실습

Object-Oriented Thought Process

Chapter 6 & 7

Designing with Objects

Mastering Inheritance and Composition

Contents

1. Term Project에 대한 Class Diagram 작성
   - UML Tool 이용
2. 대표적 시험 항목에 대하여 해당 클래스들의 Stub 프로그램 작성
3. 대표적 시험 항목에 대한 Sequence Diagram 작성

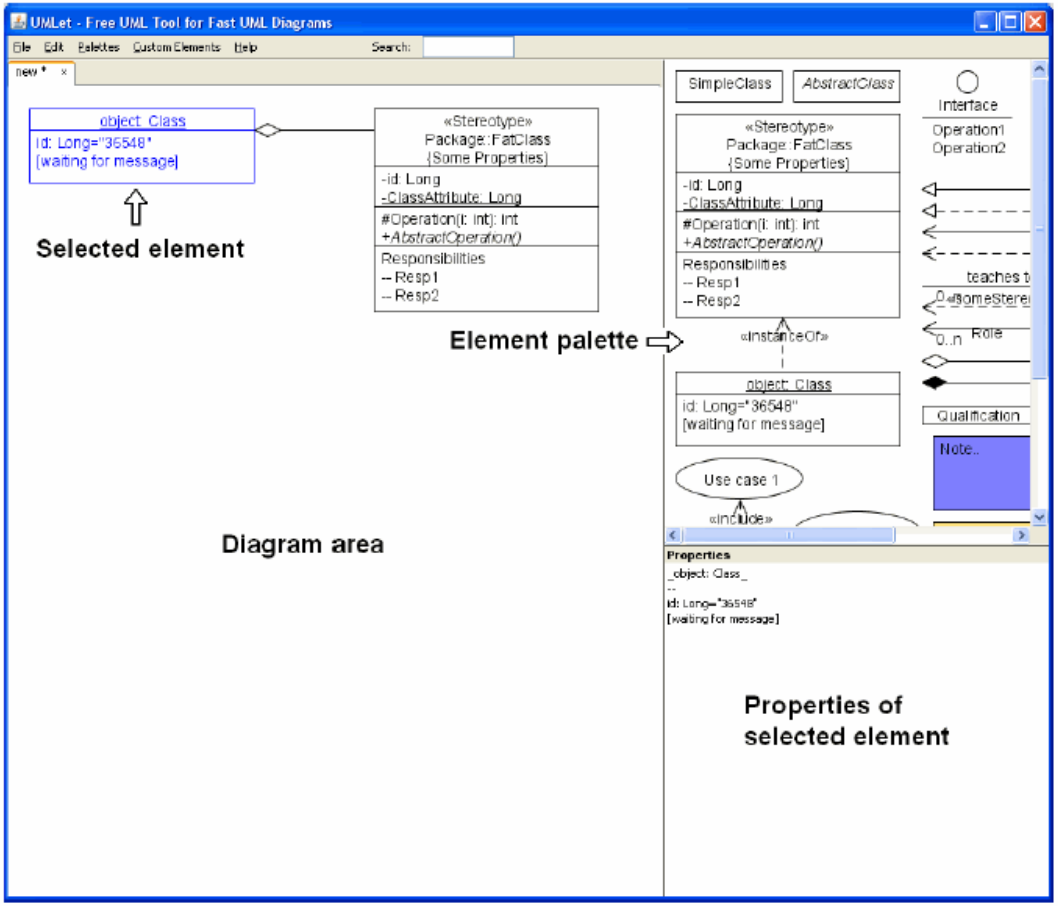
과제 제안서**[11/2**까지**]** - 작성요령

1. 환경 및 배경
   - 과제의 필요성/파급효과, 관련 기술 동향, 혁신성/독창성
2. 목표 및 내용
   - 최종 목표 (과제를 통해 개발할 내용/범위)
   - 구체적인 평가 방법 (평가 절차 및 결과)
   - 과제 내용 및 결과물 (목록 및 예상 결과물)
     - SW 설계 결과물
3. 수행 체계 및 일정 계획
   - 업무 분장, 마일스톤별 추진 일정

UML Tool: UMLet

- UMLet
  - http://www.umlet.com/
  - UMLet text-based approach is mainly aimed at fast UML sketching.

UMLet: UML Tools

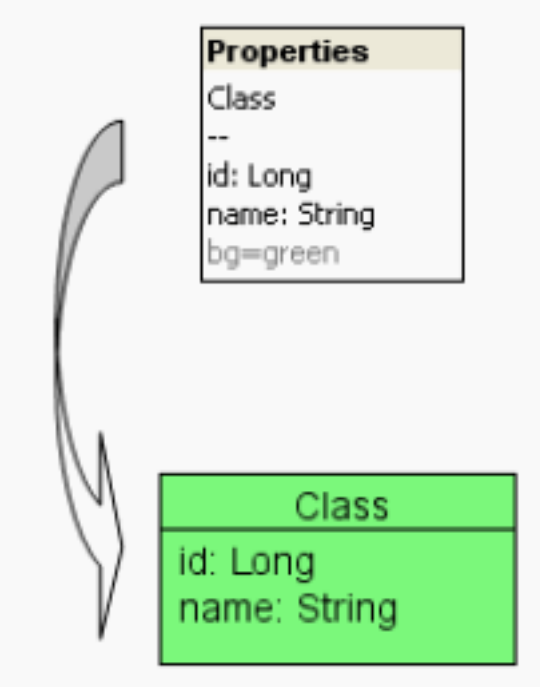


provides three panels: the diagram, the palette, and the property panel.

