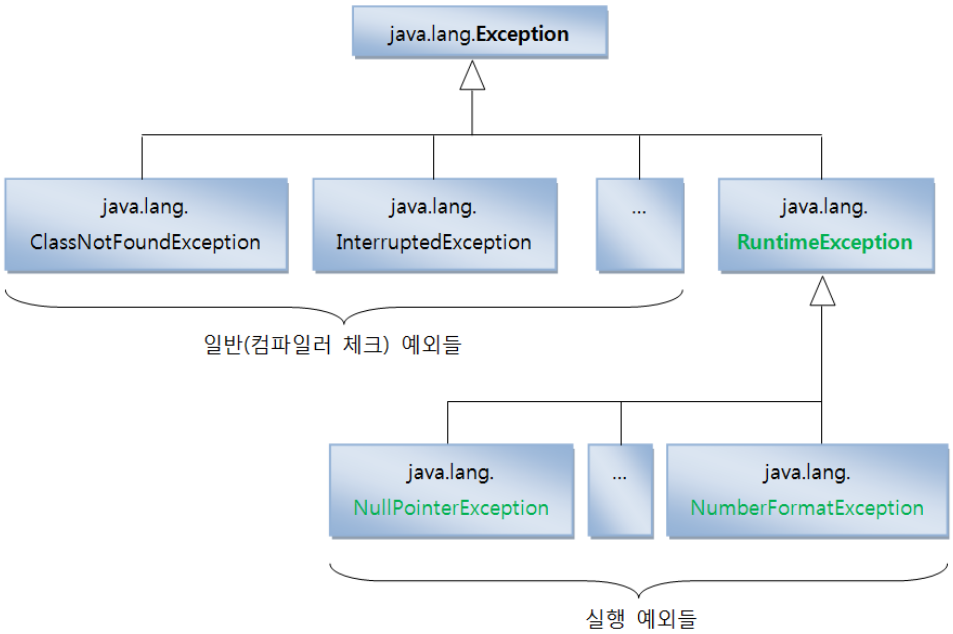
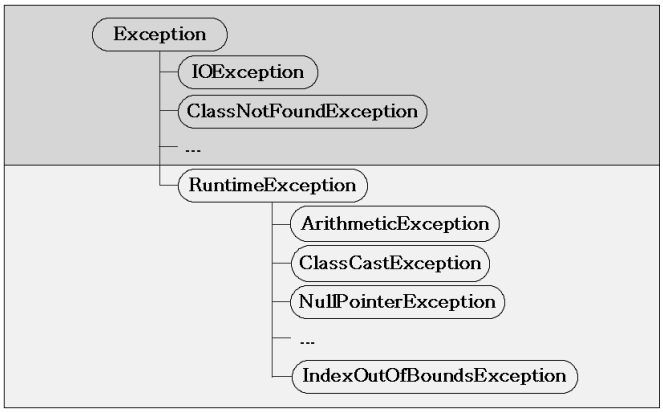
# 예외처리 (Exception Handling)

- ❖ 오류의 종류
  - ▪ 에러(Error)
    - 하드웨어의 잘못된 동작 또는 고장으로 인한 오류
    - 에러가 발생되면 프로그램 종료
    - 정상 실행 상태로 돌아갈 수 없음
  - ▪ 예외(Exception)
    - 사용자의 잘못된 조작 또는 개발자의 잘못된 코딩으로 인한 오류
    - 예외가 발생되면 프로그램 종료
    - 예외 처리 추가하면 정상 실행 상태로 돌아갈 수 있음

- ❖ 예외의 종류
  - ▪ 일반(컴파일 체크) 예외(Exception)
    - 예외 처리 코드 없으면 컴파일 오류 발생
  - ▪ 실행 예외(RuntimeException)
    - 예외 처리 코드를 생략하더라도 컴파일이 되는 예외
    - 경험 따라 예외 처리 코드 작성 필요





