제어 구조 1.
선택

Control Structure :
Selection

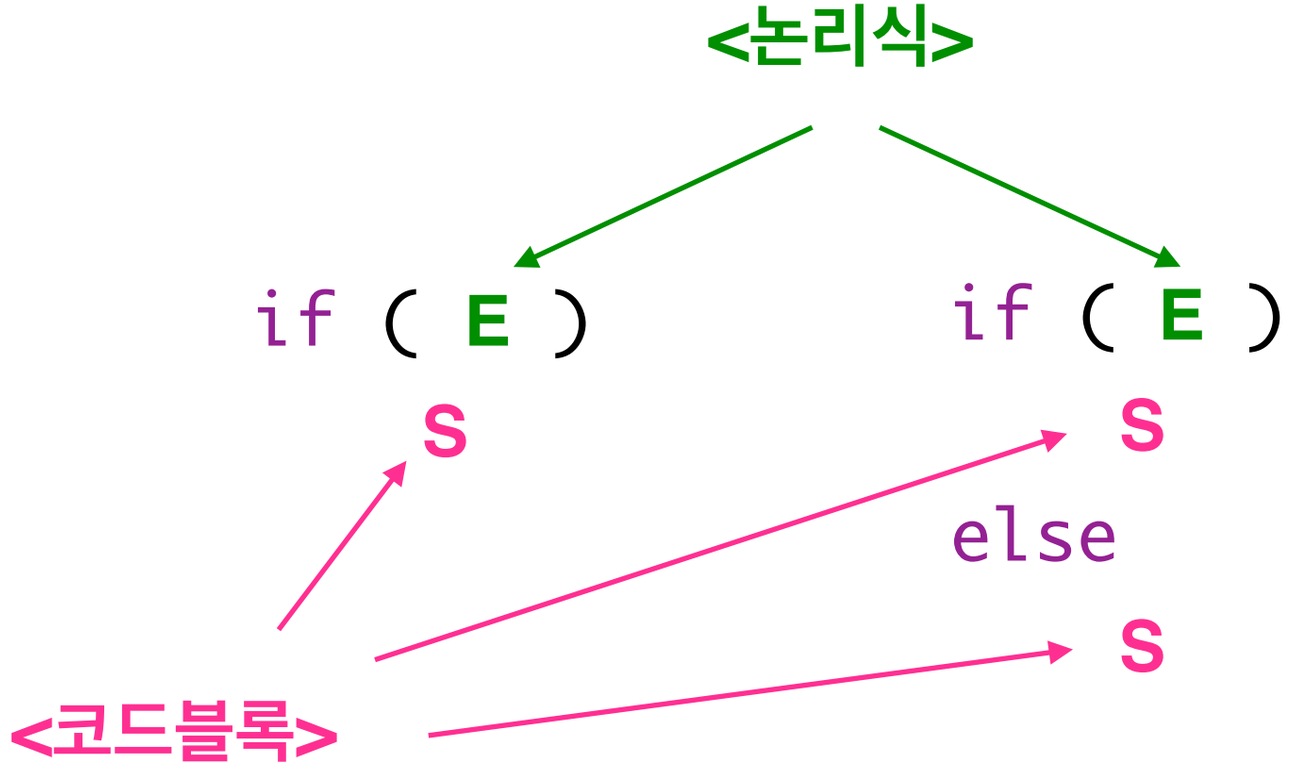
제어 구조

Control Structure

프로그램 실행 순서를 나타내주는 구조

- 기술한 순서대로 실행
- 메소드 호출

선택 구조

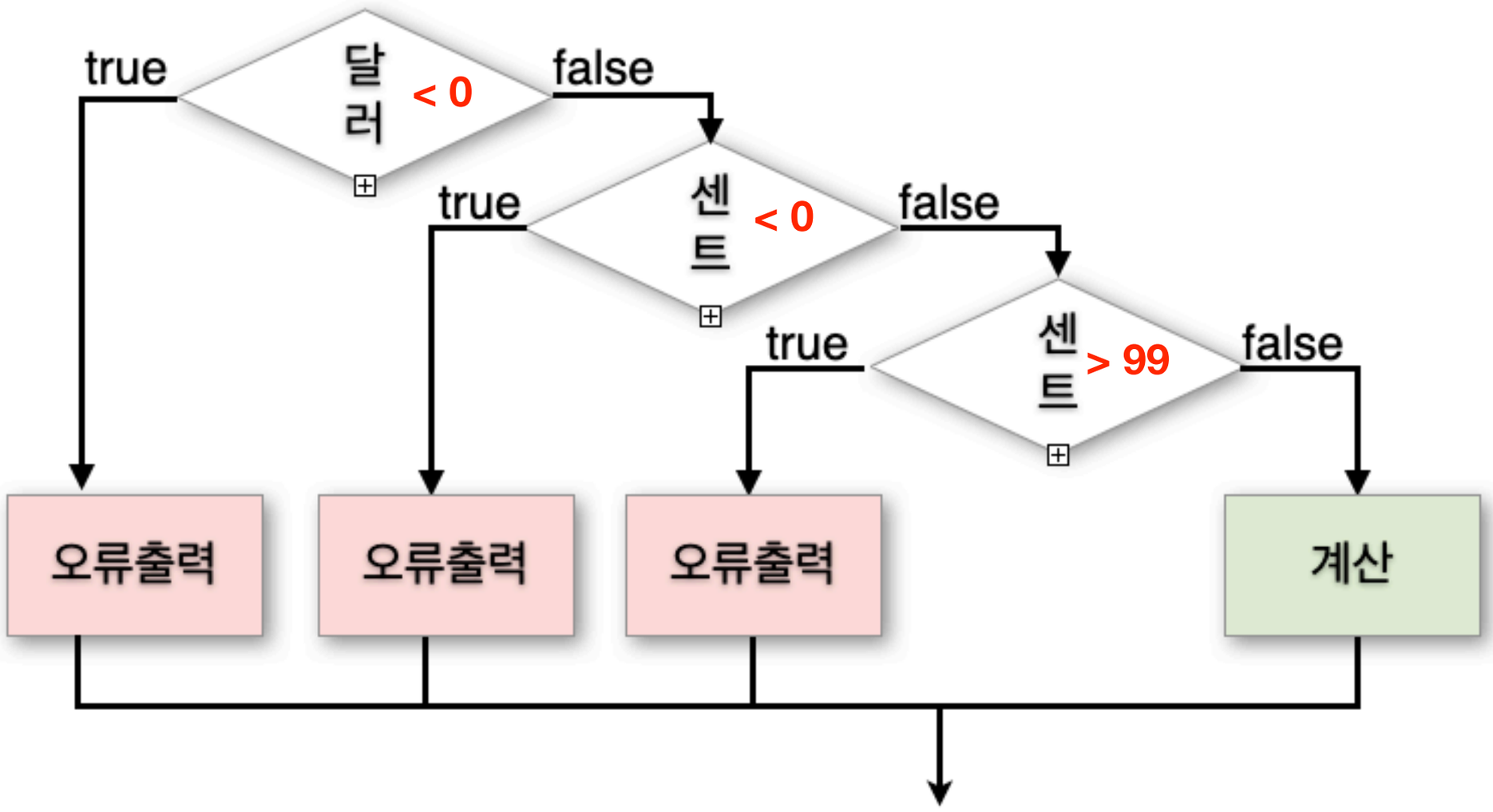


선택 적용 사례

```java
import javax.swing.JOptionPane;

public class Conditional {

    public static void main(String[] args) {
        String input = JOptionPane.showInputDialog("나이를 알려주세요.");
        int age = Integer.parseInt(input);
        if (age < 19)
            System.out.println(age + "세는 미성년입니다.");
        else
            System.out.println(age + "세는 성년입니다.");

        int n = Integer.parseInt(JOptionPane.showInputDialog("역수를 계산해드립니다."));
        if (n == 0)
            System.out.println("0은 역수가 없습니다.");
        else
            System.out.println(n + "의 역수는 " + 1.0/n + " 입니다.");

    }

}
```

동전 거슬러주기 문제

$d.cc



중첩 선택

```
1  public class MakeChange {
2
3      public static void main(String[] args) {
4          int dollars = Integer.parseInt(args[0]);
5          int cents = Integer.parseInt(args[1]);
6          if (dollars < 0) {
7              System.out.println("Error: negative dollars: " + dollars);
8          }
9          else {
10             if (cents < 0) {
11                 System.out.println("Error: negative cents: " + cents);
12             }
13             else {
14                 if (cents > 99) {
15                     System.out.println("Error: bad cents: " + cents);
16                 }
17                 else {
18                     int money = dollars * 100 + cents;
