우리에프아이에스

우리FISA 2기 클라우드 서비스 개발

2차 기술세미나

예외처리

나경률 | 2024.02.19

목차

01 프로그래밍 오류

1. 논리적 에러 - 실행은 되지만 의도와 다르게 동작하는 것

2. 컴파일 에러 - 컴파일 단계에서 오류를 발견하면 컴파일러가 에러 메시지를 출력하는 것

3. 런타임 에러 - 프로그램이 동작하는 와중에 발생하는 것

01 에러 vs 예외

에러

프로그램 코드에 의해 수습될 수 없는 심각한 오류

예외

프로그램 코드에 의해 수습될 수 있는 미약한 오류

01 예외의 종류

일반 예외 (Checked Exception)

컴파일러가 예외 처리 코드 여부를 검사하는 예외

IOException,
FileNotFoundException

실행 예외 (Unchecked Exception)

컴파일러가 예외 처리 코드 여부를 검사하지 않는 예외

NullPointerException,
ArrayIndexOutOfBoundsException

?

01 예외는 언제 발생하는가

```java
import java.io.*;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("noFile"));
        br.readLine();
        br.close();
    }
}
```

```
Exception in thread "main" java.io.FileNotFoundException Create breakpoint : noFile (No such file or directory)
    at java.base/java.io.FileInputStream.open0(Native Method)
    at java.base/java.io.FileInputStream.open(FileInputStream.java:216)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
    at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111)
    at java.base/java.io.FileReader.<init>(FileReader.java:60)
    at Main.main(Main.java:5)
```

01 예외는 언제 발생하는가

```java
public class Main {
    public static void main(String[] args) {
        String data = null;
        System.out.println(data.toString());
    }
}
```

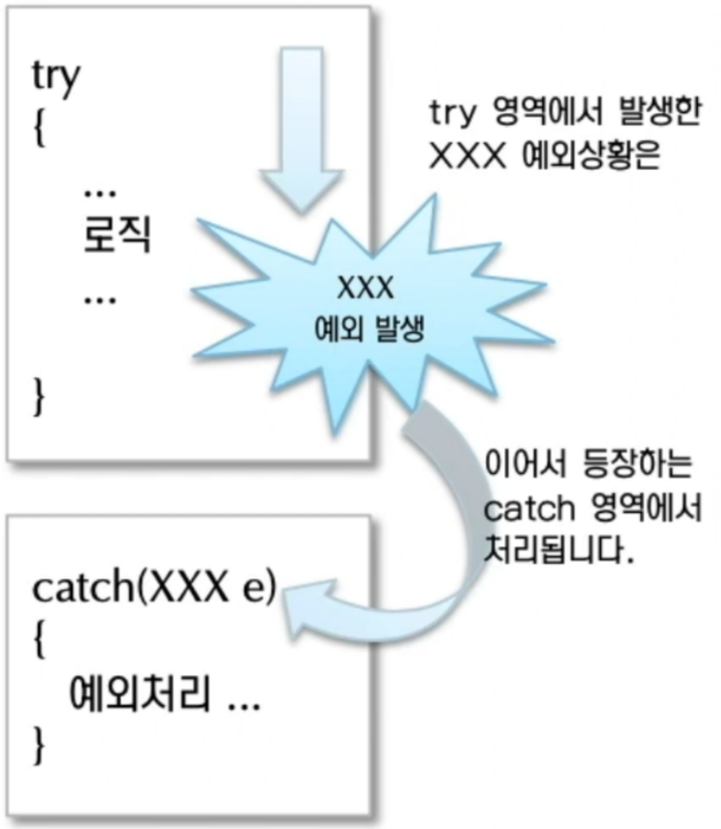
```
Exception in thread "main" java.lang NullPointerException Create breakpoint : Cannot invoke "String.toString()" because "data" is null
    at Main.main(Main.java:9)
```

