try-with-resources란?

try-with-resources 사용법 예시와 try-with-resources를 사용해야 하는 이유

- 조민규 -

자원 반납하기

- try-catch-finally (Java7이전)
- try-with-resources(Java7부터 가능)

Java7 이전의 자원 반납

- 반납이 필요한 자원들은 Closeable 인터페이스를 구현하고 있음
- try-catch-finally에서 null 검사와 함께 close를 호출
- 여러가지 단점들이 존재

Java7 이전의 자원 반납

```
public static void main(String args[]) throws IOException {
    FileInputStream is = null;
    BufferedInputStream bis = null;
    try {
        is = new FileInputStream("file.txt");
        bis = new BufferedInputStream(is);
        int data = -1;
        while((data = bis.read()) != -1){
            System.out.print((char) data);
        }
    } finally {
        // close resources
        if (is != null) is.close();
        if (bis != null) bis.close();
    }
}
```

Java7 이전의 자원 반납

- 자원 반납에 의해 코드가 복잡해짐
- 작업이 번거로움
- 실수로 자원을 반납하지 못하는 경우 발생
- 에러로 자원을 반납하지 못하는 경우 발생
- 에러 스택 트레이스가 누락되어 디버깅이 어려움

Java7 이후의 자원 반납

- try-with-resources가 자동으로 close를 호출
- AutoCloseable 인터페이스를 구현하고 있어야 함
- 앞선 문제점을 완벽하게 극복

Java7 이후의 자원 반납

```
public static void main(String args[]) throws IOException {
    try (FileInputStream is = new FileInputStream("file.txt"); BufferedInputStream bis = new BufferedInputStream(is))
        int data;
        while ((data = bis.read()) != -1) {
            System.out.print((char) data);
        }
    }
}
```

Java7 이후의 자원 반납

- 코드를 간결하게 만들 수 있음
- 번거로운 자원 반납 작업을 하지 않아도 됨
- 실수로 자원을 반납하지 못하는 경우를 방지할 수 있음
- 에러로 자원을 반납하지 못하는 경우를 방지할 수 있음
- 에러 스택 트레이스가 누락되어 디버깅이 어려움

에러 스택 트레이스가 누락되어 디버깅이 어려움

```
public class MangKyuResource implements AutoCloseable {

    @Override
    public void close() throws RuntimeException{
        System.out.println("close");
        throw new IllegalStateException();
    }

    public void hello() {
        System.out.println("hello");
        throw new IllegalStateException();
    }
}
```

에러 스택 트레이스가 누락되어 디버깅이 어려움

```
public void temp1() {
    MangKyuResource mangKyuResource = null;

    try {
        mangKyuResource = new MangKyuResource();
        mangKyuResource.hello();
    } finally {
        if (mangKyuResource != null) {
            mangKyuResource.close();
        }
    }
}
```

에러 스택 트레이스가 누락되어 디버깅이 어려움

- hello에 의한 에러 스택 트레이스가 누락됨

```
hello
close

java.lang.IllegalStateException
        at com.example.testing.composition.MangKyuResource.close(MangKyuResource.java:8)
        at com.example.testing.composition.SteakTest.temp2(SteakTest.java:49)
```

에러 스택 트레이스가 누락되어 디버깅이 어려움

```
public void temp2() {
    try (MangKyuResource mangKyuResource = new MangKyuResource()) {
        mangKyuResource.hello();
    }
}
```

에러 스택 트레이스가 누락되어 디버깅이 어려움

- hello에 의한 에러 스택 트레이스까지 보임

```
hello
close

java.lang.IllegalStateException
        at com.example.testing.composition.MangKyuResource.hello(MangKyuResource.java:13)
        at com.example.testing.composition.SteakTest.temp4(SteakTest.java:67)
    ... 생략

        Suppressed: java.lang.IllegalStateException
                at com.example.testing.composition.MangKyuResource.close(MangKyuResource.java:8)
                at com.example.testing.composition.SteakTest.temp4(SteakTest.java:68)
```

에러로 자원을 반납하지 못하는 경우를 방지할 수 있음

```
public void tryCatchFinallyExample() {
    MangKyuResource resource1 = null;
    MangKyuResource resource2 = null;

    try {
        resource1 = new MangKyuResource();
        resource2 = new MangKyuResource();
        resource1.hello();
        resource2.hello();
    } finally {
        if (resource1 != null) {
            resource1.close();
        }

        if (resource2 != null) {
            resource2.close();
        }
    }
}
```

에러로 자원을 반납하지 못하는 경우를 방지할 수 있음

- Resource2가 반납되지 않음

```
hello
hello
close
```

에러로 자원을 반납하지 못하는 경우를 방지할 수 있음

```
public void tryWithResourcesExample() {
    try (MangKyuResource resource1 = new MangKyuResource(); MangKyuResource resource2 = new MangKyuResource()) {
        resource1.hello();
        resource2.hello();
    }
}
```

에러로 자원을 반납하지 못하는 경우를 방지할 수 있음

- Resource1, Resource2까지 모두 반납됨

```
hello
hello
close
close
```