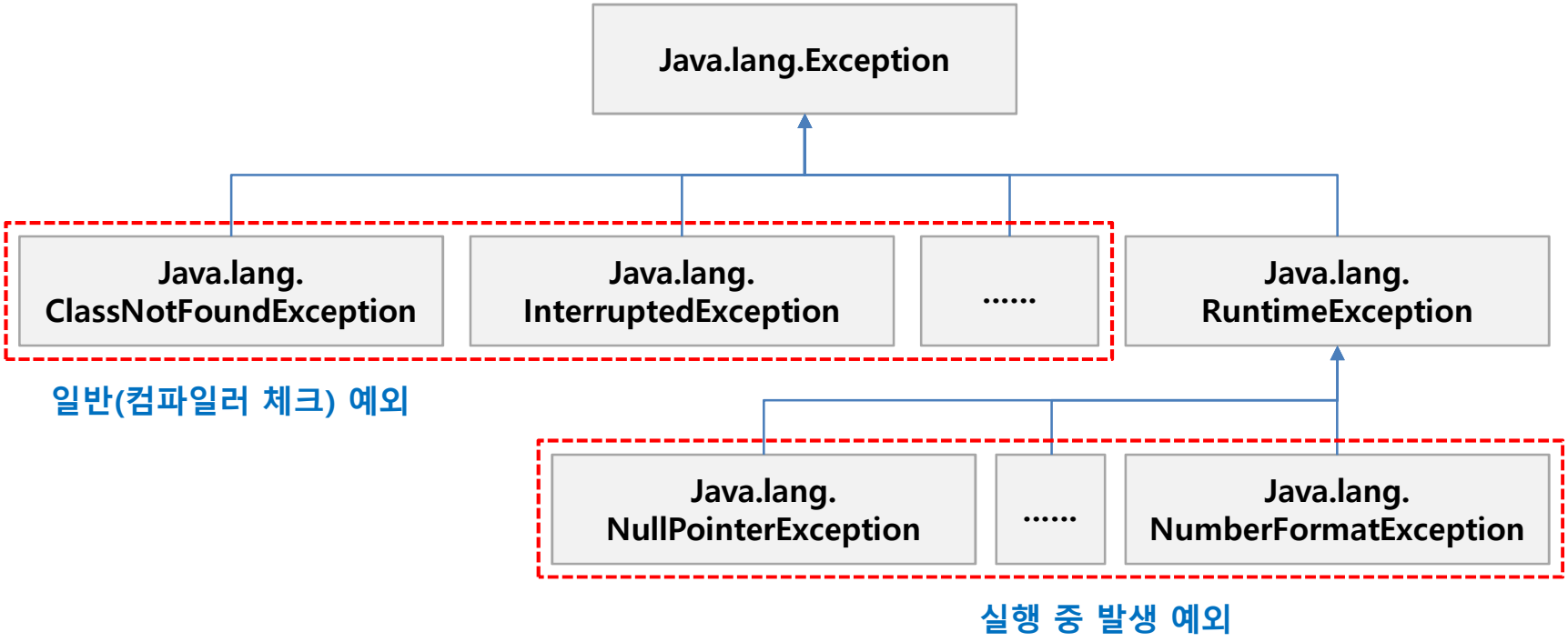
9장. 예외처리

9.1. 예외와 예외 클래스

- 컴파일러 체크 예외 : 예외 처리 코드가 없다면 컴파일이 되지 않는 예외
- 실행 중 발생 예외 : 예외 처리 코드를 생략하더라도 컴파일이 되는 예외

예외 클래스

❖ 실행 중 발생예외는 컴파일시 체크가 되지 않기 때문에 예외처리가 필요함.
❖ 예외 처리가 되지 않을 경우 운영 시스템이 Down 될 수 있으므로 주의 필요함



Exam_01

9.2. RuntimeException 종류

- Runtime 중 발생 가능 예외는 다양하나 아래 4건에 대해서 주로 발생함.
- 주로 발생 예외 : NullPointer, ArrayIndexOutOfBound, NumberFormat, ClassCast 관련 예외