01 예외는 언제 발생하는가

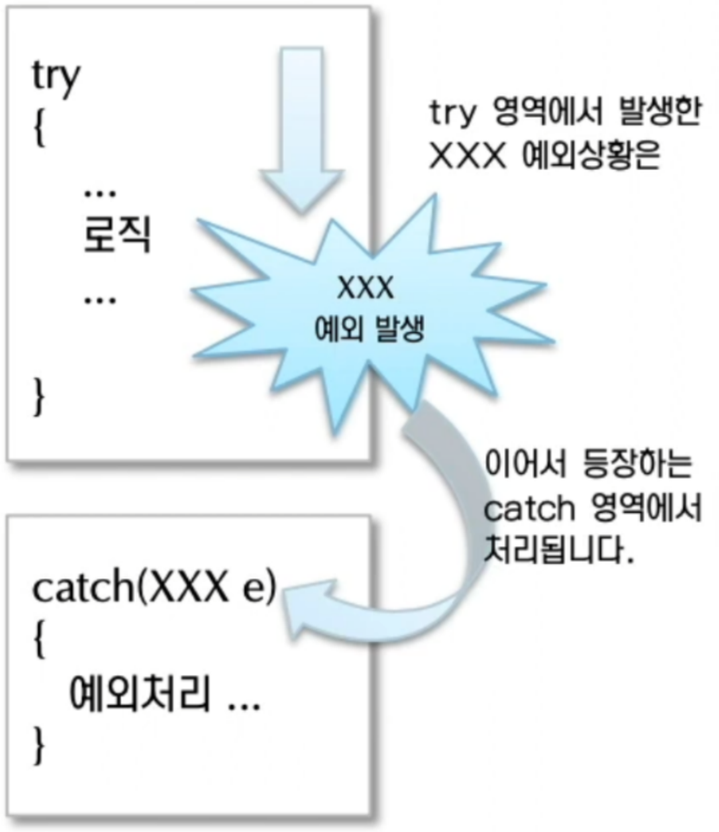
```java
public class Main {
    public static void main(String[] args) {
        int num = 5 / 0;
    }
}
```

```
Exception in thread "main" java.lang.ArithmeticException Create breakpoint : / by zero
    at Main.main(Main.java:12)
```

02 try-catch-finally



02 try-catch-finally




```java
import java.io.*;

public class Main {
    public static void main(String[] args) throws IOException {
        int num;
        try {
            num = 5 / 0;
        } catch(ArithmeticException e) {
            num = -1;
        } finally {
            System.out.println("예외처리 여부와 관계없이 실행");
        }
    }
}
```

02 throw, throws

throw

메서드 내에서 예외를 발생시키는 데 사용

```
throw new FoolException()
```

throws

메서드 선언부에서 사용되며, 해당 메서드가 처리하지 않은 예외를 호출자에게 전달

```
public void sayNick(String nick)
throws FoolException
```

02 예외 종류

```java
public class Sample {
    public void sayNick(String nick) {
        if("바보".equals(nick)) {
            return;
        }
        System.out.println("당신의 별명은 "+ nick +" 입니다.");
    }

    public static void main(String[] args) {
        Sample sample = new Sample();
        sample.sayNick("바보");
        sample.sayNick("야호");
    }
}
```

```
당신의 별명은 야호 입니다.
```

02 RuntimeException

```
class FoolException extends RuntimeException {
}

public class Sample {
    public void sayNick(String nick) {
        if("바보".equals(nick)) {
            throw new FoolException();
        }
        System.out.println("당신의 별명은 "+nick+" 입니다.");
    }

    public static void main(String[] args) {
        Sample sample = new Sample();
        sample.sayNick("바보");
        sample.sayNick("야호");
    }
}
```

```
Exception in thread "main" FoolException
    at Sample.sayNick(Sample.java:7)
    at Sample.main(Sample.java:14)
```

02 Exception

```
class FoolException extends Exception {
}

public class Sample {
    public void sayNick(String nick) {
        try {
            if("바보".equals(nick)) {
                throw new FoolException();
            }
            System.out.println("당신의 별명은 "+nick+" 입니다.");
        }catch(FoolException e) {
            System.err.println("FoolException이 발생했습니다.");
        }
    }

    public static void main(String[] args) {
        Sample sample = new Sample();
        sample.sayNick("바보");
        sample.sayNick("야호");
    }
}
```

```
FoolException이 발생했습니다.
당신의 별명은 야호 입니다.
```