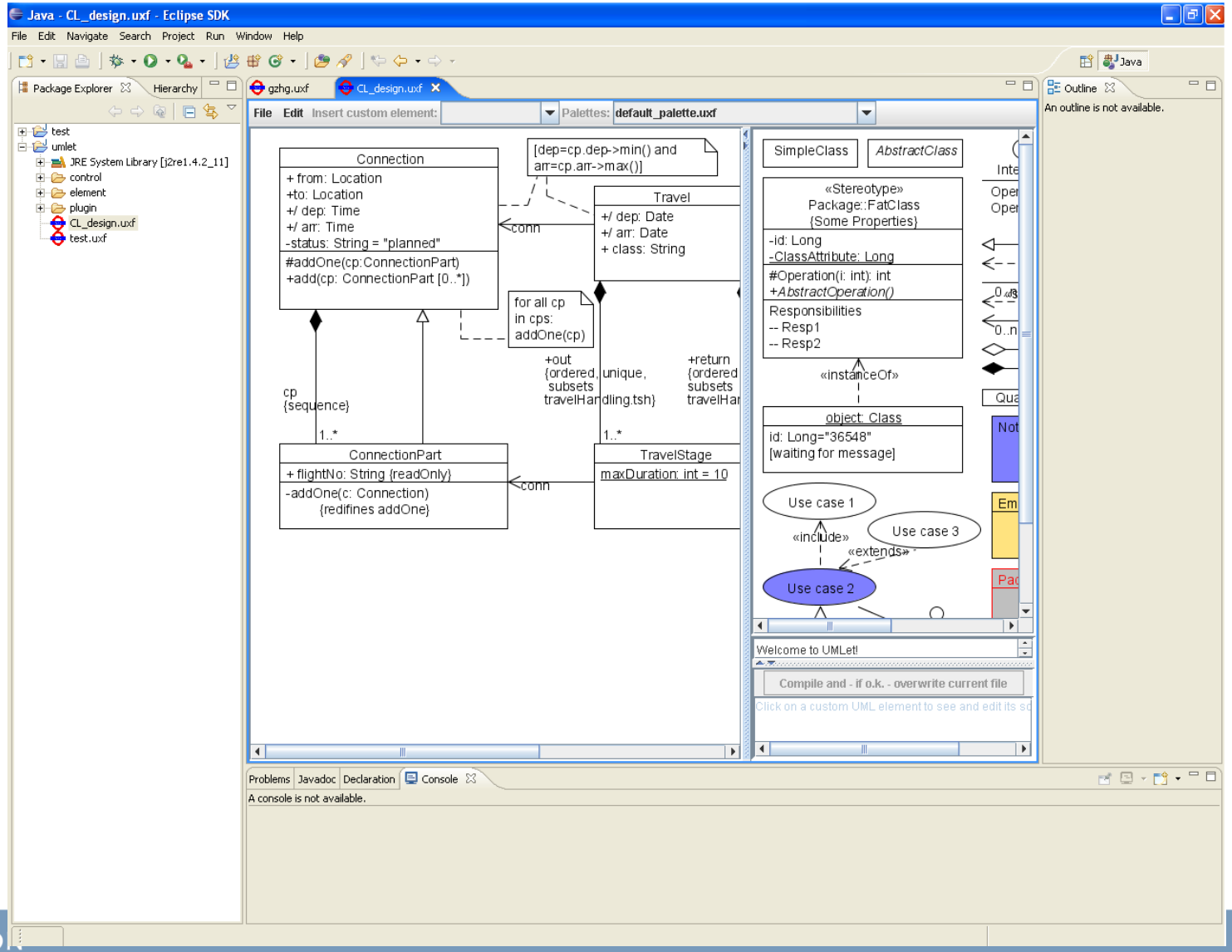
- The diagram panel displays the diagram and lets the user modify the UML elements' location;
- the palette panel lists the available elements;
- and the property panel lets users view and modify element properties.

- illustrates how the property text of a UML class element is interpreted by that class element to draw itself:

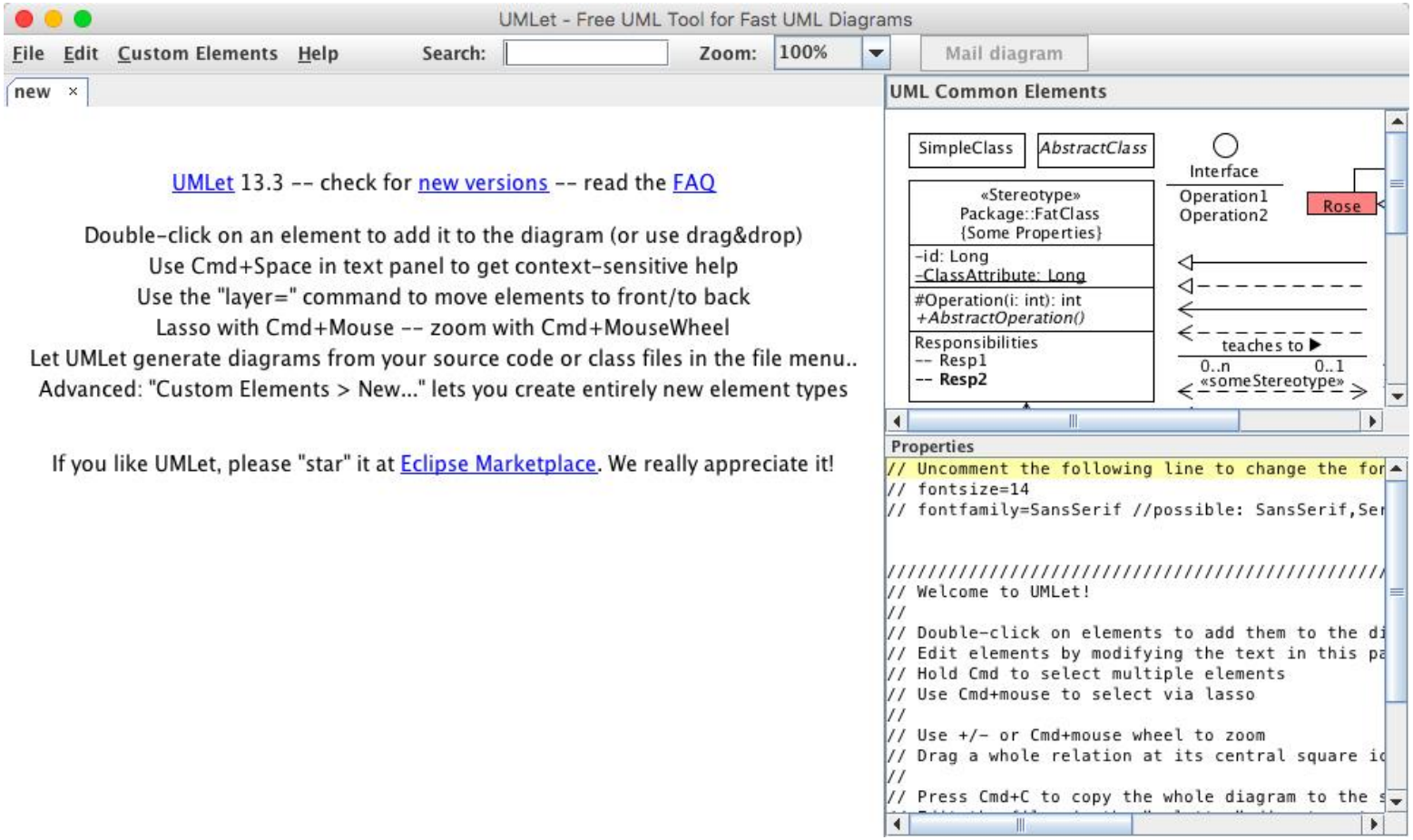
  - Class name and printed on top of the class, aligned centrally.
  - Subsequent lines are treated as method or attribute names, which are printed left-aligned.
  - The line "––" (a double dash) is interpreted as a horizontal line separating class name, attribute names, and method names;
  - the line "bg=green" sets the element's background color to green.