RuntimeException 종류

NullPointerException

❖ 객체 참조가 없는 상태에서 메서드를 호출할 때 발생

```
Car car = null;
car.run( );                    // 예외발생
```

ArrayIndexOutOfBoundsException

❖ 배열에서 인덱스 범위를 초과하여 사용할 경우 발생

```
Int [ ] human= {10,20,30} ;
System.out.println (human[5]);

// human은 0~2의 배열만 있으므로 예외발생
```

NumberFormatException

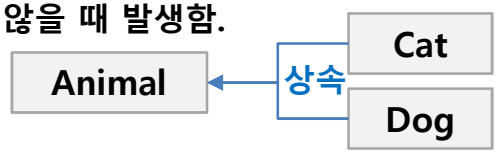
❖ 문자열을 숫자로 변환하는 경우 주로 발생함
❖ 숫자로 변환할 수 없는 문자가 포함되어 있을 때

```
String stringHuman = "a100";
Int intHuman = Integer.parseInt(stringHuman);
// string_human에는 문자 a가 포함되어 있어.
   숫자로 변환이 불가함. ➔ Exception 발생
```

ClassCastException

Exam_02

❖ 타입변환이 되지 않을 때 발생함.

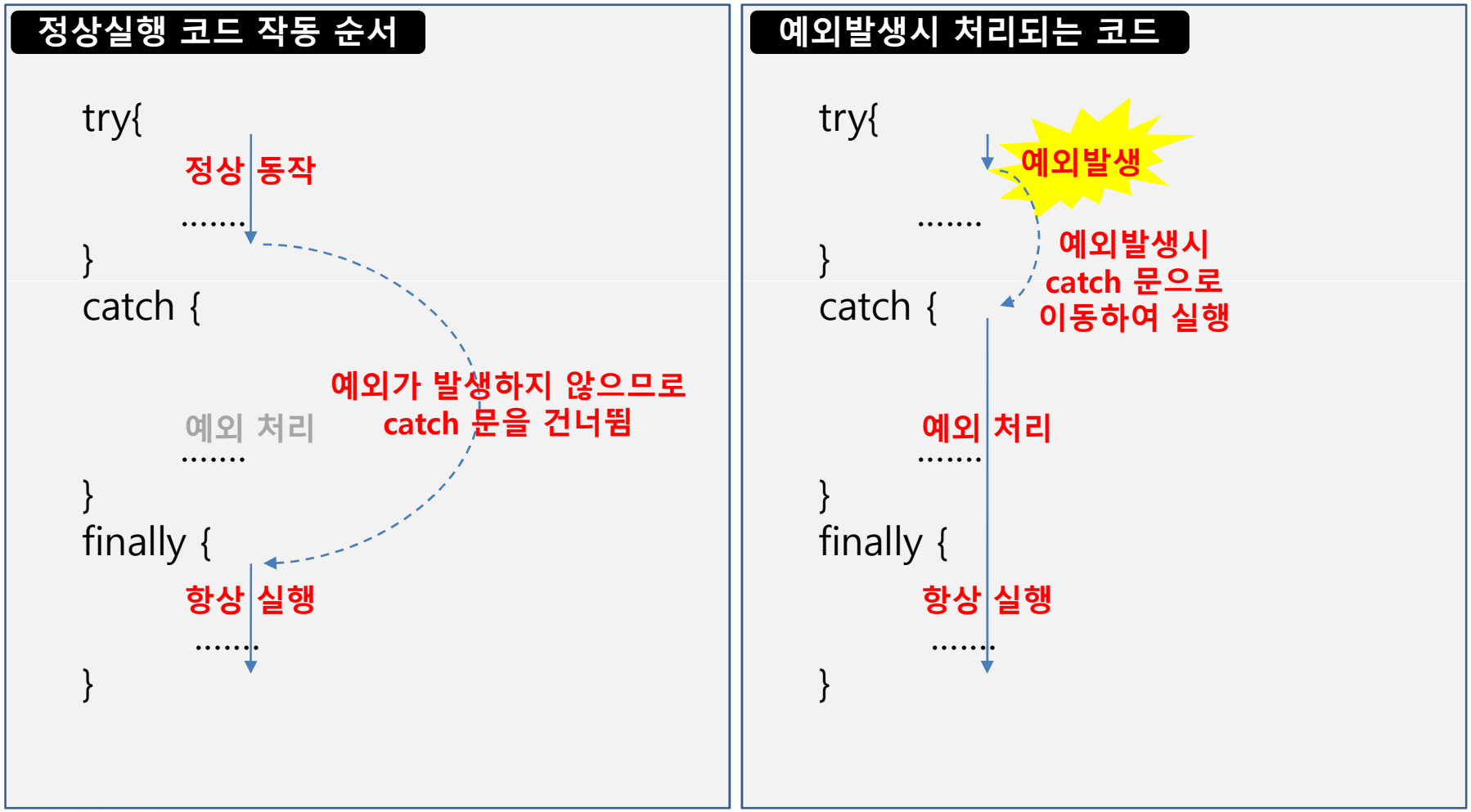



```
Animal animal = new Dog( );
Cat cat = (Cat) Animal;
// Animal은 Dog으로 변환가능
// 그러나 Animal을 Cat으로 변환시 Exception 발생
```