19                     System.out.println("quarters = " + (money / 25));
20                     money = money % 25;
21                     System.out.println("dimes = " + (money / 10));
22                     money = money % 10;
23                     System.out.println("nickels = " + (money / 5));
24                     money = money % 5;
25                     System.out.println("pennies = " + money);
26                 }
27             }
28         }
29     }
30 }
```

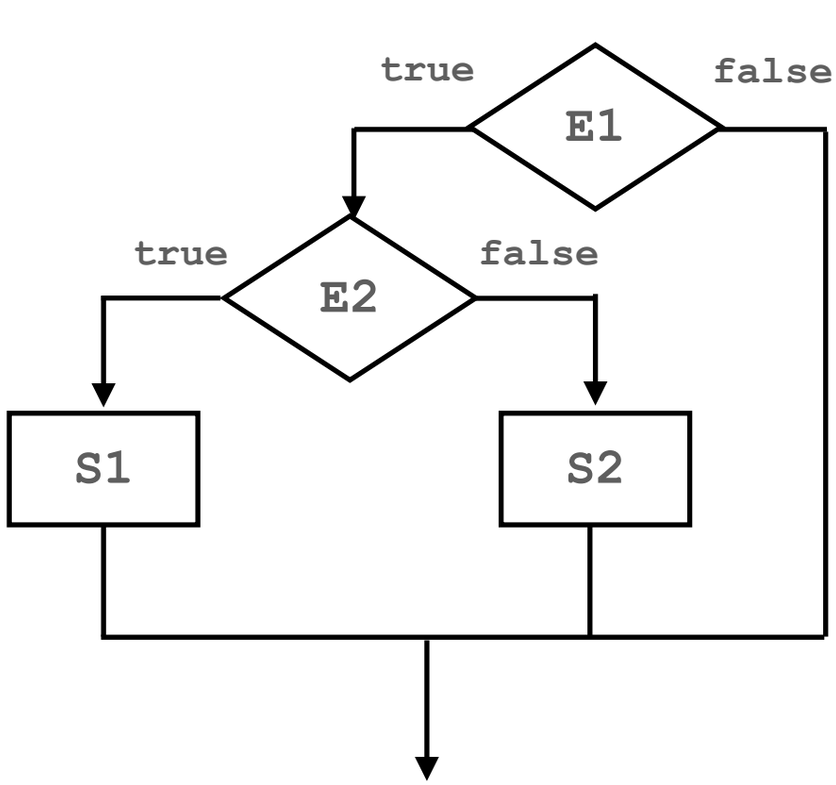
중첩 선택 - 펼치기

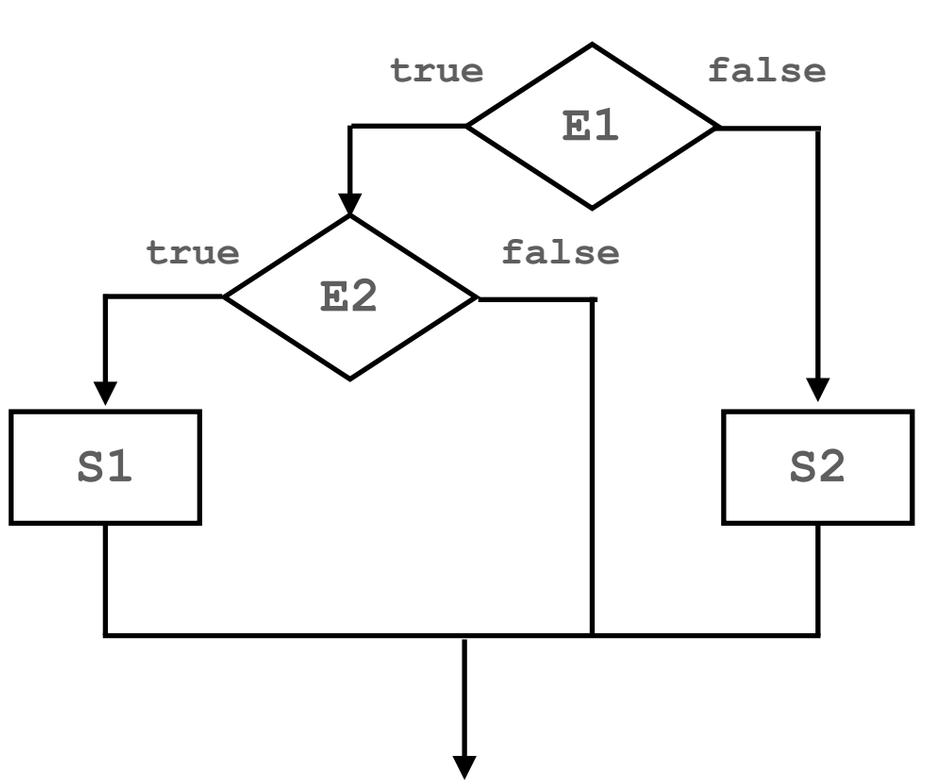
```
1  public class MakeChange {
2
3      public static void main(String[] args) {
4          int dollars = Integer.parseInt(args[0]);
5          int cents = Integer.parseInt(args[1]);
6          if (dollars < 0)
7              System.out.println("Error: negative dollars: " + dollars);
8          else if (cents < 0)
9              System.out.println("Error: negative cents: " + cents);
10         else if (cents > 99)
11             System.out.println("Error: bad cents: " + cents);
12         else {
13             int money = dollars * 100 + cents;
14             System.out.println("quarters = " + (money / 25));
15             money = money % 25;
16             System.out.println("dimes = " + (money / 10));
17             money = money % 10;
18             System.out.println("nickels = " + (money / 5));
19             money = money % 5;
20             System.out.println("pennies = " + money);
21         }
22     }
23 }
```

