Recitation 06 3/16/23, 2:03 PM

# **CS2030S**

# Programming Methodology II

# Recitation 06

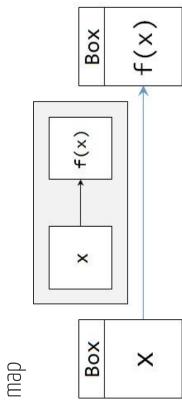
CS2030S: Programming Methodology II -- Adi Yoga Sidi Prabawa

Question 1

# Question 1

### Preliminary - map - flatMap Code

### **Preliminary**

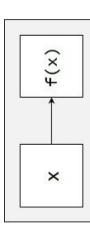


- Open the box
   Operate with function
   Put into new box

### Preliminary - map - flatMap Code

# **Preliminary**

шар



#### ×

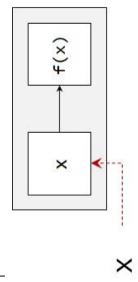
- Open the box
   Operate with function
   Put into new box

# Question 1

### Preliminary - map - flatMap Code

# **Preliminary**

шар



- 1. Open the box
- 2. Operate with function3. Put into new box

Preliminary
- map
- flatMap
Code

**Preliminary** 

шар

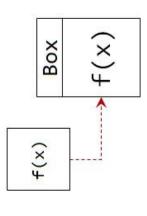
f(x)

- 1. Open the box
- 2. Operate with function3. Put into new box

### Preliminary - map - flatMap Code

# **Preliminary**

шар



- Open the box
   Operate with function
   Put into new box

# Question 1

Preliminary
- map
- flatMap
Code

**Preliminary** 

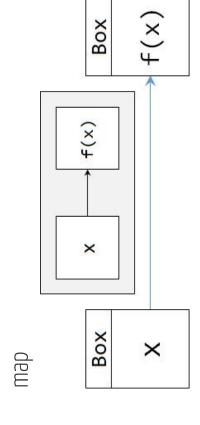
шар

f(x) Box

- Open the box
   Operate with function
   Put into new box

### Preliminary - map - flatMap Code

### **Preliminary**



- Open the box
   Operate with function
   Put into new box

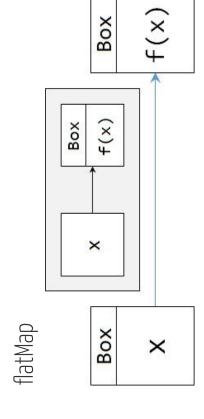
Recitation 06

# Question 1

### Preliminary

-*map* **- flatMap** Code

## **Preliminary**



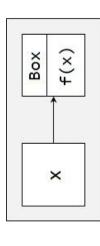
- 1. Open the box
- 2. Operate with function3. Compose the two "context"

# Question 1

### **Preliminary** - *map* **-** *flatMap* **Code**

# **Preliminary**

flatMap



#### ×

- Open the box
   Operate with function
   Compose the two "context"

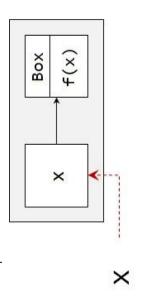
Recitation 06

# Question 1

### **Preliminary** - *map* **-** *flatMap* **Code**

# **Preliminary**

flatMap



- 1. Open the box
- 2. Operate with function 3. Compose the two "context"

**Preliminary** - *map* **-** *flatMap* **Code** 

**Preliminary** 

flatMap

(×) **+**(×) Box

- 1. Open the box
- 2. Operate with function3. Compose the two "context"

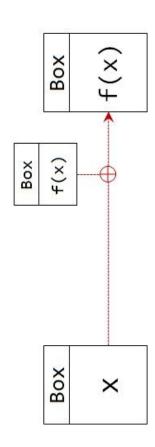
Recitation 06

# Question 1

**Preliminary** - *map* - **flatMap** Code

## **Preliminary**

flatMap



#### Steps

- Open the box
   Operate with function
   Compose the two "context"

# Question 1

**Preliminary** - *map* **-** *flatMap* **Code** 

**Preliminary** 

flatMap

f(x) Box

- Open the box
   Operate with function
   Compose the two "context"

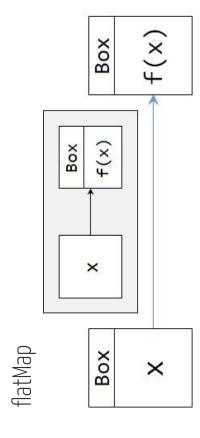
Recitation 06

# Question 1

### Preliminary

*- map* **- flatMap** Code

# **Preliminary**



#### Steps

- 1. Open the box
- 2. Operate with function3. Compose the two "context"