02 예외 던지기

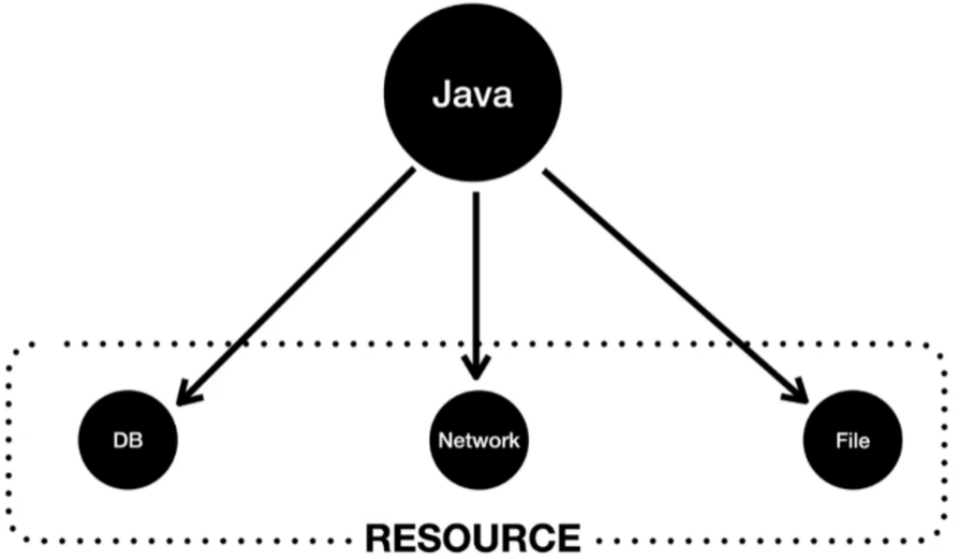
```
class FoolException extends Exception {
}

public class Sample {
    public void sayNick(String nick) throws FoolException {
        if("바보".equals(nick)) {
            throw new FoolException();
        }
        System.out.println("당신의 별명은 "+ nick +" 입니다.");
    }

    public static void main(String[] args) {
        Sample sample = new Sample();
        try {
            sample.sayNick("바보");
            sample.sayNick("야호");
        } catch (FoolException e) {
            System.err.println("FoolException이 발생했습니다.");
        }
    }
}
```

```
FoolException이 발생했습니다.
```

02 자바 resource의 예외 처리



02 try/catch 문제점

```java
import java.io.FileWriter;

public class Main {
    public static void main(String[] args) {
        FileWriter file = null;
        file = new FileWriter("data.txt");
        file.write("Hello World");
        file.close();
    }
}
```

```java
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        FileWriter file = null;
        try {
            file = new FileWriter("data.txt");
            file.write("Hello World");
        } catch (IOException e) {
            throw new RuntimeException(e);
        } finally {
            try {
                file.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```

02 try-with-resource

```
try(파일 열기 or 자원 할당) {
    ...
}
```

```java
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        // 파일을 열고 모두 사용되면 자동으로 닫아준다
        try (FileWriter file = new FileWriter("data.txt")) {
            file.write("Hello World");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

02 try-with-resource

AutoCloseable

```
public interface AutoCloseable {
    /**
     * Closes this resource, relinquishing any underlying resources.
     * This method is invoked automatically on objects managed by the
     * {@code try}-with-resources statement
     */
    void close() throws Exception;
}
```

02 사용자 정의 예외

```
class BalanceInsufficientException extends Exception {
    public BalanceInsufficientException() {}
    public BalanceInsufficientException(String message) {
        super(message);
    }
}
```

02 사용자 정의 예외

```
public class Account {
    private long balance;
    public Account(){}
    public long getBalance(){
        return balance;
    }
    public void deposit(int money) {
        balance += money;
    }
    public void withdraw(int money) throws BalanceInsufficientException {
        if(balance < money) {
            throw new BalanceInsufficientException(
                "잔고 부족: " + (money - balance) + "원이 부족"
            );
        }
        balance -= money;
    }
}
```

02 사용자 정의 예외

```java
public class AccountExample {
    public static void main(String[] args) {
        Account account = new Account();
        account.deposit(10000);
        System.out.println("예금: " + account.getBalance() + "원");

        try {
            account.withdraw(20000);
        } catch (BalanceInsufficientException e) {
            String message = e.getMessage();
            System.out.println(message);
            e.printStackTrace();
        }

    }
}
```

```
예금: 10000원
잔고 부족: 10000원이 부족
BalanceInsufficientException: 잔고 부족: 10000원이 부족
    at Account.withdraw(Account.java:12)
    at AccountExample.main(AccountExample.java:8)
```

03 결론

어디서 예외처리를 하는 것이 좋을까?

"예외 처리는 문제를 예방하고 해결하기 위한 필수적인 도구이다." -

우리에프아이에스

2차 기술세미나

THANK YOU FOR ATTENTION