1. Java와 C#에서 reference(참조)에 대한 산술 연산을 허용하지 않는다. 만일 허용하면 어떤 일이 발생하겠는가?

# 다음 코드에서 문제점을 찾아보시오.

```
public class Stack {
  private Object[] elements;
  private int size = 0;
  private static final int DEFAULT_INITIAL_CAPACITY = 16;
  public Stack() {
     elements = new Object[DEFAULT_INITIAL_CAPACITY];
  }
  public void push(Object e) {
    ensureCapacity();
    elements[size++] = e;
  }
  public Object pop() {
    if (size == 0)
       throw new EmptyStackException();
    return elements[--size];
  }
```

```
private void ensureCapacity() {
   if (elements.length == size)
   elements =
   Arrays.copyOf(elements, 2 * size
   + 1);
   }
} // doubly grow
```

# *"Memory Leak"?*

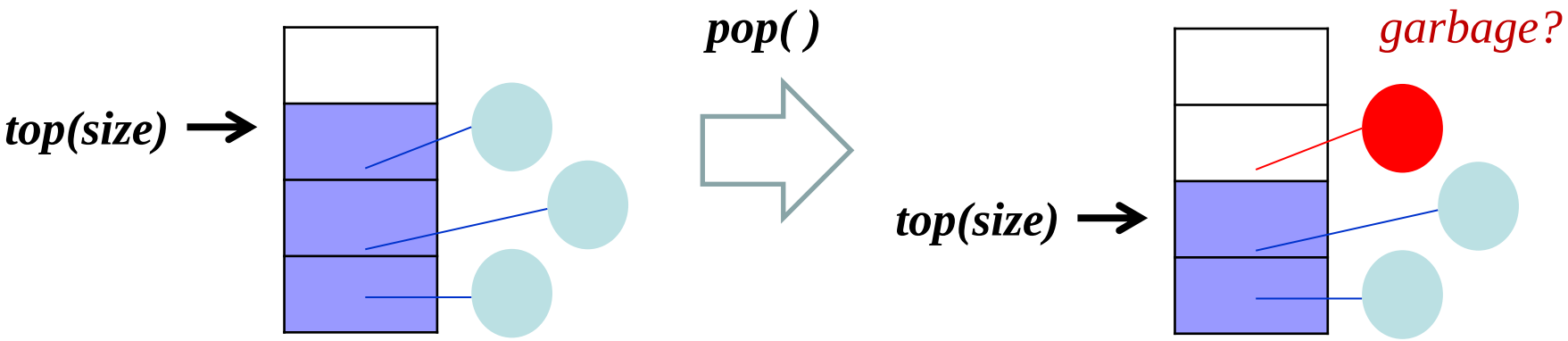
```
public class Stack {
  private Object[] elements;
  private int size = 0;
  private static final int DEFAULT_INITIAL_CAPACITY = 16;
  public Stack() {
     elements = new Object[DEFAULT_INITIAL_CAPACITY];
  }
  public void push(Object e) {
    ensureCapacity();
    elements[size++] = e;
  }
  public Object pop() {
    if (size == 0)
       throw new EmptyStackException();
    return elements[--size];
  }
```

```
private void ensureCapacity() {
   if (elements.length == size)
   elements =
   Arrays.copyOf(elements, 2 * size
   + 1);
   }
} // doubly grow
```

# Object 의 Stack 문제

```
public Object pop() {
   if (size == 0)
      throw new EmptyStackException();
   return elements[--size];
}
```

*garbage collector 가*
*garbage 라는 것을 알수 있을까?*



# 해결: Nulling-Out 방법

```
public Object pop() {
   if (size == 0)
      throw new EmptyStackException();
   Object result =  elements[--size];
   elements[size] = null; // 명시적으로 null 을 지정하기
   return result;
}
```