### Preliminary

Code

**- Questions** - Transformed

#### Code

#### **Original**

```
# Maybe<Internship> match(Resume r) {
    if (r == null) {
        return Maybe.none();
    }

# Maybe<List<String> optList = r.getListOfLanguages();

# List<String> list;
    if (optList.equals(Maybe.none())) {
        list = List.of();
    } else {
        list = optList.get(); // cannot call
    }

# if (list.contains("Java")) {
        return Maybe.of(findInternship(list));
    } else {
        return Maybe.none();
}
```

```
Questions

1. What is the type getListOfLanguages()?

2. What is the type contains("Java")?
```

Of

Of

```
3. What is the type of findInternship(list)?
```

### Preliminary **Code**

- Questions **- Transformed** 

#### Code

**Original** 

Transformed

```
Maybe<List<String>> optList = r.getListOfLanguages();
Maybe<Internship> match(Resume r) {
                                                                                                                                                                                                                                                                                                                                                                     return Maybe.of(findInternship(list));
                                                                                                                                                                                   if (optList.equals(Maybe.none())) {
                                                                                                                                                                                                                                                                             list = optList.get(); // cannot call
                                                                                                                                                                                                                                                                                                                                        if (list.contains("Java")) {
                                                            return Maybe.none();
                                                                                                                                                                                                                                                                                                                                                                                                                                    return Maybe.none();
                                                                                                                                                     List<String> list;
                                                                                                                                                                                                               list = List.of();
                              if (r == null) {
                                                                                                                                                                                                                                                 } else {
                                                                                                                                                                                                                                                                                                                                                                                                    } else {
```

18 / 25

3/16/23, 2:03 PM

Question 2

### **Question**

```
Heap
Stack/Heap Diagram
                                                                                                                                                                                                                       Stack
                                                                                                                                                                                                                          Producer<Integer>p = () -> a.get();
                                                                                                                    public int get() {
                                                                  public A(int x) {
                                                                                                                                                                                                              A a = new A(5);
                                                                                                                                               return this.x;
                              class A {
   private int x;
                                                                               this.x = x;
                                                                                                                                  // Line A
Code
```

p.produce();

# Question 3

21/26

Preliminary - *Code* - *Design* Compute

### **Preliminary**

### Original Code

```
static long sum(long n, long result) { \label{eq:static_long} \mbox{if} (n == 0) \ \{
                                                                                                                                    return sum(n - 1, n + result);
                                                                    return result;
                                                                                                  } else {
```

# Rewritten Code

```
return new Recursive\Leftrightarrow(() -> sum(n - 1, n + s));
static Compute<Long> sum(long n, long s) {
                                                                      return new Base<>(() -> s);
                                    if (n = 0) {
                                                                                                           } else {
```

#### Usage

```
Compute<Long> result = sum(n, 0);
                                                                     while (result.isRecursive()) {
static long summer(long n) {
                                                                                                             result = result.recurse();\\
                                                                                                                                                                                   return result.evaluate();
```

# Question 3

### Preliminary - *Code* - **Desígn** Compute

# **Preliminary**

Design

```
return new Recursive<>(() -> sum(n - 1, n + s));
static \; Compute < \!\! Long > sum(long \; n, \, long \; s) \; \{ \\ if \; (n == 0) \; \{
                                                                                  return new Base<>(() -> s);
                                                                                                                             } else {
```

Class Diagram

```
Compute<Long> result = sum(n, 0);
                                                                    while (result.isRecursive()) {
static long summer(long n) {
                                                                                                           result = result.recurse();
                                                                                                                                                                                 return result.evaluate();
```

### Preliminary **Compute**

### Compute

Usage

```
return new Recursive<>(() -> sum(n - 1, n + s));
static \; Compute < \!\! Long > sum(long \; n, \, long \; s) \; \{ \\ if \; (n == 0) \; \{
                                                                                   return new Base<>(() -> s);
                                                                                                                               } else {
```

Classes

```
Compute < Long > result = sum(n, 0);
                                                                     while (result.isRecursive()) {
static long summer(long n) {
                                                                                                           result = result.recurse();
                                                                                                                                                                                 return result.evaluate();
```

jshell>/exit | Goodbye