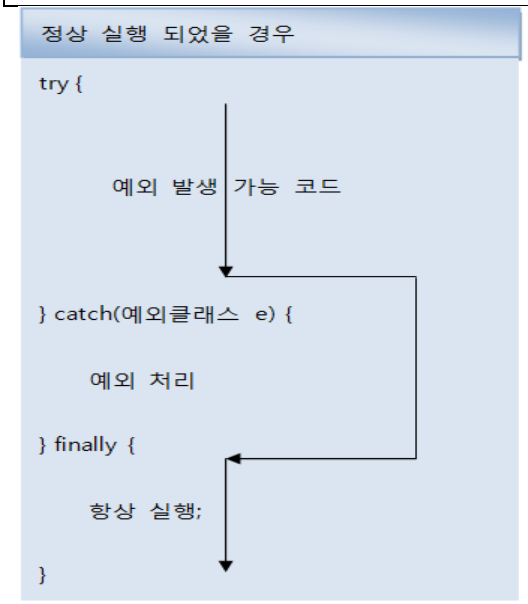

[그림8-2] Exception클래스와 RuntimeException클래스 중심의 상속계층도

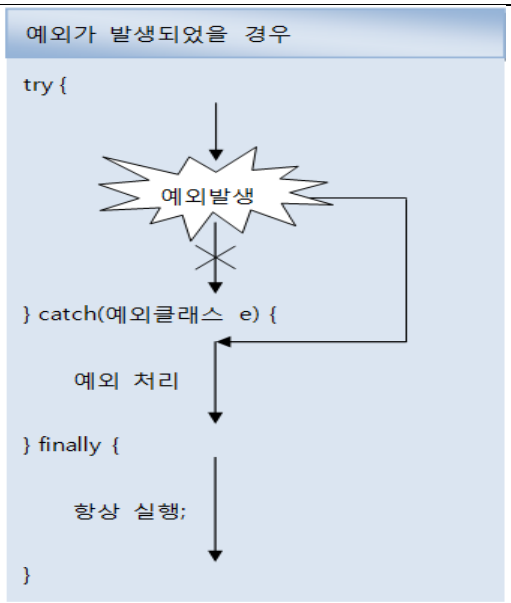
### 실행 예외 예제

NullPointerException

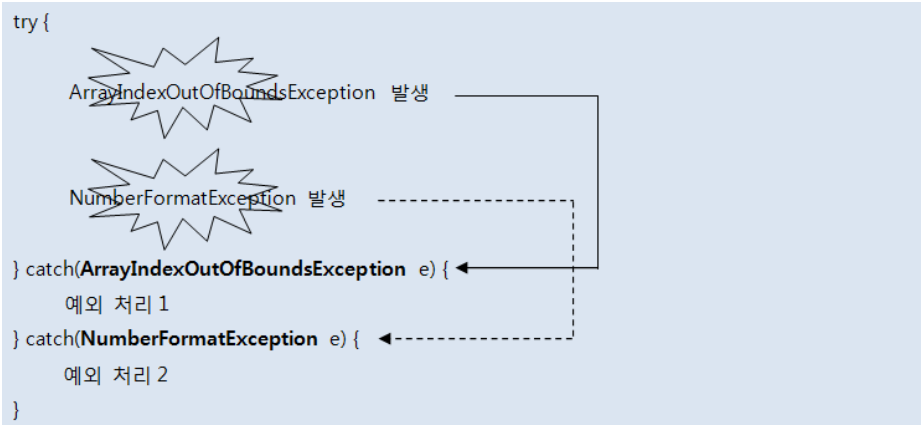
```
String data = null;
System.out.println(data.toString());
```

- ❖ 예외 처리 코드
  - 예외 발생시 프로그램 종료 막고, 정상 실행 유지할 수 있도록 처리
    - 일반 예외: 반드시 작성해야 컴파일 가능
    - 실행 예외: 컴파일러가 체크해주지 않으며 개발자 경험 의해 작성
  - try – catch – finally 블록 이용해 예외 처리 코드 작성




- ❖ 다중 catch
  - 예외 별로 예외 처리 코드 다르게 구현



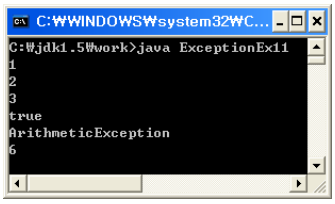

### 실습예제1

**[예제8-11]**/ch8/ExceptionEx11.java

```
class ExceptionEx11 {
   public static void main(String args[]) {
        System.out.println(1);
        System.out.println(2);
        try {
            System.out.println(3);
            System.out.println(0/0);
            System.out.println(4);        // 실행되지 않는다.
        } catch (ArithmeticException ae)  {
            if (ae instanceof ArithmeticException)
                System.out.println("true");
                System.out.println("ArithmeticException");
        } catch (Exception e) {
            System.out.println("Exception");
        }   // try-catch의 끝
        System.out.println(6);
    }   // main메서드의 끝
}
```

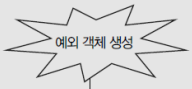
0으로 나눠서 ArithmeticException을 발생시킨다.

