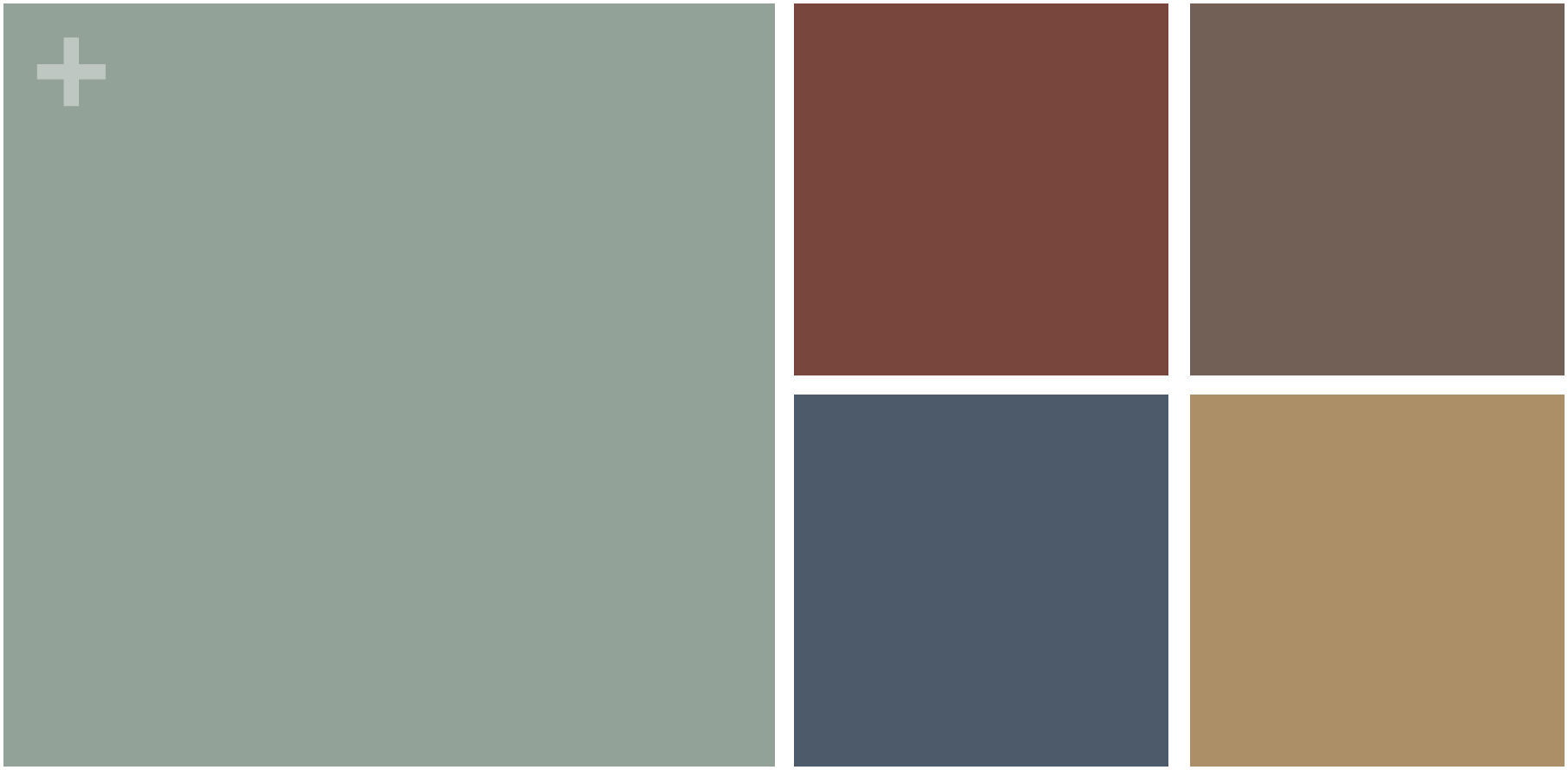


Chapter 21. 예외 처리
Chapter 22. 제네릭과 컬렉션

1. 예외 처리

- 다음 장에 나오는 예시코드에서 예외가 발생할 수 있는 위치에 **try/catch**문을 작성하여라.
- 예외 처리는 밑에 나열되어 있는 예외 만을 처리한다
  - **ArithmeticException**(어떤 수를 0으로 나눈 경우)
  - **NegativeArraySizeException**(배열의 크기가 음수 값인 경우)
  - **ArrayIndexOutOfBoundsException**(배열을 참조하는 인덱스가 잘못된 경우)

```
public class ExceptionTest {
  public static void main(String[] args) {
    int[] list;
    int sum = 0, count;

    Scanner sc = new Scanner(System.in);
    System.out.print("정수의 개수:");
    count = sc.nextInt();
    list = new int[count];

    for(int i = 0; i < count; i++) {
      System.out.print("정수를 입력하시오:");
      list[i] = sc.nextInt();
    }

    for(int i = 0; i < count; i++) {
      sum += list[i];
    }
    System.out.println("평균은 " + sum/count);
  }
}
```

