제15장 예외 (Exception)

예외 (Exception)

프로그램을 설계할 때

실행시에 예외 상황이 발생 할 가능성이 있는 것을 예측하여, 사전에 예외 처리가 되도록 할 필요가 있음.

이럴 때 적절한 조치가 없으면 프로그램은 에러가 나며 죽어 버림.

예상 외의 상황에 적절한 조치를 취하는 것을 배운다.

에러의 종류와 대응책

1. 문법 에러 (syntax error)
2. 실행 시 에러 (runtime error)
3. 논리 에러 (logic error)

에러 상황 별 처리

	syntax error	runtime error	logic error
원인	코드의 형식적 오류	실행 중에 예상외의 사태가 발생하여 동작이 중지됨	기술한 처리 내용에 논리적인 오류가 있음
알아 채는 방법	컴파일하면 에러 남	실행하면 도중에 강제 종료 됨	실행하면 예상외의 값이 나옴
해결 방법	컴파일러의 지적을 보고 수정	에러	원인을 스스로 찾아서 해결해야 함

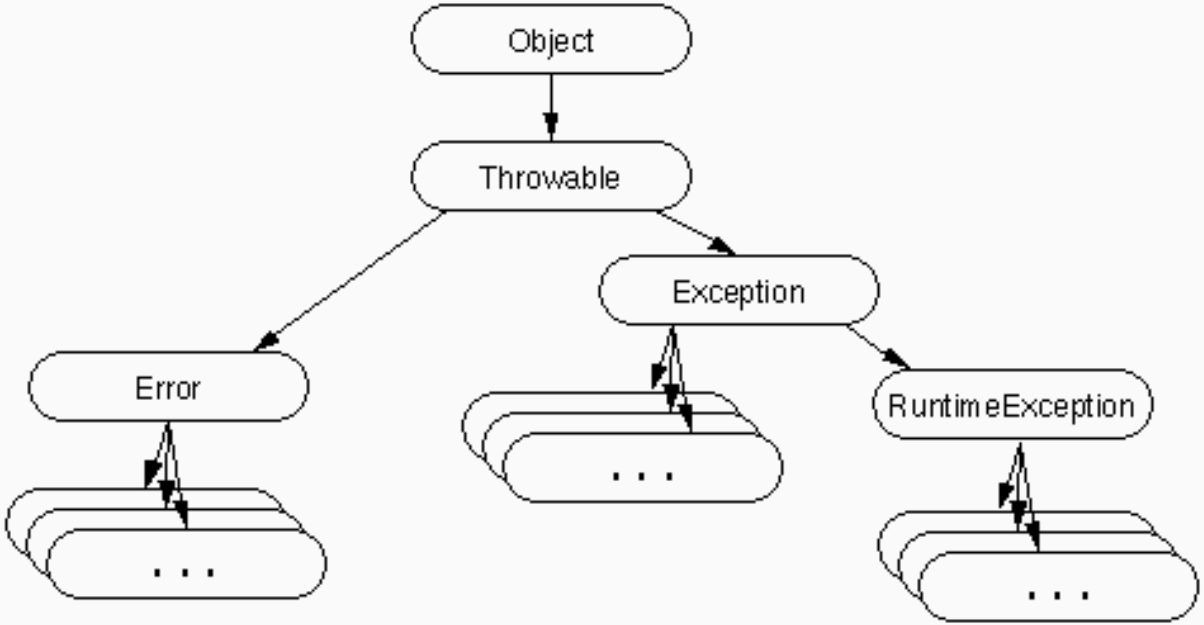
예외적인 상황들

- 메모리 부족
- 파일을 찾을 수 없음
- null 인 변수를 참조 했음

예외 처리의 흐름

```
try {
    // 일반 적인 실행 부분
    Thread.sleep(3000);
} catch (InterruptedException e) {
    // 예외가 발생했을 때 처리를 하는 부분
    System.out.println("에러 발생!. 종료 합니다");
    System.exit(1);
}
```

예외의 종류



15-1 예외처리를 하지 않는 경우


```java
import java.io.FileWriter;

public class Main {
    public static void main(String[] args) {
        // FileWriter 는 IOException을 발생시킬 가능성이 있음
        // try-catch 로 감싸지 않으면 컴파일 에러
        FileWriter fw = new FileWriter("data.txt");
    }
}
```

15-2 try-catch 문으로 Exception 계열 예외의 발생을 준비


```
4  public class Main {
5      public static void main(String[] args) {
6          try {
7              FileWriter fw = new FileWriter("data.txt");
8          } catch (IOException e) {
9              // 예외처리를 하여 프로그램이 강제 종료 되지 않음
10             System.out.println("에러가 발생 했습니다");
11         }
12     }
13 }
```

예외 처리를 상위 메소드에 미루는 방법


```java
public class Main {
    // throws 로 예외 처리를 상위로 미룸
    public static void main(String[] args) throws IOException {
        FileWriter fw = new FileWriter("data.txt");
    }
}
```

