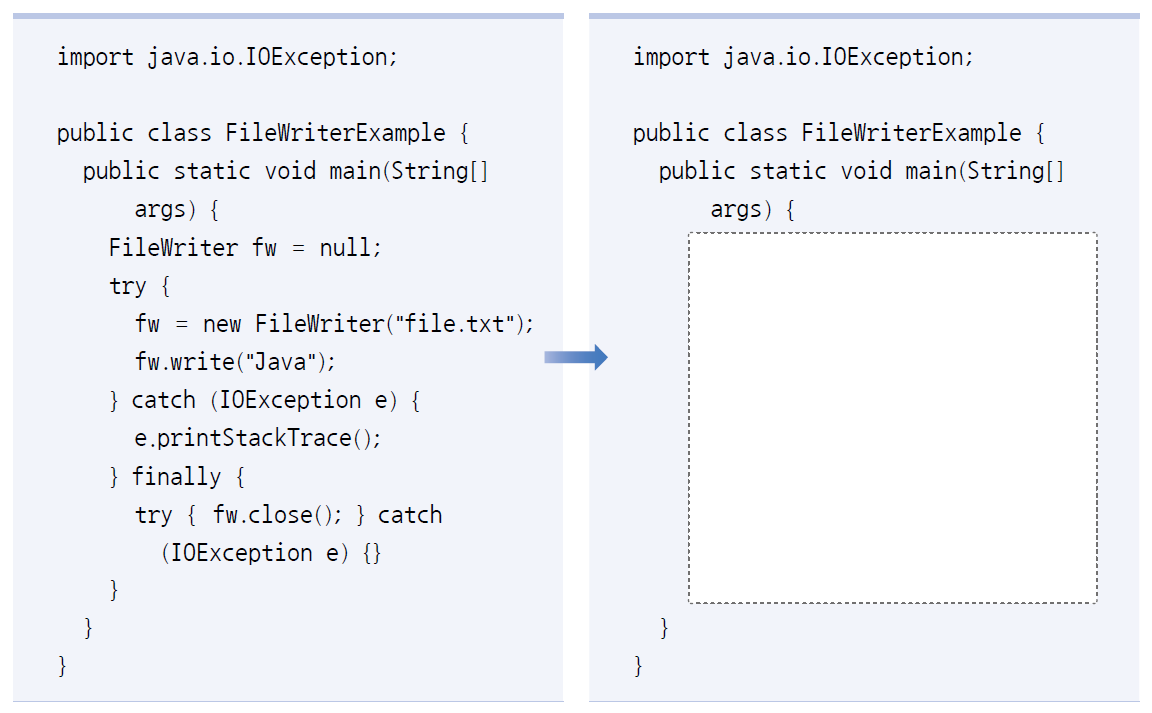
8. FileWriter는 파일을 열고 데이터를 저장하는 클래스입니다. 예외 발생 여부와 상관 없이 마지막

에는 close ( ) 메소드를 실행해서 파일을 닫아주려고 합니다. 왼쪽 코드는 try-catch-finally를 이

용해서 작성한 코드로, 리소스 자동 닫기를 이용하도록 수정하고 싶습니다. 수정한 코드를 오른쪽에

작성해보세요.





try( FileWriter fw = new FileWriter("file.txt") ) {

fw.write("Java");

} catch (IOException e) {

e.printStackTrace();

}