Dangling Else

```
if ( E1 )
if ( E2 )
S1
else S2
```

which?





논리 연산

연산	이름	의미

`E1 && E2`	논리곱
	conjunction
	and
		`true  && true  => true`
		`false && E2    => false`
		`E1    && false => false`

`E1 || E2`	논리합
	disjunction
	or
		`false || false => false`
		`true  || E2    => true`
		`E1    || true  => true`

`!E`	논리역 negation not	`!true  => false` `!false => true`

단축 계산

Short-circuit Evaluation

논리 연산 적용 사례

```
1  public class MakeChange {
2
3      public static void main(String[] args) {
4          int dollars = Integer.parseInt(args[0]);
5          int cents = Integer.parseInt(args[1]);
6          if (dollars < 0 || cents < 0 || cents > 99) {
7              System.out.println("Error: bad input: " + dollars + " " + cents);
8          }
9          else {
10             int money = dollars * 100 + cents;
11             System.out.println("quarters = " + (money / 25));
12             money = money % 25;
13             System.out.println("dimes = " + (money / 10));
14             money = money % 10;
15             System.out.println("nickels = " + (money / 5));
16             money = money % 5;
17             System.out.println("pennies = " + money);
18         }
19     }
20 }
```

예외를 발생시켜 오류처리하기

```
1  public class MakeChange {
2
3      public static void main(String[] args) {
4          int dollars = Integer.parseInt(args[0]);
5          int cents = Integer.parseInt(args[1]);
6          if (dollars < 0 || cents < 0 || cents > 99) {
7              throw new RuntimeException("Error: bad input: " + dollars + " " + cents);
8          }
9          else {
10             int money = dollars * 100 + cents;
11             System.out.println("quarters = " + (money / 25));
12             money = money % 25;
13             System.out.println("dimes = " + (money / 10));
14             money = money % 10;
15             System.out.println("nickels = " + (money / 5));
16             money = money % 5;
17             System.out.println("pennies = " + money);
18         }
19     }
20 }
```

System.*exit*(0);

```
1  public class MakeChange {
2
3      public static void main(String[] args) {
4          int dollars = Integer.parseInt(args[0]);
5          int cents = Integer.parseInt(args[1]);
6          if (dollars < 0 || cents < 0 || cents > 99) {
7              System.exit(0);
8          }
9          else {
10             int money = dollars * 100 + cents;
11             System.out.println("quarters = " + (money / 25));
12             money = money % 25;
13             System.out.println("dimes = " + (money / 10));
14             money = money % 10;
15             System.out.println("nickels = " + (money / 5));
16             money = money % 5;
17             System.out.println("pennies = " + money);
18         }
19     }
20 }
```

프로그램 실행 끝!

switch

```
if (i == 2)
    System.out.println("2");
else {
    if (i == 5) {
        System.out.println("5");
        j = j + 1;
    }
    else {
        if (i == 7) {
            System.out.println("7");
            j = j - 1;
        }
        else
            System.out.println("none");
    }
}
```

```
switch (i) {
    case 2: {
        System.out.println("2");
        break;
    }
    case 5: {
        System.out.println("5");
        j = j + 1;
        break;
    }
    case 7: {
        System.out.println("7");
        j = j - 1;
        break;
    }
    default:
        System.out.println("none");
}
```

switch

```
switch ( EXPRESSION ) {
    case VALUE : {
        STATEMENTS
        break;
    }
    case VALUE : {
        STATEMENTS
        break;
    }
    /* ... */
    case VALUE : {
        STATEMENTS
        break;
    }
    default:
        STATEMENTS
}
```

매우 제한적!

값만 사용 가능

break를 쓰지 않으면
끝으로 가지 않고
다음 case로 넘어감

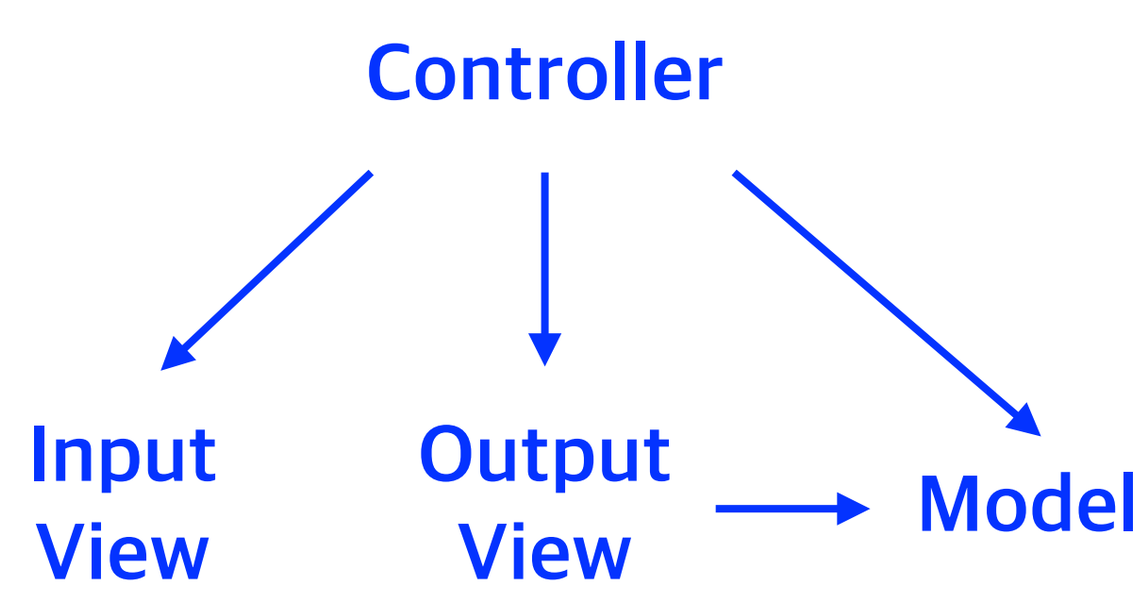
권장하지 않음!!!!!