ArithmeticException을 제외한 모든 예외가 처리된다.




❖ printStackTrace()

- 예외 발생 코드 추적한 내용을 모두 콘솔에 출력
- 프로그램 테스트하면서 오류 찾을 때 유용하게 활용

```
try {
        

} catch(예외클래스  e) {
    //예외가 가지고 있는 Message 얻기
    String message = e.getMessage();

    //예외의 발생 경로를 추적
    e.printStackTrace();
}
```

**실습예제2**

**[예제8-12]/ch8/ExceptionEx12.java**

```
class ExceptionEx12 {
    public static void main(String args[]) {
        System.out.println(1);
        System.out.println(2);
        try {
            System.out.println(3);
            System.out.println(0/0); // 예외발생!!!
            System.out.println(4);   // 실행되지 않는다.
        } catch (ArithmeticException ae)  {
            ae.printStackTrace();
            System.out.println("예외메시지 : " + ae.getMessage());
        }   // try-catch의 끝
        System.out.println(6);
    }   // main메서드의 끝
}
```

참조변수 ae를 통해, 생성된 ArithmeticException인 스턴스에 접근할 수 있다.

0으로 나눌 경우 예외 발생 예제 코드 (정상적으로 동작하기 위해 try~catch 필요함)

**실습예제3**

```
import java.text.MessageFormat;
import java.util.Scanner;

public class ExceptionEx1 {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in); // 매개변수로 스트림을 요구한다.
                System.out.print("첫번째 숫자: ");
                String s1 = scanner.nextLine();
                System.out.println("두번째 숫자: ");
                String s2 = scanner.nextLine();

                int v1 = Integer.parseInt(s1);
                int v2 = Integer.parseInt(s2);

                int result = v1 / v2;

                System.out.println(MessageFormat.format("{0} 나누기 {1}는 {2}입니다. ", v1, v2, result));
        }
}
```

배열의 범위를 벗어날 경우 예외처리. 실습해 보시오.

**실습예제4**

```
public class ArrayException {
	public static void main (String[] args) {
		int[] intArray = new int[5];
		intArray[0] = 0;
		try {
			for (int i = 0; i < 5; i++) {
				intArray[i+1] = i+1 + intArray[i]; // i=4인 경우 예외 발생
				System.out.println("intArray["+i+"]"+"="+intArray[i]);
			}
		}
		catch (ArrayIndexOutOfBoundsException e) {
			System.out.println("배열의 인덱스가 범위를 벗어났습니다.");
		}
	}
}
```

```
intArray[0]=0
intArray[1]=1
intArray[2]=3
intArray[3]=6
배열의 인덱스가 범위를 벗어났습니다.
```

숫자 변형 오류에 따른 예외처리 예제

```
public class NumException {
	public static void main (String[] args) {
		String[] stringNumber = {"23", "12", "998", "3.141592"};
		try {
			for (int i = 0; i < stringNumber.length; i++) {
				int j = Integer.parseInt(stringNumber[i]); // "3.141592"에서 예외 발생
				System.out.println("숫자로 변환된 값은 " + j);
			}
		}
		catch (NumberFormatException e) {
			System.out.println("정수로 변환할 수 없습니다.");
		}
	}
}
```

```
숫자로 변환된 값은 23
숫자로 변환된 값은 12
숫자로 변환된 값은 998
정수로 변환할 수 없습니다.
```

수고했습니다. 아래 문제를 코딩해 보세요.

**문제**

1부터 10까지 반드시 1초 간격으로 출력되도록 하세요. (Hint, Thread.sleep(…) ) 사용하세요