5. 1 예외 처리

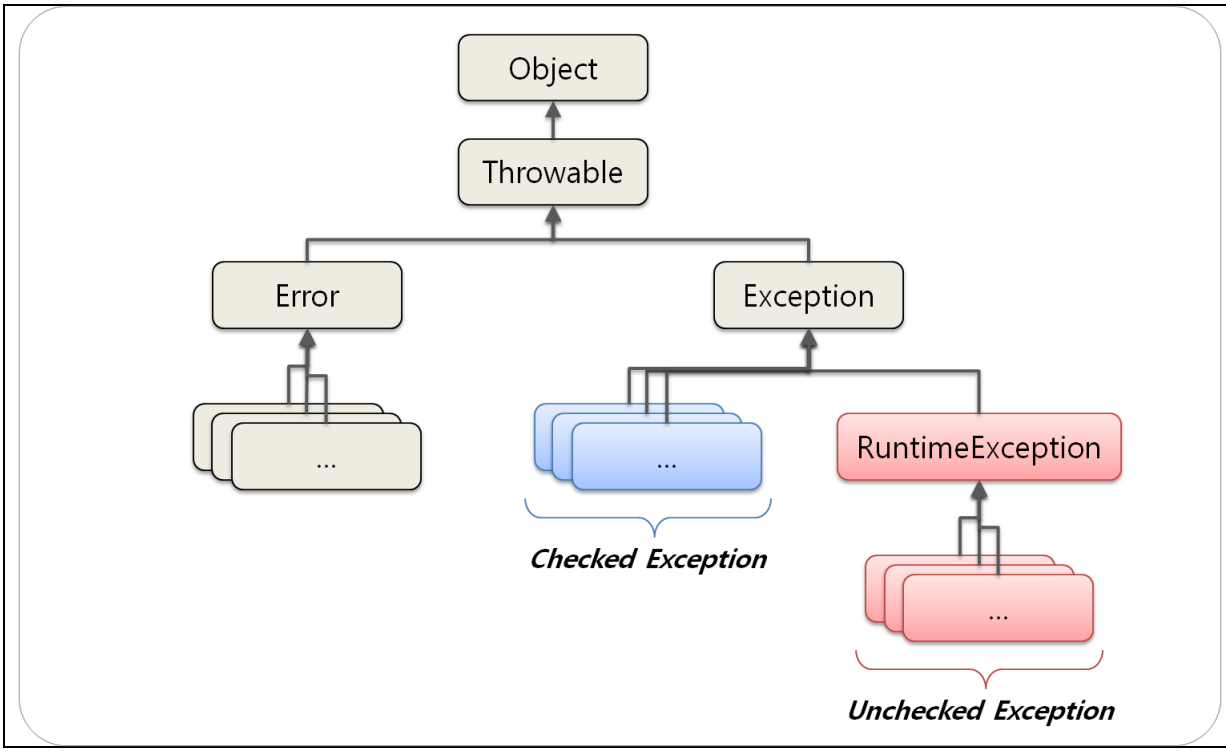
5.1.1 예외 던지기

throw문으로 예외 클래스 객체를 던진다.

```
public static int randInt(int low, int high){
        if(low > high){
                throw new IllegalArgumentException();
        }
        return low + (int)(Math.random() * (high - low+ 1));
    }
```

5.1.2 예외 계층

Throwable 객체를 상속받아서 구현한다.



5.1.3 검사예외 선언

메서드 헤더에 throws선언

```
public void write(Object obj, String filename) throws IOException, ReflectiveOperationException{
}
```

5.1.4 예외잡지

예외는 try catch 문으로 선언한다.

```
try{
        statment
    }catch(Exception1 e){
        handler1
    }catch(Exception2 e){
        handler2
    }

try{
        statment
    }catch(Exception1 e1 | Exception2 e2 | Exception3 e3){
        handler1
    }
```

5.1.6 finally 절

```
try{
        작업을 수행한다.
    }finally{
        정리 작업을 한다.
    }
```

5.2 단정

일반적으로 사용하는 방어적 프로그래밍방벅

Assert condition;

Assert condition : expression;

Assert x >= 0;

Assert x>= 0 : x;'

5.3 로깅

문제의 원인이나 과정을 출력문에 나타내는 기능

Logger logger = Logger.*getLogger*(App.**class**);

5.3.3 로깅 레벨

SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST 로 구성된다.

Looger.warning(message);

Looger.fine(message)

6.1 제네릭클래스

타입매개변수를 한 개 이상 받는 클래스

```
public class Entry<K, V>{
        private K key;
        private V value;

        public Entry(K key, V value){
                this.key = key;
                this.value = value;
        }

        public K getKey(){ return key; }
        public V getValue(){ return value; }
}
```

6.2 제네릭 메서드

타입 매개변수를 받는 메서드

```
        public static<T> void swap(T[] array, int i, intj)
```

6.3 타입 경계

특정 클래스를 확장하거나 특정 인터페이스를 타입 매개변수로 제한 할수있다.

```
public static <T extends AutoCloseable> void closeAll(ArrayList<T> elems)
        throws Exception{
                for(T elem : elems){
                        elem.close();
                }
        }
```

6.4 타입 가변성과 와일드카드

와일드카드란 매개변수와 반환 타입이 변하는 방식을 지정 (null은 지정불가능)

```
public static void Employeees(ArrayList<? extends Employee> staff){
        for(int i = 0; i < staff.size(); i++){
            Employee e = staff.get(i);
            System.out.println(e.getName());
        }
    }
```

6.4.3 타입 변수를 이용한 와일드카드

```
public boolean addAll(Collection<? extends E> c)
```

6.4.5 와일드카드 캡쳐

?를 타입인수로 사용 할수 있지만 타입으로는 사용할수 없다.

```
public static void swap(ArrayList<?> elements, int i, int j){
        ? temp = elements.get(i) // 작동하지 않는다.
    }
```

6.5 자바 가상 머신에서 보는 제네릭

제네릭 타입을 정의하면 해당 타입은 로(raw) 타입 으로 컴파일된다.

```
public class Entry{
    private Object key;
    private Object value;

    public Entry(Object key, Object value){
        this.key = key;
        this.value = value;
    }

    public Object getKey(){ return key; }
    public Object getValue(){ return value; }
}
```

6.6제네릭제약

1. 기본타입 인수를 사용할수없다.

   - Int, char 등 기본타입 선언 불가

2. 실행 시간에는 모든 타입이 Raw로 형태다.

3. 타입 변수를 인스턴스화할 수 없다.

   - T(..) 또는 new T[...] 사용불가

4. 매개변수화된 타입의 배열을 생성할 수 없다.

5. 정적 컨텍스트에서는 클래스 타입 변수가 유효하지 않다.

   - 정적 변수나 메서드에서는 K와 V타입 변수를 사용할수 없다.

6. 제네릭 클래스의 객체는 에외로 던지거나 잡을수 없다.

```
public class Problem<T> extends Exception
```