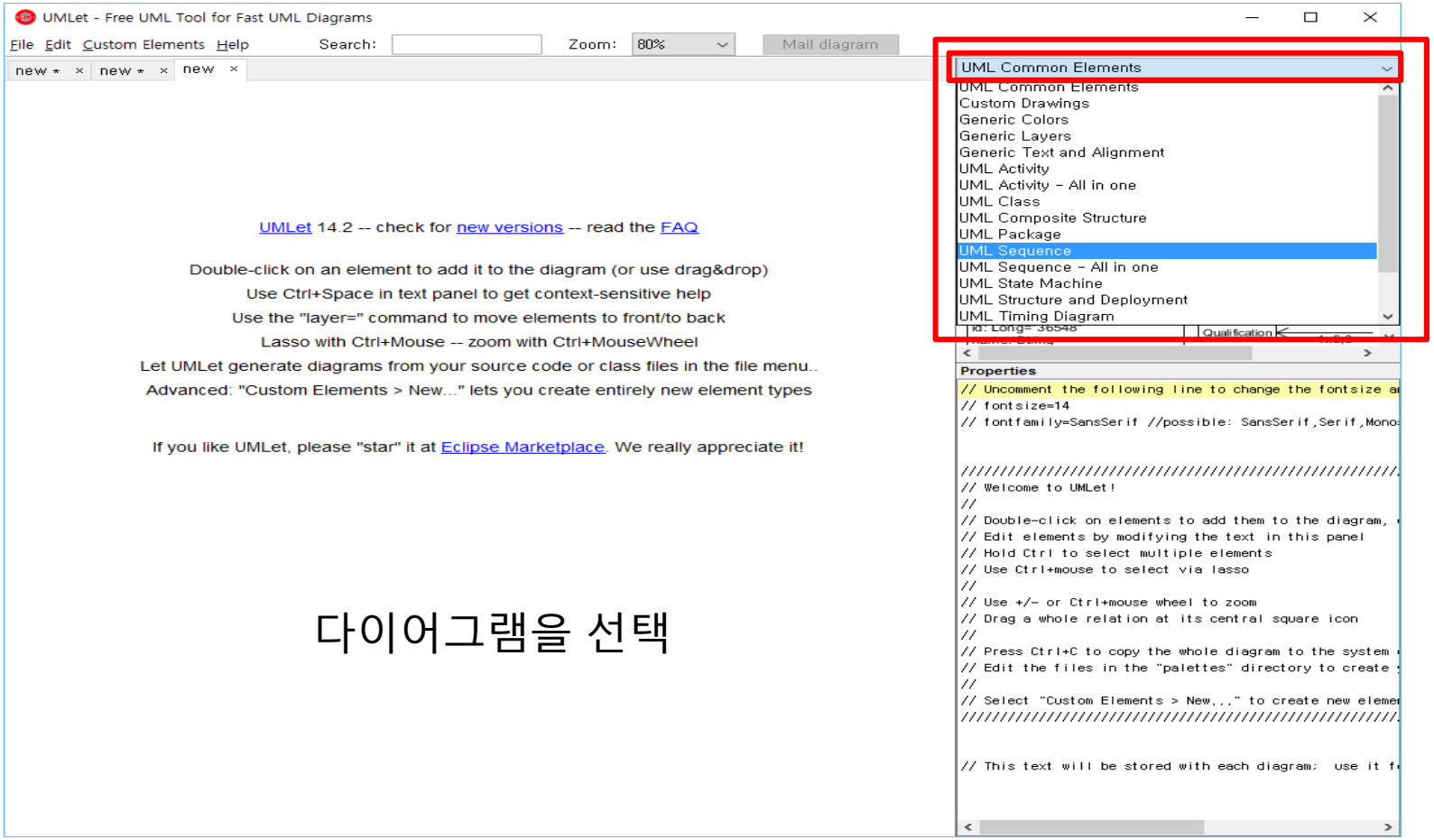
UMLet Plugin for Eclipse



Stand-alone UMLet

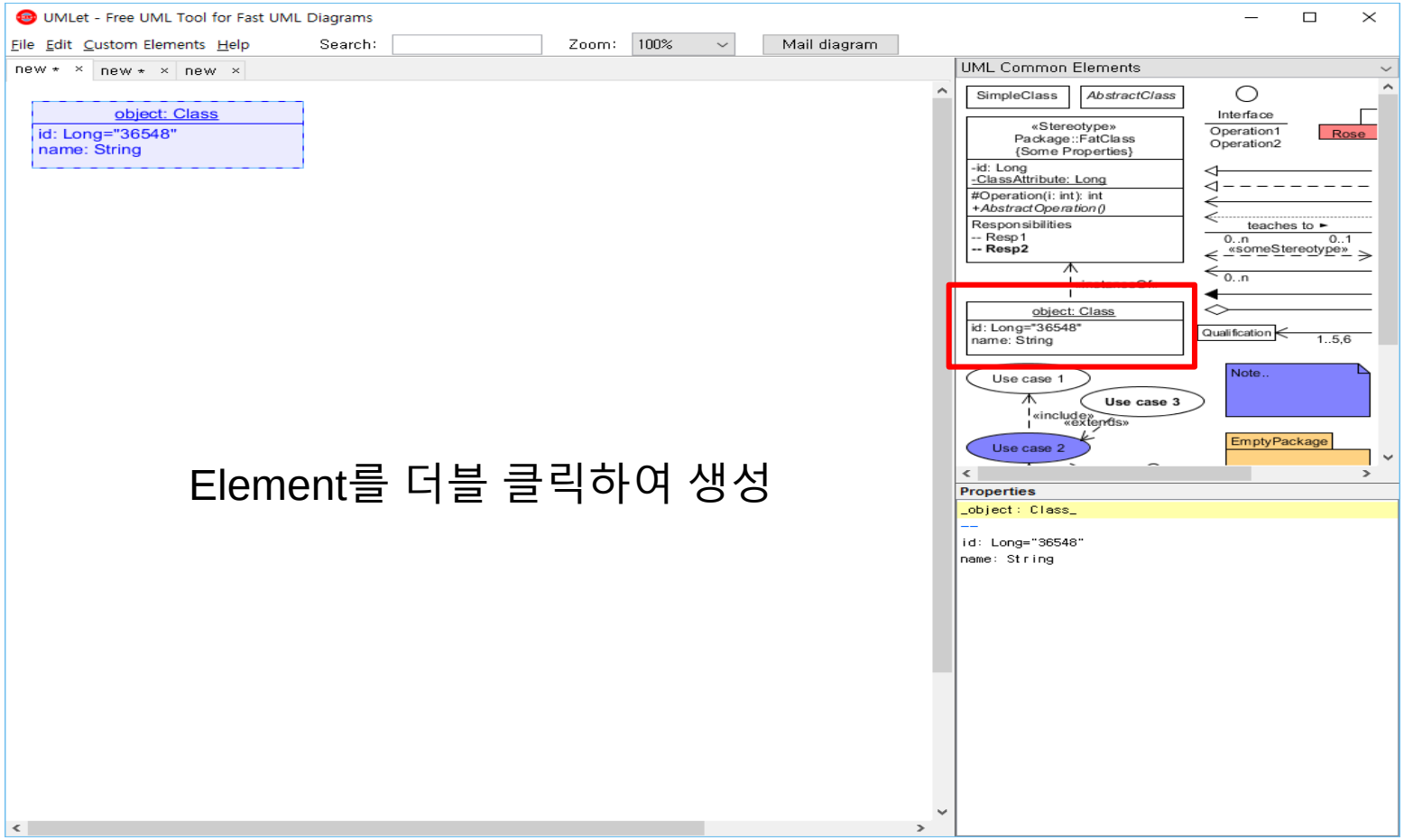


Class Diagram



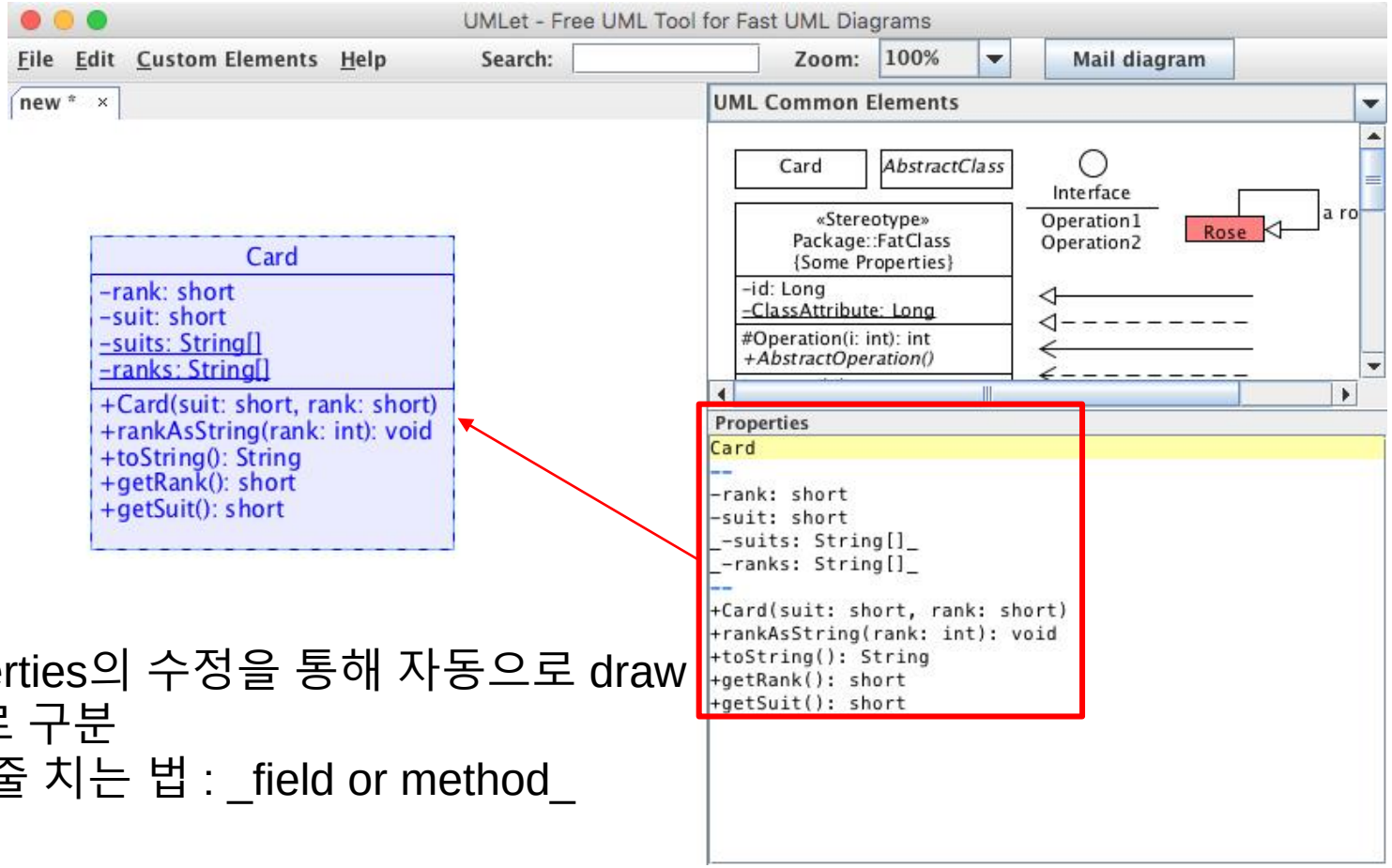
다이어그램을 선택

Class Diagram



Element를 더블 클릭하여 생성

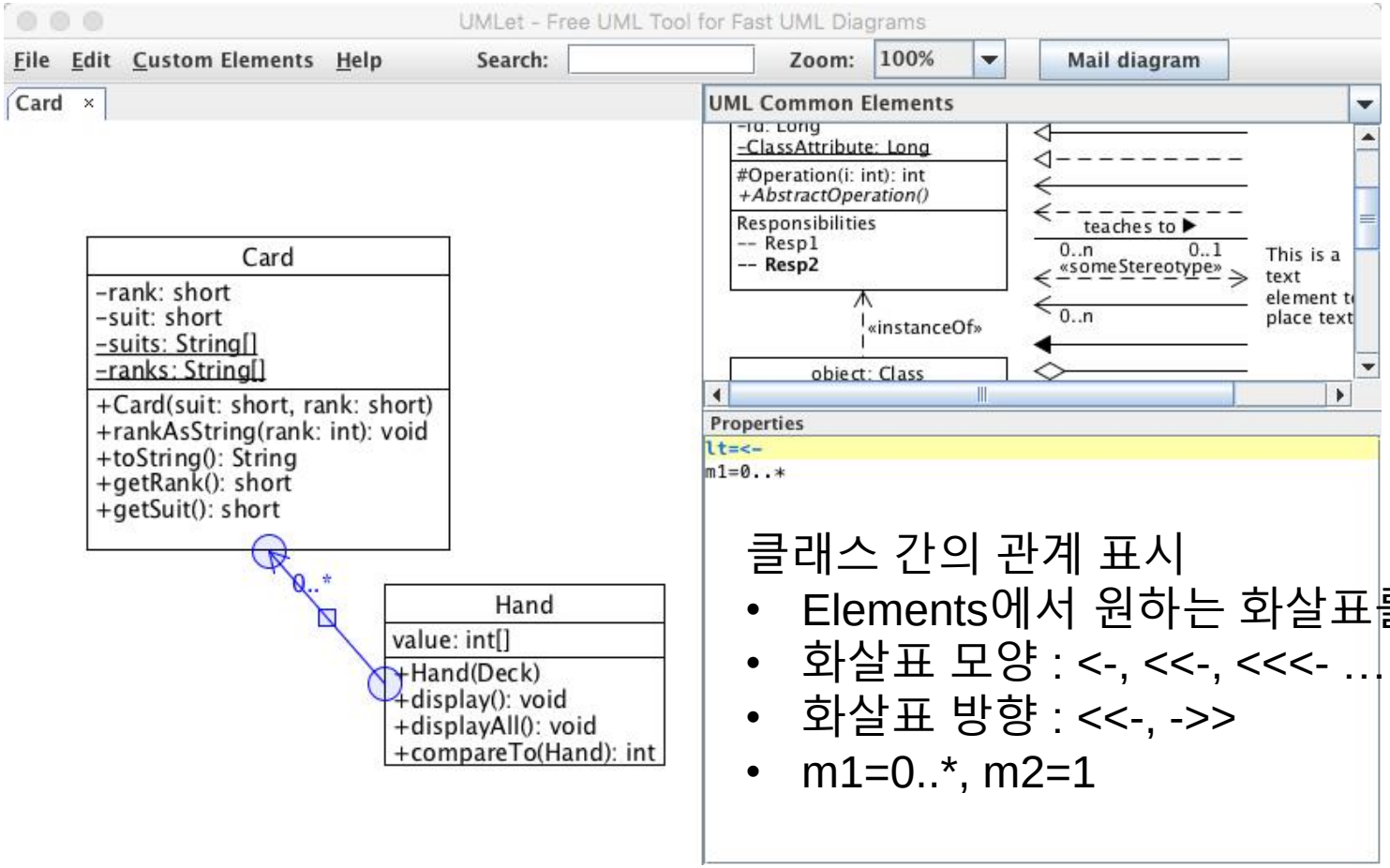
Class Diagram



Properties의 수정을 통해 자동으로 draw

- --로 구분
- 밑줄 치는 법 : _field or method_

Class Diagram



클래스 간의 관계 표시

- Elements에서 원하는 화살표를 더블 클릭
- 화살표 모양 : <-, <<-, <<<- …
- 화살표 방향 : <<-, ->>
- m1=0..*, m2=1

[ref] Stub Method

- [ref. Wiki] A method stub or simply stub in software development is a piece of code used to stand in for some other programming functionality.
- A stub may simulate the behavior of existing code (such as a procedure on a remote machine) or be a temporary substitute for yet-to-be-developed code.