switch vs. **if … else if … else**

```
switch (i) {
    case 2: {
        System.out.println("2");
        break;
    }
    case 5: {
        System.out.println("5");
        j = j + 1;
        break;
    }
    case 7: {
        System.out.println("7");
        j = j - 1;
        break;
    }
    default:
        System.out.println("none");
}
```

```
if (i == 2)
    System.out.println("2");
else if (i == 5) {
    System.out.println("5");
    j = j + 1;
}
else if (i == 7) {
    System.out.println("7");
    j = j - 1;
}
else
    System.out.println("none");
```

MVC 아키텍처
전형 패턴



재사용

조립식

교체용이

애플리케이션 설계+구현 절차

스텝 1. 유즈케이스(use-case) 수집 => View(사용자 인터페이스) 설계

스텝 2. 클래스 다이어그램 작성

스텝 3. 클래스 별로 클래스 명세 작성

스텝 4. 클래스 별로 코드 작성 및 테스트

스텝 5. 통합 테스트

사례 학습

지갑 애플리케이션

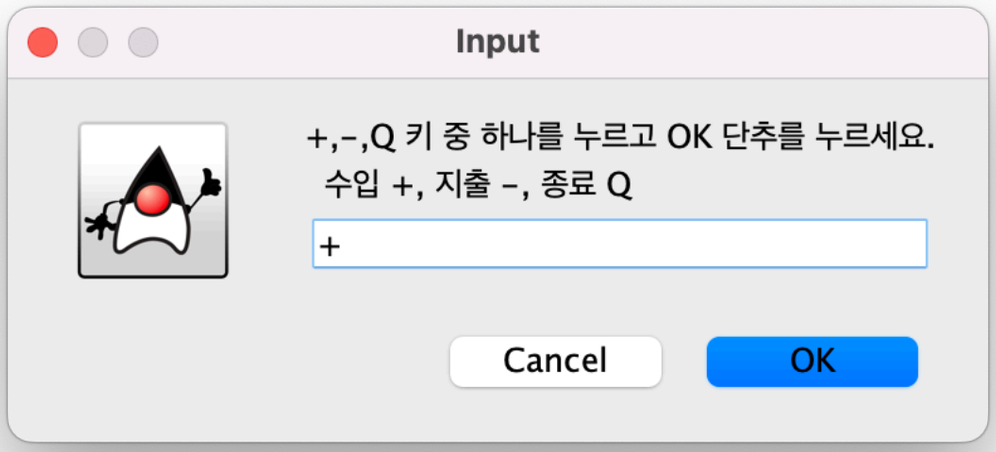
수입

지출

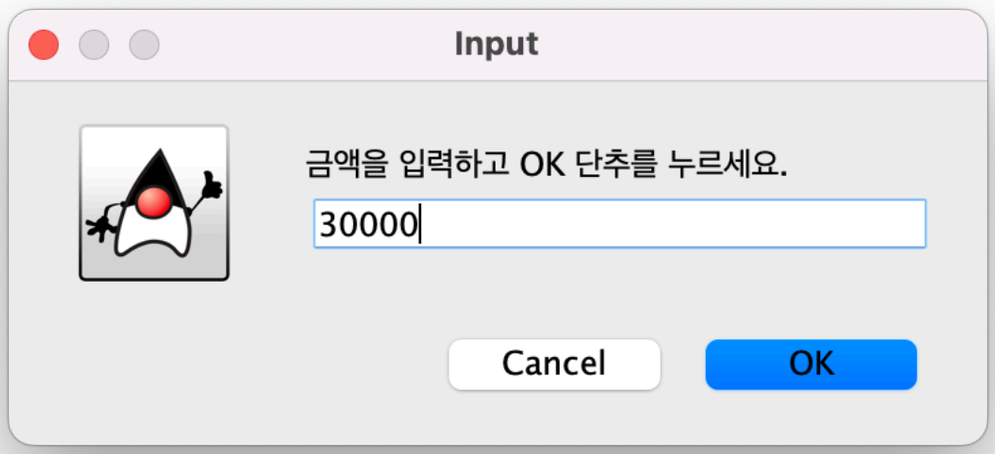
스텝 1. 유즈케이스 수집 + 사용자 인터페이스 설계

사용자 입력

- 서비스 : 수입(+), 지출(-), 종료(Q,q)



- 금액 : 정수 (음수 불가)



에리카

Input

+,-,Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 종료 Q

+

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

30000

Cancel OK

에리카

30000 수입
잔액 = 30000

에리카

30000 수입
잔액 = 30000

Input

+,-,Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 종료 Q

-

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

12000

Cancel OK

에리카

12000 지출
잔액 = 18000

에리카

12000 지출
잔액 = 18000

Input

+,-,Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 종료 Q

-

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

20000

Cancel OK

에리카

지출 실패
잔액 = 18000

에리카

지출 실패
잔액 = 18000

Input

+,-,Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 종료 Q

-

Cancel OK

Input

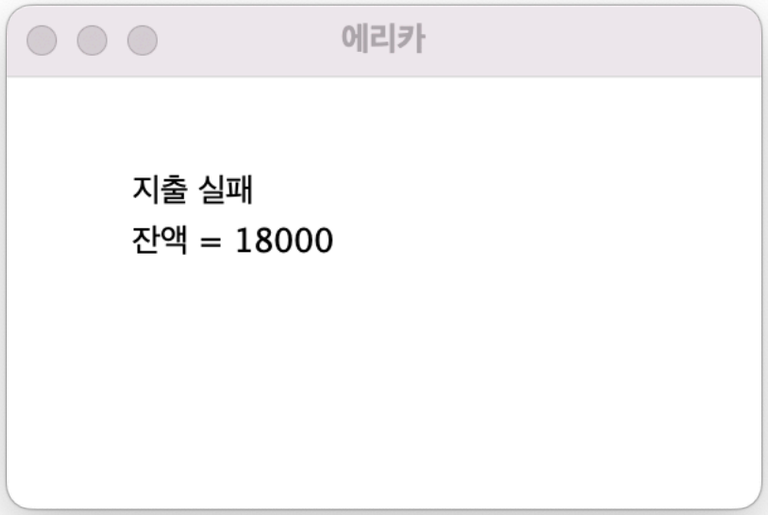
금액을 입력하고 OK 단추를 누르세요.

