실습 진행 순서

1. 과제 4 설명
   1) GenericStack : 20분
   2) ParaStack : 20분
   3) ParaStackNoLimit : 10분

2. GenericStack
   - 특정 type의 data만 취급하는 type-specific 이 아닌,
   - 모든 data를 취급할 수 있는 stack
   - 저장 공간은 Object[] 로 구성
   - isEmpty(), push(), pop() 만 제공
   - 한정된 개수의 data만 수용한다고 가정
   - Test class 작성해서 test

실습 진행 순서 (2)

3. ParaStack
   - stack instance를 생성할 때, 특정 type 의 data 만 취급하도록 지정
   - 즉, 프로그래밍은 generic 하게 하고, 여러 type에 type-specific 하게 사용
   - GenericStack 과 같은 구조
   - type을 parameter로 지정할 수 있도록 (다음 페이지 예 참조)
   - ArrayList 등이 실제 이 방식을 사용

4. ParaStackNoLimit
   - ParaStack 을 extend 해서
   - data 수용 개수에 제한이 없도록
   - array 크기가 모두 소진되면, array 크기를 2배로
   - 구체 방법은 참조 코드에 (마지막 페이지)

Parameterized Generic Programming 예

- Type-safe generic programming
- Parameterized Type 사용

```
public class ParaStack<E> {

  private ArrayList<E> list = new ArrayList<E>();

  public void push(E o) {
    list.add(o);
  }
  public E pop() {
    E o = list.get(getSize() - 1);
    list.remove(getSize() - 1);
    return o;
  }

  public boolean isEmpty() {
    return list.isEmpty();
  }
  public int getSize() { return list.size(); }
}
```

<사용 예>

```
…
ParaStack<String> s =
    new ParaStack<String>();

s.push(“first”);

s.push(2); // error

…
```

Stack capacity 늘리는 방법

```
/** Push a new integer into the top of the stack */

 public void push(int value) {
   if (size >= elements.length) {
     int[] temp = new int[elements.length * 2];
     System.arraycopy(elements, 0, temp, 0, elements.length);
     elements = temp;
   }

   elements[size++] = value;
 }
```