9.3. 예외처리 코드 (try - catch - finally)

- 예외가 발생하면 프로그램 종료를 막고, 정상 실행상태를 유지하는 코드.
- how : 예외가 발생하면, catch문의 예외처리 코드를 실행하여 시스템의 이상현상을 방지함

예외처리코드



9.4. 예외 종류에 따른 처리코드

- 다중 catch : 예외 종류에 따라 예외 처리코드를 구분하여 처리함
- Multi catch : 여러 개의 예외를 동시에 처리할 수 있도록 함.

예외종류에 따른 처리

다중 catch

```
try{
        .....

}
catch (NullPointerException ne) {
   NullPointerException 예외처리;
}
catch (NumberFormatException fe) {
   NumberFormatException 예외처리;
}
finally {
        예외 처리 후 항상 실행
}
```

NumberFormat 예외발생

NullPointer 예외발생

Multi catch

❖ 자바 7부터 지원되는 것으로 여러 개의 예외를 같은 코드로 처리할 수 있음

```
try{
        .....

}
catch(NullPointerException | NumberFormatException e) {
   2가지 예외 발생시 예외처리함;
}
```

NumberFormat 예외발생

NullPointer 예외발생

9.5. 예외 넘기기 및 예외 정보 얻기

- 예외 떠넘기기 : throws 문을 통해 예외 처리가 되어 있는 class로 떠넘길 수 있음.
- 예외 정보 얻기 : Exception 객체를 통한 예외의 정보를 얻을 수 있음

예외넘기기 및 예외정보얻기

예외 넘기기

Exam_06

❖ 수많은 예외처리를 try ~ catch문으로 처리하지 않고 메서드 선언부에서 호출한 곳으로 예외를 던질 수 있음

```
public class A {
  public static void main (String [ ] args) {
    Test test = new Test( ) ;
    try {
          test.test ("1", "r");
   } catch (NumberFormatException e) {
          System.out.println ("숫자가 아닙니다.");
}
```

A가 숫자형태가 아닐 경우 Exception 발생
➔ 호출한 클래스로 예외 throw 함

```
public class Test {
  public void Test (String a, String b)
                    throws NumberFormatException {
    System.out.println (Integer.parseInt(b));
  }
}
```

예외 정보 얻기

Exam_05

❖ getMessage : 예외의 정보를 얻음
❖ printStackTrace
: 예외발생코드를 추적하여 콘솔에 출력함

```
try{
        .....
}
catch(Exception e) {
   String message = e.getMessage( ); // 예외 정보 얻기
   e.printStackTrace( ) ; // 예외의 발생 경로 추적

}
```

예외발생