예외 인스턴스 이용 방법


```java
public class Main {
    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter("data.txt");
        } catch (IOException e) {
            // 상세한 에러 메세지를 취득하는 방법
            System.out.println("에러 : " + e.getMessage());
            // JVM에서 어떤 경로로 메소드 호출이 일어나서
            // 어디서 에러가 났는지 경로를 출력하는 방법
            e.printStackTrace();
        }
    }
}
```

15-3 예외를 퉁쳐서 처리


```java
import java.io.FileWriter;

public class Main {
    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter("data.txt");
            fw.write("hello!");
            fw.close();
        } catch (Exception e) {     // Exception 계열 전부를 캐치 가능
            System.out.println("뭔가 에러 발생");
        }
    }
}
```

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

public class Main {
    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter("data.txt");
            fw.write("hello!");
        } catch (IOException e) {
            System.out.println("뭔가 에러 발생");
        }
        fw.close();     // try-catch 다음에 close 하면 에러
    }
}
```

finally 로 항상 해야하는 처리


```java
public static void main(String[] args) {
    FileWriter fw = null;
    try {
        fw = new FileWriter("data.txt");
        fw.write("hello!");
    } catch (IOException e) {
        System.out.println("뭔가 에러 발생");
    } finally {
        // 파일, DB접속, 네트워크 접속 등에서는
        // 반드시 finally 로 처리해야 함
        try {
            fw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Java7 이후에는 자동 close() 처리되는 try-catch


```java
// try 다음 괄호 안에서 생성한 파일객체는
// finally 블록에서의 close() 생략 가능
try (FileWriter fw = new FileWriter("data.txt")) {
    fw.write("hello!");
} catch (IOException e) {
    System.out.println("뭔가 에러 발생");
}
```

에러의 전달


```java
public static void subsub() throws IOException {
    // try-catch가 없어도 됨
    FileWriter fw = new FileWriter("data.txt");
}
```

15-5 예외 인스턴스를 자신에게 던지기


```java
public class Person {
    int age;

    public void setAge(int age) {
        if (age < 0) {
            throw new IllegalArgumentException("나이는 음수가 될수 없음. 지정한 값=" + age);
        }
        this.age = age;      // 문제가 없을 경우, 필드에 값을 설정
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        person.setAge(-128);     // 오류 값을 설정하여 예외 발생
    }
}
```

15-6 오리지날 예외 클래스 정의


```java
public class UnsupportedMusicFileException extends Exception {
    // 생성자
    public UnsupportedMusicFileException(String msg) {
        super(msg);
    }
```

```java
public class Main {
    public static void main(String[] args) {
        try {
            // 테스트로 예외를 발생 시킴
            throw new UnsupportedMusicFileException("음악 파일이 아닙니다");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

제 15장의 정리

에러
- syntax error, runtime error, logic error 의 3종류
- 예외처리를 할 때는, runtime error를 대처한다

예외의 종류
- API에는 여러가지 예외적 상황을 표현하는 예외 클래스가 준비되어 있다.
- 예외 클래스는 Error 계열, Exeption 계열, RuntimeException 계열로 크게 나뉜다
- 예외 클래스를 상속하여 오리지날 예외 클래스를 정의할 수 있다.

예외 처리
- try-catch 문을 사용하면, try 블록 내에 예외가 발생했을 때 catch 블록에서 처리가 옮겨진다
- finally 블록으로 나중에 꼭 해야하는 처리를 할 수 있다
- Exception 계열 예외가 발생할 가능성이 있을 때에는 try-catch 문이 필수다
- throw 선언을 하면, 예외 처리를 호출하는 주체에 위임하는 할 수 있다.
- throw 문을 사용하면 임의로 예외를 발생시킬 수 있다

연습문제 15-1

다음과 같은 프로그램을 작성, 실행 후 runtime error 를 발생시키시오.

1. String 형 변수 s를 선언하고, null 을 대입한다
2. s.length() 의 내용을 표시하려고 한다

연습문제 15-2

연습 15-1 에서 작성한 코드를 수정하여, try-catch() 문을 사용하여 예외처리를 하시오. 예외처리에는 다음의 처리를 수행하시오.

1. “NullPointerException 예외를 catch 하였습니다” 를 표시한다
2. “---- stack trace (여기부터) ----” 를 표시한다
3. stack trace 를 표시한다
4. “---- stack trace (여기까지) ----” 를 표시한다

연습문제 15-3

Integer.parseInt() 메소드를 사용하여, 문자열 “Three” 의 변환결과를 int형 변수 i 에 대입하는 코드를 작성하시오.

parseInt() 메소드가 어떤 예외를 발생시킬 가능성이 있는지 API 레퍼런스를 조사하고, 올바른 예외처리를 기술하시오.

연습문제 15-4

기동 직후에 “프로그램 시작" 을 표시하고

IOException 을 임의로 발생시켜 이상 종료 되도록 프로그램을 작성하시오.

연습문제 15-5

Day 3

- Repository 인터페이스를 수정하여 UserRepositoryException을 던지도록 하세요. UserCsvFileRepository 클래스에서는 이 예외를 처리하여 적절한 오류 메시지를 반환하세요.
- UserCsvFileRepository 클래스의 create, read, update, delete 메소드에서 발생하는 예외에 대한 단위 테스트를 작성하세요.