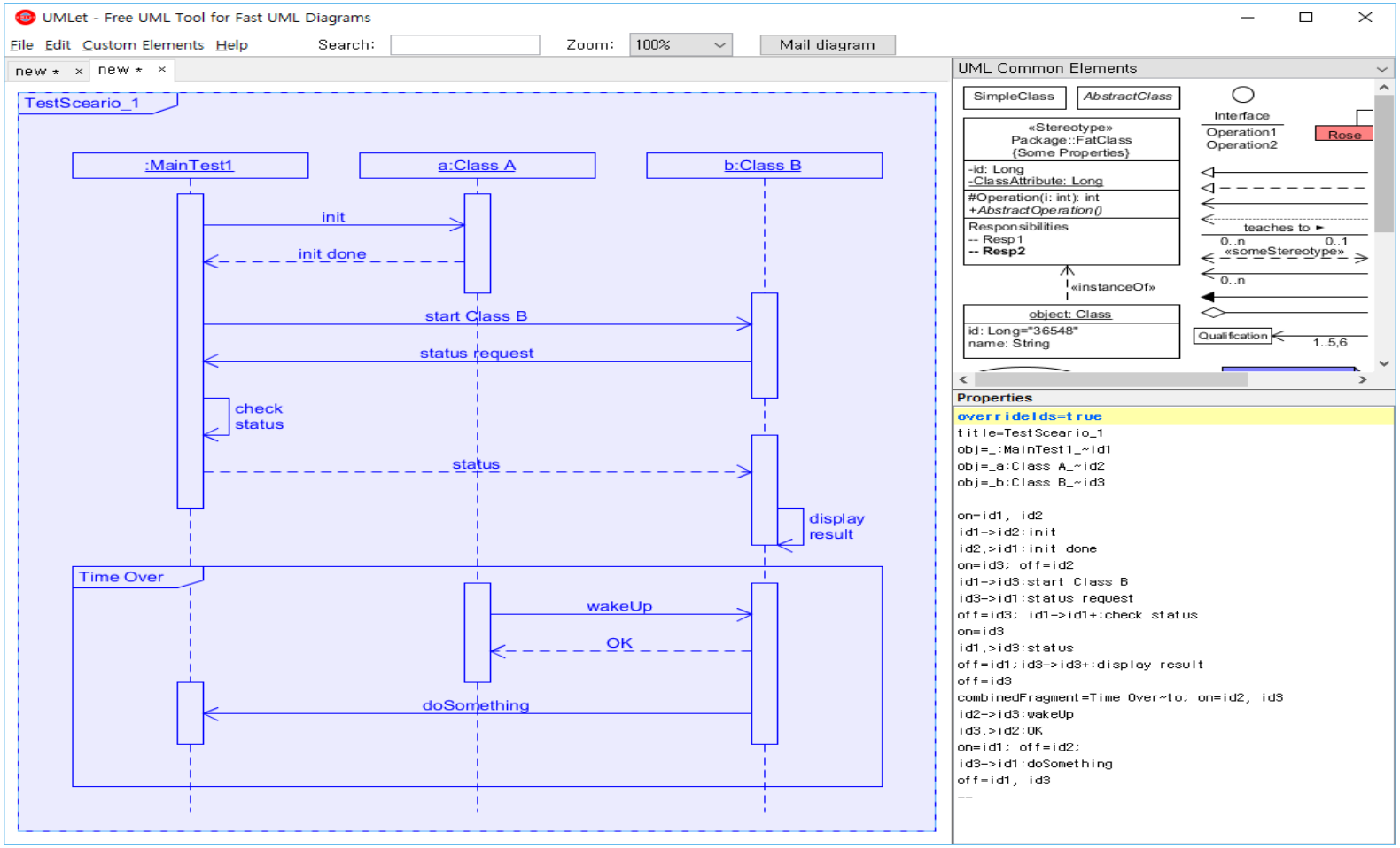
```
BEGIN
   Temperature = ThermometerRead(Outside)
   IF Temperature > 40 THEN
      PRINT "It's HOT!"
   END IF
END
```

```
BEGIN ThermometerRead(Source insideOrOutside)
   RETURN 28
END ThermometerRead
```

Testing the Interface

```
public class DataBaseReader {
  private String db[] = {
            "Record1",
            "Record2",
            "Record3",
            "Record4",
            "Record5"};
  private boolean DBOpen = false;
  private int pos;

  public void open(String Name){
    DBOpen = true;
  }

  public void close(){
    DBOpen = false;
  }

  public void goToFirst(){
    pos = 0;
  }

  public void goToLast(){
    pos = 4;
  }

  public int howManyRecords(){
    int numOfRecords = 5;
    return numOfRecords;
  }

  public String getRecord(int key){
   /* DB Specific Implementation */
    return db[key];
  }

  public String getNextRecord(){
   /* DB Specific Implementation */
    return db[pos++];
  }
}
```

Sequence Diagram



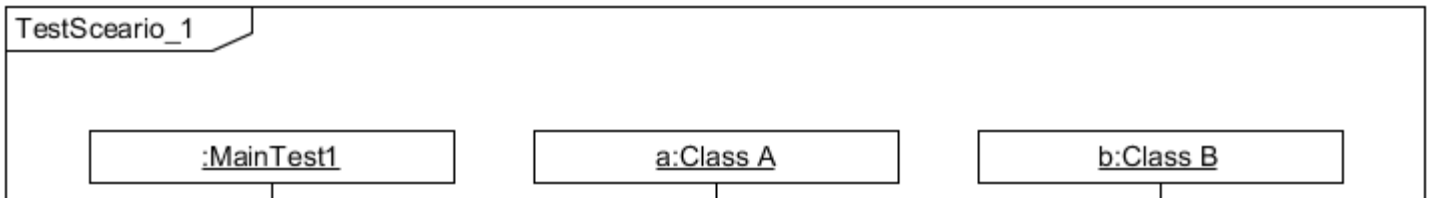
Sequence Diagram

- 제목 설정
  - Title = 제목

```
title=TestSceario_1
```

- 클래스 생성
  - Obj = 클래스명~클래스id

```
obj=_:MainTest1_~id1
obj=_a:Class A_~id2
obj=_b:Class B_~id3
```