예외 처리방법 예시

- 다음과 같이 **try/catch**문을 작성하여 예외처리를 해준다.
- **catch**문 안에 있는 "**ArithmeticException**"은 처리하고자 하는 예외 상황에 따라 교체하여 작성한다.

```
try{
    //예외가 발생할 수 있는 코드
    result = 10/a;
}
catch(ArithmeticException e){
    //예외 발생시 동작하는 코드
    //예외 처리
    System.out.println("0으로 나눌수 없습니다.");
}
```

ArithmeticException
(어떤 수를 0으로 나눈 경우)

- 다음 코드에서 나눗셈연산에 의해 예외가 발생 될 수 있으므로 **try/catch**문을 사용하여 예외처리를 한다.

```
        for(int i = 0; i < count; i++) {
            sum += list[i];
        }
        System.out.println("평균은 " + sum/count);
    }
```

NegativeArraySizeException (배열의 크기가 음수 값인 경우)

- 다음 코드에서 배열을 생성하면서 예외가 발생 될 수 있으므로 **try/catch**문을 사용하여 예외처리를 한다.

```
Scanner sc = new Scanner(System.in);
System.out.print("정수의 개수: ");
count = sc.nextInt();
list = new int[count];
```

ArrayIndexOutOfBoundsException (배열을 참조하는 인덱스가 잘못된 경우)

- 다음 코드에서 배열을 참조하면서 예외가 발생 될 수 있으므로 **try/catch**문을 사용하여 예외처리를 한다.

```
for(int i = 0; i < count; i++) {
    System.out.print("정수를 입력하시오: ");
    list[i] = sc.nextInt();
}

for(int i = 0; i < count; i++) {
    sum += list[i];
}
```

2. 제네릭과 컬렉션

- 키(K)과 값(V)을 갖는 Pair<K,V> 제네릭 클래스를 정의하라.
- 키를 Calendar로 하고, 값을 String으로 하는 Event 클래스를 정의하라
- Event 객체를 키 값 (Calendar)으로 값을 비교할 수 있도록 Comparable 인터페이스를 구현하라. (Hint: compareTo() 메소드 구현)
- Event 객체를 java.util 패키지에서 제공하는 Vector에 저장하라.

키(K)과 값(V)을 갖는 Pair<K,V> 제네릭 클래스를 정의

- Pair<K,V> 제네릭 클래스는 타입 매개변수인 K와 V를 필드로 포함한다.
- 각 필드 값들을 설정 및 제공을 받기 위한 Set함수와 Get함수를 구현 한다

키(K)과 값(V)을 갖는 Pair<K,V> 제네릭 클래스를 정의

- 제네릭 클래스 정의하는 방법
  - T를 갖는 Box<T> 제네릭 클래스 예시

```
public class Box<T> {
    private T data;

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

}
```

키 입력을 Calendar 로 하고, 값을 String으로 하는 Event 클래스를 정의하라

- Pair<K,V> 제네릭 클래스를 사용하여 Event클래스를 정의한다.
  - Event 클래스의 필드로 Pair<K,V> 를 포함하여 해당 객체의 이름은 p라고 한다.
  - 생성자에 year(년), month(월), date(일), e(이벤트내용)을 입력 받아 p의 값을 세팅한다.

키 입력을 Calendar 로 하고, 값을 String으로 하는 Event 클래스를 정의하라

- 제네릭 클래스 사용 방법
  - T를 String으로 하는 StringBox 클래스 정의 예시

```
public class StringBox {
    Box<String> box;

}
```

```
public class Box<T> {
    private T data;

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

}
```

Event 객체를 키 값 (Calendar)으로 값을 비교할 수 있도록 Comparable 인터페이스를 구현하라.

- Comparable 인터페이스를 사용방법
  - implements Comparable 추가

```
public class Event implements Comparable{
```

- compareTo 함수 구현
  - Calendar자체에서 compareTo가 구현되어 있으므로 이를 활용하여 구현한다.

```
@Override
public int compareTo(Object o) {
    // TODO Auto-generated method stub
    Event other = (Event)o;

    return this.p.getKey().compareTo(other.p.getKey());
}
```

Event 객체를 키 값 (Calendar)으로 값을 비교할 수 있도록 Comparable 인터페이스를 구현하라.