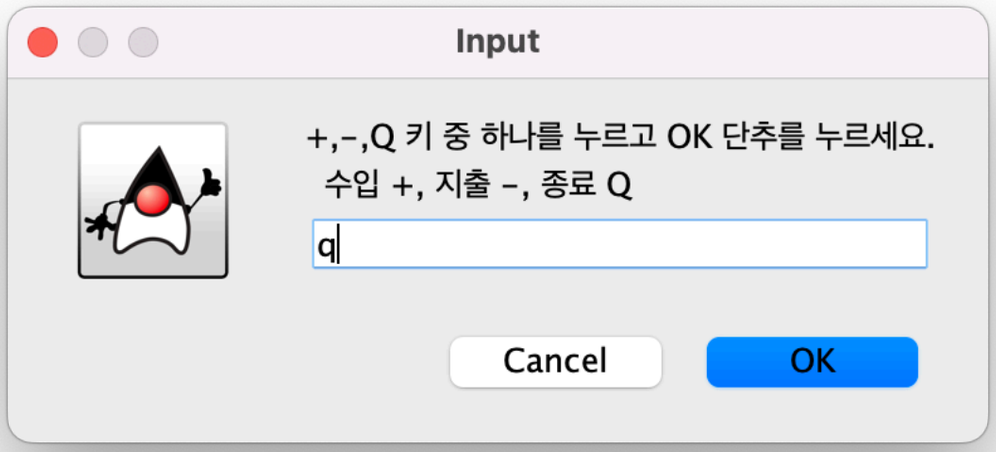
-5000000

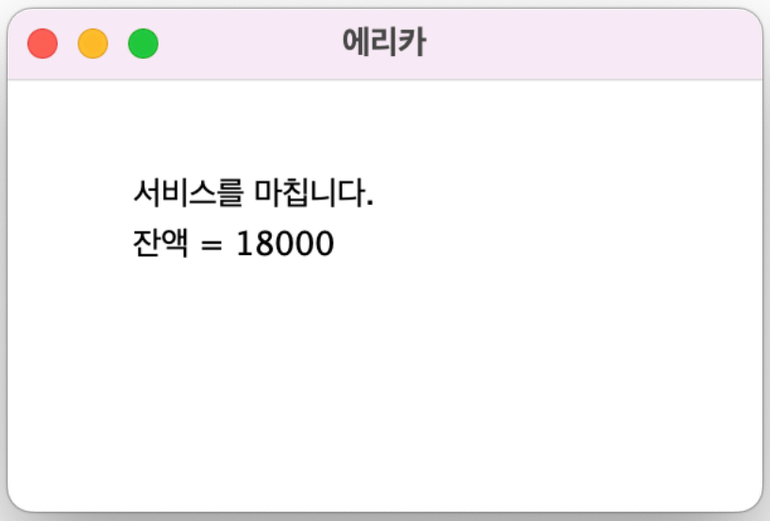
Cancel OK

에리카

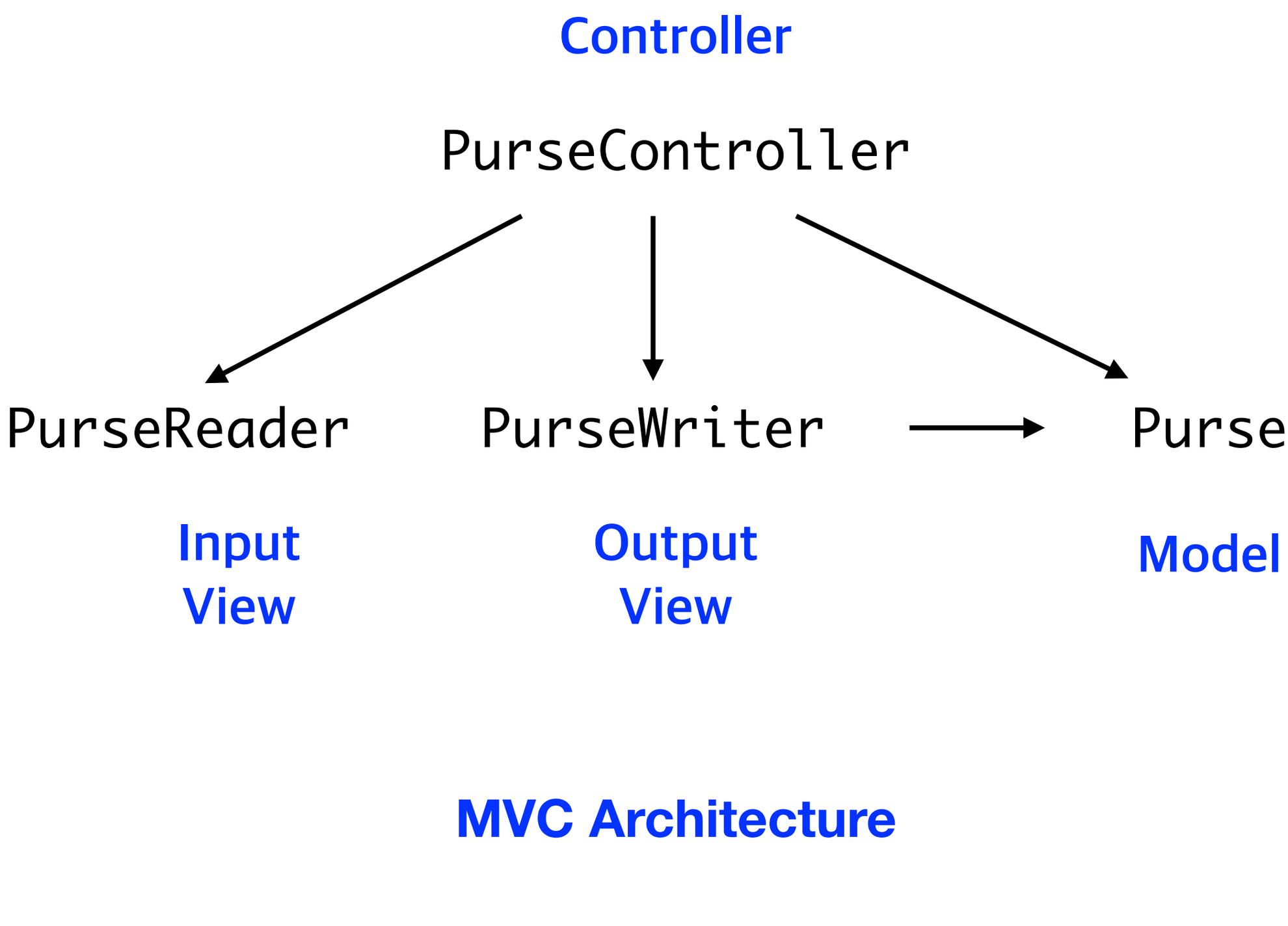
지출 실패
잔액 = 18000


에리카
지출 실패
잔액 = 18000


Input
+,-,Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 종료 Q
q
Cancel
OK


에리카
서비스를 마칩니다.
잔액 = 18000

스텝 2. 클래스 다이어그램 작성

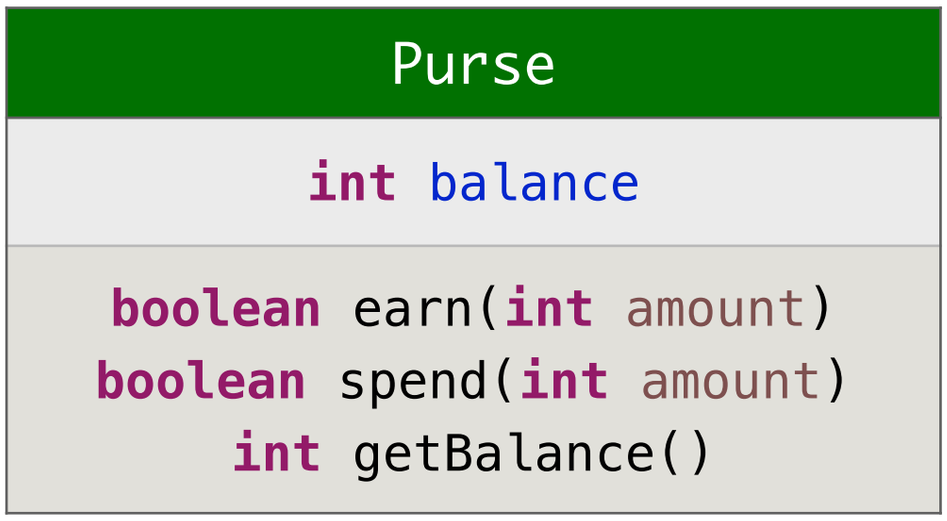


MVC Architecture

Starter

Controller

Model



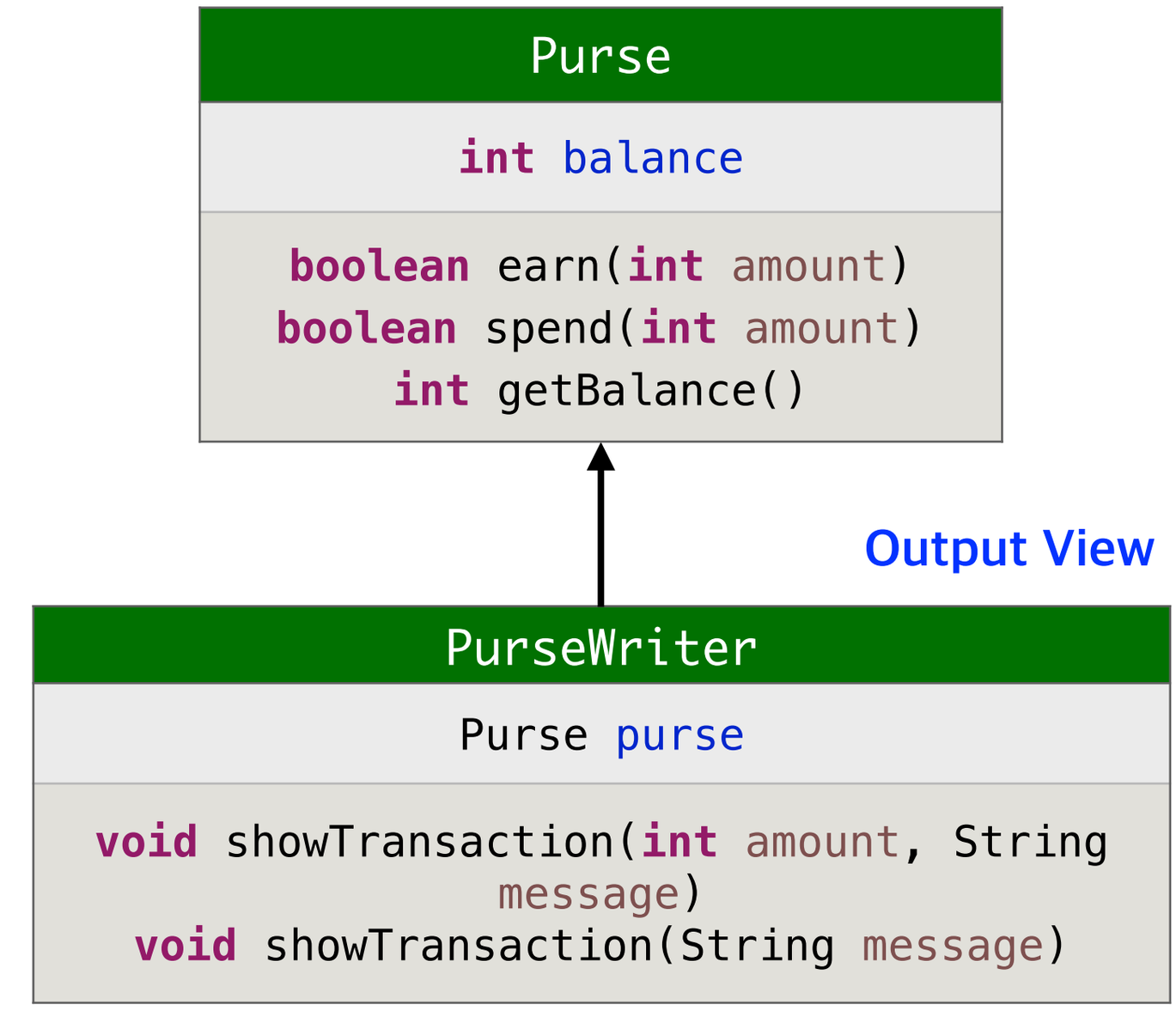
Input View

Output View

Starter

Controller

Model



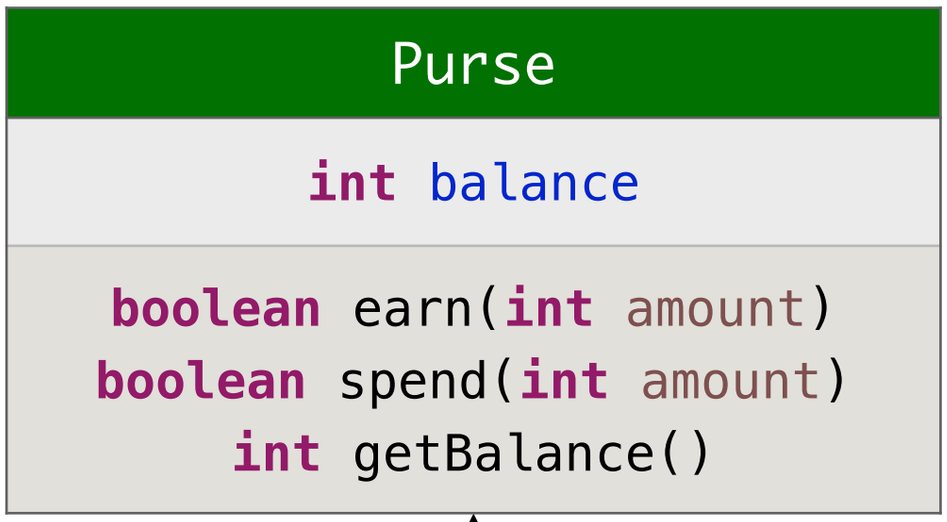
Input View

Output View

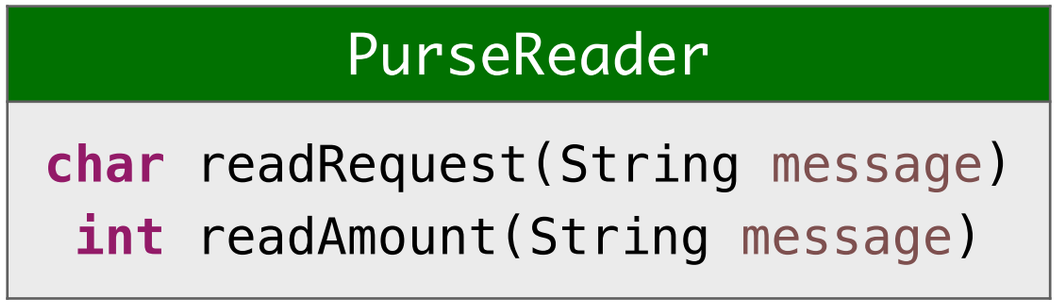
Starter

Controller

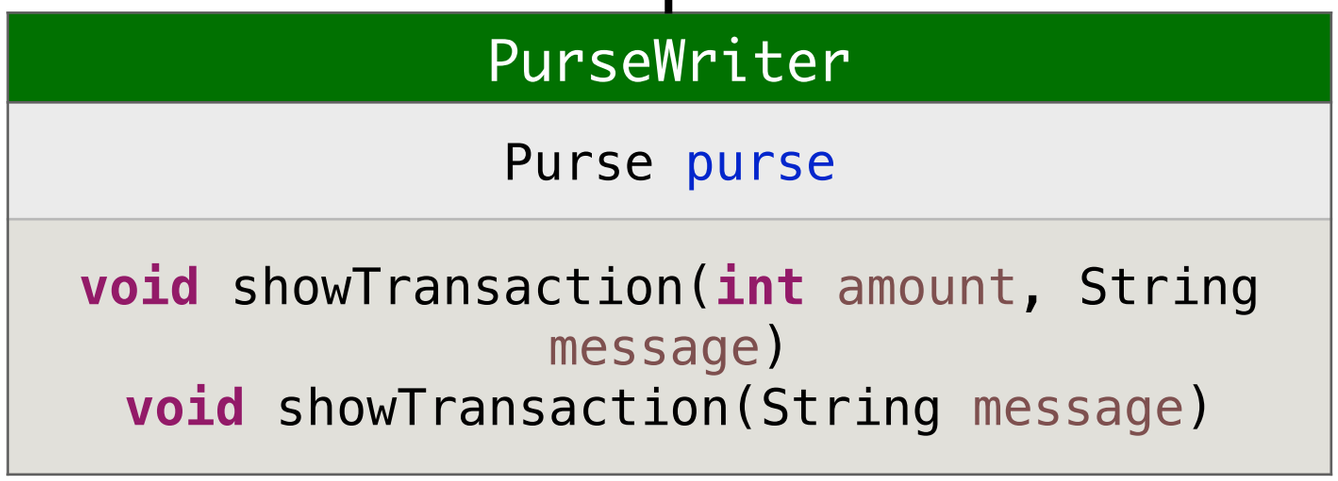
Model



Input View



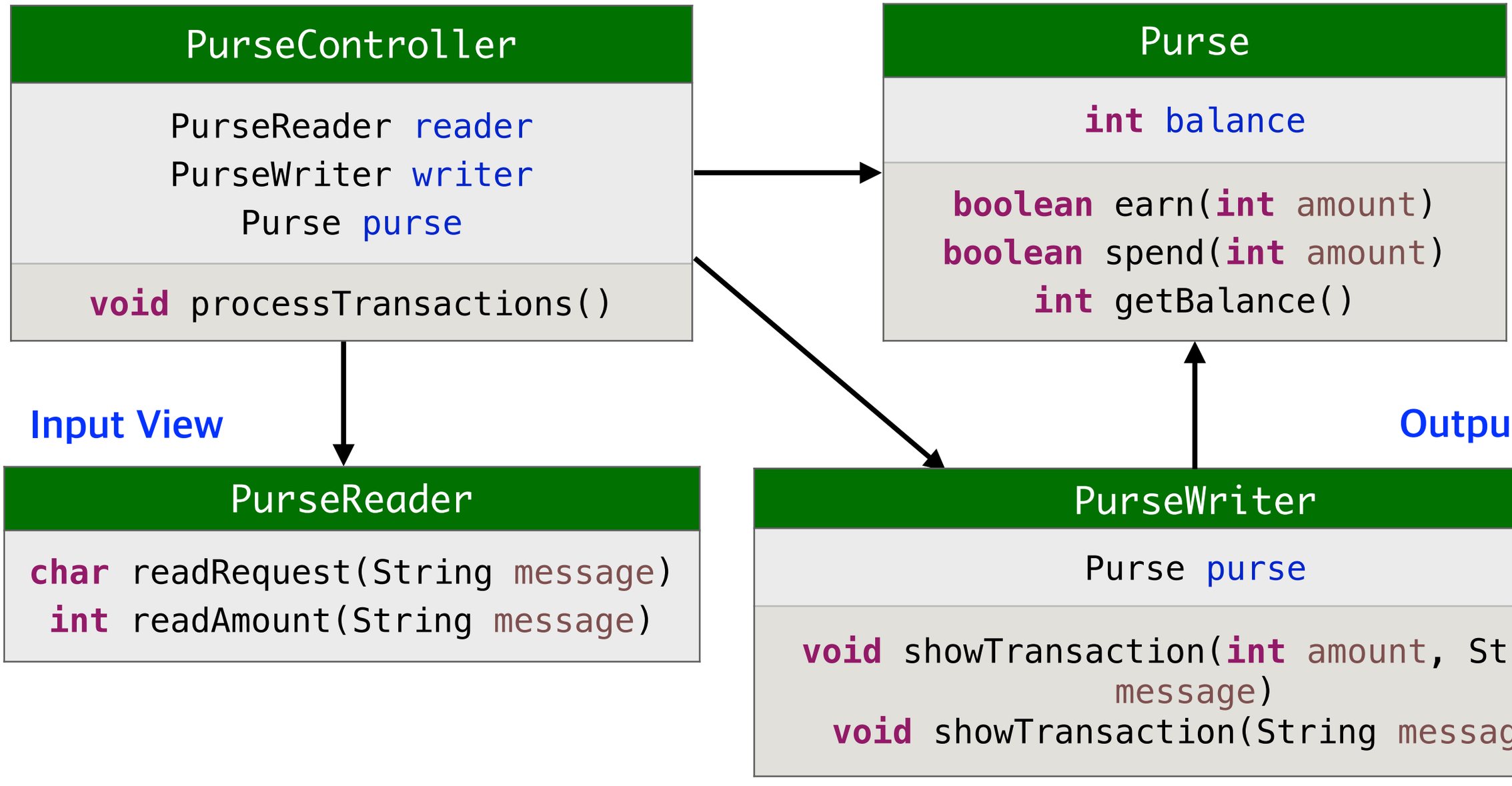
Output View



Starter

Controller

Model



Input View

Output View

Starter

```
PurseManager
static void main(String[] args)
```

Controller

```
PurseController
PurseReader reader
PurseWriter writer
Purse purse
void processTransactions()
```

Model

```
Purse
int balance
boolean earn(int amount)
boolean spend(int amount)
int getBalance()
```

Input View

```
PurseReader
char readRequest(String message)
int readAmount(String message)
```

Output View

```
PurseWriter
Purse purse
void showTransaction(int amount, String message)