Sequence Diagram

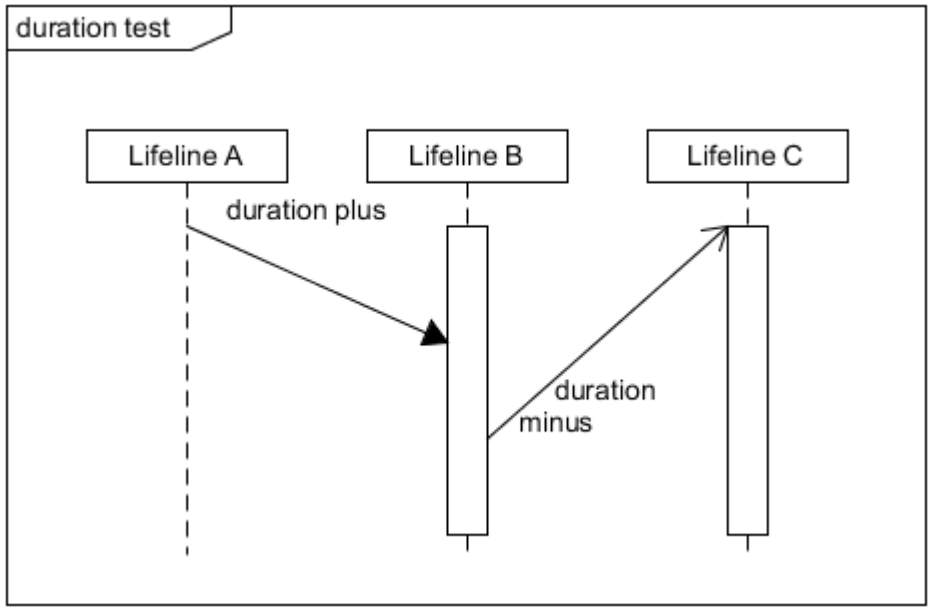
- Lifeline on/off
  - on(or off)=클래스 id

- 메시지(화살표)
  - id1->id2:메시지
  - Id2.>id1:return 메시지
  - id1->id1+
    - 셀프 메시지
    - 뒤에 duration(+n)표시 필수

```
on=id1, id2
id1->id2:init
id2.>id1:init done
on=id3; off=id2
id1->id1+:check status
```

Sequence Diagram

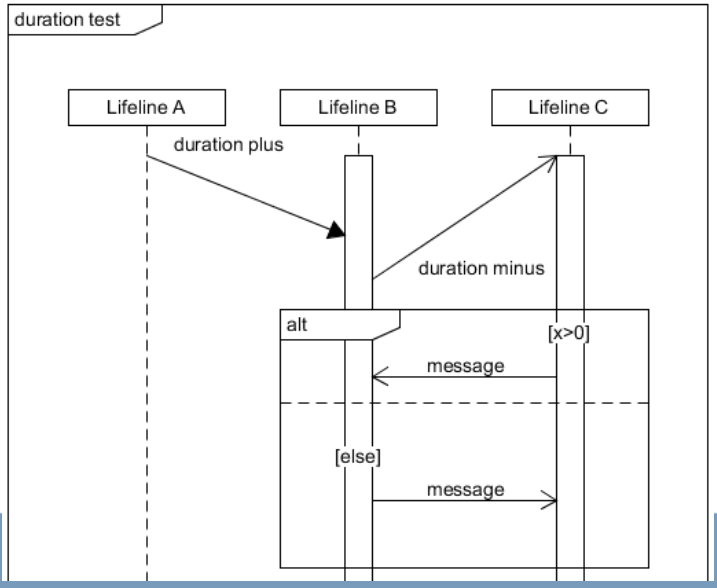
- Duration
  - 일정한 시간이 지난 뒤 메시지가 도착하도록 함
  - +n(or -n)
  - ‘+++’과 ‘+3’은 같음(마찬가지로 ‘-’과 ‘-1’도 같음)




```
title=duration test
obj=Lifeline A~a
obj=Lifeline B~b
obj=Lifeline C~c

a->>>b++ : duration plus; on=b; on=c
tick=2
b->c-3 :    duration minus
```

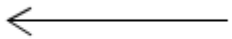
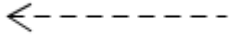
Sequence Diagram

- combinedFragment
  - sub-diagram, alt, loop등의 표시 가능
  - 생성 : combinedFragment=이름~id 포함할 class
  - ..=id : 구역 나눔
  - -- : close 표시




```
combinedFragment=alt~t b c; c:[x>0]
c->b: message
..=t
b:[else]
b->c: message
--
```

Sequence Diagram

- 기타
  - 화살표
    - -> : 메시지 
    - ->>> : call 메시지 
    - .> : return 메시지 
  - a;b : a와 b를 동시에 수행
  - on=id1,id2,id3 : id1, id2, id3의 Lifeline을 동시에 on

Sequence Diagram

Example