- Comparable 인터페이스 사용 방법

```
public class StringBox implements Comparable{
    Box<String> box;
    int num;

    @Override
    public int compareTo(Object o) {
        // TODO Auto-generated method stub
        StringBox temp = (StringBox) o;

        if(this.num < temp.num) return -1;
        else if(this.num > temp.num) return 1;
        return 0;
    }

}
```

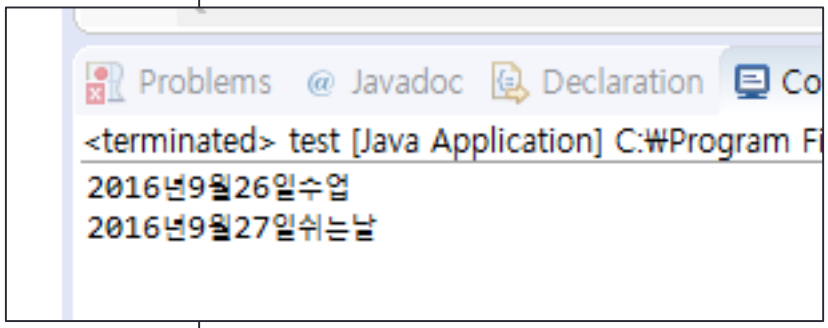
Event 객체를 java.util 패키지에서 제공하는 Vector에 저장하라.

- Vector 사용 방법

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    StringBox b;
    Vector<StringBox> v = new Vector<>();

    b = new StringBox();

    v.add(b);
}
```

실습 테스트

```java
import java.util.Calendar;
import java.util.Vector;

public class test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Event e;
        Vector<Event> v = new Vector<>();

        e = new Event(2016,9,26,"수업");
        v.add(e);
        e = new Event(2016,9,27,"쉬는날");
        v.add(e);

        for(int i=0; i<v.size(); i++)
        {
            System.out.println(v.get(i).p.getKey().get(Calendar.YEAR) + "년" +
                    v.get(i).p.getKey().get(Calendar.MONTH) + "월" +
                    v.get(i).p.getKey().get(Calendar.DATE) + "일" +
                    v.get(i).p.getValue());
        }
    }

}
```



[Take Home 과제] Calendar+

- 일정을 관리하기 위한 MyCalendar클래스를 만들어라
- 일정은 List를 이용하여 관리한다.
  - List는 직접 구현하지 않고 Java에서 제공되는 컬렉션 중 Linked List를 사용한다.
  - List의 원소(타입매개변수)로는 <실습2>에서 만든 Event 클래스를 사용한다.
- List에 일정을 add/delete/search할 수 있는 메소드를 구현하라.
- EventIndexPrint, EventPrint, EventAllPrint 메소드를 구현하라.
- EventSort를 구현하라.

[Take Home 과제] Calendar+

- **Add** 메소드
  - 인자 값 : 년, 월, 일, 이벤트내용
  - **List**에 이벤트 추가
- **Delete** 메소드
  - 인자 값 : 년, 월, 일
  - 해당 일자의 이벤트를 제거, 해당 일자에 이벤트 없으면 그에 대한 메시지 출력
- **Search** 메소드
  - 인자 값 : 년, 월, 일
  - 해당 일자의 이벤트를 찾아 인덱스 값을 반환

[Take Home 과제] Calendar+

- **EventIndexPrint**
  - 인자 값 : 인덱스 번호
  - 전달 받은 인덱스 번호에 위치한 이벤트정보를 출력
- **EventPrint**
  - 인자 값 : 년, 월, 일
  - 해당 일자의 이벤트정보를 출력, 해당 일자에 이벤트가 없는 경우 그에 대한 메시지 출력
- **EventAllPrint**
  - 인자 값 : 없음
  - 등록되어 있는 모든 이벤트정보를 출력

[Take Home 과제] Calendar+

- EventSort
  - 인자 값 : 없음
  - Java에서 제공하는 Collections를 이용하여 정렬하라(PowerJava566p~569p참조)

[Take Home 과제] Calendar+

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    MyCalendar mycalendar = new MyCalendar();

    mycalendar.add(2020,11,11, "빼빼로데이");
    mycalendar.add(2000,5,5, "어린이날");
    mycalendar.add(2020,1,1, "새해");
    mycalendar.add(2100,3,1, "3.1절");

    mycalendar.EventPrint(2020, 1, 1);

    mycalendar.del(2020, 1, 1);
    mycalendar.del(2020, 1, 2);

    mycalendar.EventPrint(2020, 1, 1);

    System.out.println("<<정렬전 모든 일정>>");
    mycalendar.EventAllPrint();

    mycalendar.sort();
    System.out.println("<<정렬후 모든 일정>>");
    mycalendar.EventAllPrint();

}
```

```
<terminated> test (1) [Java Applic
2020년1월1일 : 새해
삭제하고자 하는 일정이 없습니다.
일정을 찾지 못했습니다.
<<정렬전 모든 일정>>
2020년11월11일 : 빼빼로데이
2000년5월5일 : 어린이날
2100년3월1일 : 3.1절
<<정렬후 모든 일정>>
2000년5월5일 : 어린이날
2020년11월11일 : 빼빼로데이
2100년3월1일 : 3.1절
```

과제 제출 방법 및 기한

- 주석을 사용하여 코드 설명
- 보고서 제출 없음, 프로젝트 만을 압축하여 제출
- 프로젝트 및 압축파일 이름 형식 : “Lab03”
- 다음주 화요일(10/4) 23:59까지