void showTransaction(String message)
```

스텝 3. 클래스 명세 작성

Model

class	Purse	계좌 개설 및 운영
constructor		
field	`private int balance`	• 초기 잔액은 항상 0 • 불변식(Invariant) : balance >= 0
method	`boolean earn(int amount)`	• 인수가 음수이면 잔액 변동없이 false를 리턴하고, • 그렇지 않고 인수가 0 이상의 정수이면 잔액을 증가시키고 true를 리턴한다.
	`boolean spend(int amount)`	• 인수가 음수이거나 잔액을 초과하면 잔액 변동없이 false를 리턴하고, • 그렇지 않으면 잔액을 감소시키고 true를 리턴한다.
	`int getBalance()`	• 잔액을 리턴한다.

accessor method

mutator method

스텝 4. 코드 작성 및 테스트

```
public class Purse {

    private int balance;

    public boolean earn(int amount) {
        if (amount < 0)
            return false;
        else  {
            balance += amount;
            return true;
        }
    }

    public boolean spend(int amount) {
        if (amount < 0 || amount > balance)
            return false;
        else  {
            balance -= amount;
            return true;
        }
    }

    public int getBalance() {
        return balance;
    }

}
```

Purse 클래스가
정상 작동하는지
임시 main 메소드를
작성하여 테스트해보자
(테스트 완료후 제거)

스텝 3. 클래스 명세 작성

View

class	PurseWriter **extends** JPanel	
field	**private** Purse purse	
	String last_transaction	
constructor	PurseWriter(String title, **int** x, **int** y, Purse p)	첫 문자열 인수는 윈도우의 타이틀로 만들고, 둘째, 셋째 인수는 윈도우의 위치좌표 넷째 인수는 필드 변수 세팅
