예외처리

예외처리

- 자바에서 말하는 예외
  - 프로그램 실행 중 발생하는 예외적인 상황
  - 단순한 문법 오류가 아닌 실행 중간에 발생하는 정상적이지 않은 상황

```
public class ExceptionCase {
    public static void main(String[] args) {
        int num = 2;
        String str = null;
        if (num % 2 != 0) {
            str = "apple";
        }
        boolean isApple = str.equals("apple");
        System.out.println(isApple);
    }
}
```

  - 위 코드는 문법적인 오류가 없지만 str이 null이기 때문에 equals 메소드 실행 시 NullPointerException 오류 발생
  - 출력 코드는 실행되지 않으며 콘솔을 통해 오류 메시지 확인 가능

예외처리

- 예외처리 : try ~ catch

```
try {
      코드 실행(관찰 영역)
} catch (Exception e) {
      오류 발생 시 처리 영역
}
```

  - try 영역에서 예외 상황이 발생하면 JVM이 예외 종류에 해당하는 Exception 인스턴스 생성 후 catch 구문의 파라메터 e로 전달
  - try ~ catch 사용 시 예외가 처리된 것으로 간주하고 코드 계속 실행

```
public class ExceptionCase {
    public static void main(String[] args) {
        try {
            int num = 2;
            String str = null;
            if (num % 2 != 0)
                str = "apple";
            boolean isApple = str.equals("apple");
        } catch (NullPointerException e) {
            e.printStackTrace();
        }
        System.out.println("나머지 실행");
    }
}
```

예외처리

- 적절한 예외 처리
  - 아래 코드에서 try ~ catch를 사용하여 "끝~!"이 출력되도록 적절한 예외 처리 진행
  - 단, 입력 오류 시 InputMismatchException 인스턴스 발생

```
public class ExceptionCase2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("숫자 입력 : ");
        int n1 = sc.nextInt();  // 예외 발생 가능 : 숫자가 아닌 문자 입력
        System.out.println("입력한 숫자 = " + n1);
        System.out.println("끝~!");
    }
}
```

예외처리

- 둘 이상의 예외를 처리하기

```
public class ExceptionCase3 {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            System.out.print("숫자 입력 : ");
            int num = sc.nextInt();      // 입력 오류 발생 가능
            String str = null;
            if (num % 2 != 0) {
                str = "apple";
            }
            boolean isApple = str.equals("apple");
        } catch (NullPointerException | InputMismatchException e) {
            e.printStackTrace();
        }
        System.out.println("나머지 실행");
    }
}
```

예외처리

- Throwable 클래스와 예외처리의 책임 전가
  - 예외 클래스의 최상위 클래스는 java.lang.Throwable이며 아래의 메소드 존재
    - getMessage( ) : 예외의 원인을 담고 있는 문자열 반환
    - printStackTrace( ) : 예외가 발생한 위치와 호출된 메소드의 정보 출력

```
public class ExceptionMessage {
    public static void main(String[] args) {
        md1(3);
        System.out.println("끝!");
    }
    public static void md1(int n) {
        md2(n, 0);
    }
    public static void md2(int n1, int n2) {
        int r = n1 / n2;        // 예외 발생 지점
    }
}
```

  - 메소드 호출 흐름 : main -> md1 -> md2
  - md1과 md2에서 예외 처리 안하므로 예외처리 책임을 md1으로 넘기며 md1은 다시 main으로 예외처리의 책임 넘김
  - 콘솔의 메시지 확인 시 예외의 발생 및 이동 경로 확인 가능

예외처리

- 따라서 md2에서 발생한 예외를 main 메소드에서 try ~ catch로 처리

```
public class ExceptionMessage2 {
    public static void main(String[] args) {
        try {
            md1(3);         // md1으로부터 예외가 넘어옴
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("끝!");
    }
    public static void md1(int n) {
        md2(n, 0);          // md2로부터 예외가 넘어옴
    }
    public static void md2(int n1, int n2) {
        int r = n1 / n2;        // 예외 발생 지점
    }
}
```

예외처리

- 예외의 종류
  - ArrayIndexOutOfBounds : 배열 접근에 잘못된 인덱스 값 사용
  - ClassCastException : 허용할 수 없는 형 변환을 강제로 진행하는 경우
  - NullPointerException : null이 저장된 참조변수를 대상으로 메소드 호출
  - ArithmeticException : 수학적 연산 오류
- try ~ catch ~ finally
  - try 구문 실행 중 예외 발생 여부에 관계 없이 finally 구문은 무조건 실행

예외처리

- try ~ catch ~ finally
  - try 구문 실행 중 예외 발생 여부에 관계 없이 finally 구문은 무조건 실행

```
public class TryCatchFinally {
    public static void main(String[] args) {
        try {
            int num = 2;
            String str = null;
            if (num % 2 != 0)
                str = "apple";
            boolean isApple = str.equals("apple");
        } catch (NullPointerException e) {
            e.printStackTrace();
        } finally {
            System.out.println("항상 출력됩니다.");
        }
        System.out.println("나머지 실행");
    }
}
```