method	**void** paintComponent(Graphics g)	최근 거래 결과와 잔액을 윈도우에 출력
	void showTransaction(**int** amount, String message) **void** showTransaction(String message)	거래 결과를 last_transaction에 저장하고 윈도우를 다시 그림

스텝 4. 코드 작성 및 테스트

```
import javax.swing.*;
import java.awt.*;

public class PurseWriter extends JPanel {

    private Purse purse;
    String last_transaction;

    public PurseWriter(String title, int x, int y, Purse p) {
        purse = p;
        JFrame f = new JFrame();
        f.getContentPane().add(this);
        f.setTitle(title);
        f.setLocation(x, y);
        f.setSize(300, 200);
        f.setVisible(true);
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    protected void paintComponent(Graphics g) {
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, 300, 222);
        g.setColor(Color.BLACK);
        g.drawString(last_transaction, 50, 50);
        g.drawString("잔액 = " + purse.getBalance(), 50, 70);
    }

    public void showTransaction(int amount, String message) {
        last_transaction = amount + " " + message;
        this.repaint();
    }

    public void showTransaction(String message) {
        last_transaction = message;
        this.repaint();
    }
}
```

스텝 3. 클래스 명세 작성

View

class	PurseReader	
field		
method	`char readRequest(String message)`	• 메시지 문자열을 인수로 받아서 사용자에게 보여주고 • 받은 입력 문자열의 첫 문자를 리턴
	`int readAmount(String message)`	• 메시지 문자열을 인수로 받아서 사용자에게 보여주고 • 받은 입력 문자열을 정수로 변환하여 리턴

스텝 4. 코드 작성 및 테스트

```
import javax.swing.*;

public class PurseReader {

    public char readRequest(String message) {
        String input = JOptionPane.showInputDialog(message);
        return input.charAt(0);
    }

    public int readAmount(String message) {
        String input = JOptionPane.showInputDialog(message);
        input = input.trim();
        if (input.length() > 0)
            return Integer.parseInt(input);
        else {
            JOptionPane.showMessageDialog(null, "금액 입력 오류");
            return 0;
        }
    }

}
```

스텝 3. 클래스 명세 작성

Controller

class	PurseController	

field	`private PurseReader reader;`
	`private PurseWriter writer;`
	`private Purse purse;`	

constructor	`PurseController(PurseReader r, PurseWriter w, Purse p)`	• 전달받은 인수로 필드 변수를 세팅
method	`void processTransactions()`	• 서비스는 수입 +, 지출 -, 종료 Q 중 하나를 선택하여 입력하게 하고, +,-를 선택한 경우 이어서 금액을 입력하게 한다. • 주어진 금액을 증감한다. • Q 요청을 할 때까지 고객의 요청을 무한 반복 처리

스텝 4. 코드 작성 및 테스트

```
public class PurseController {

    private PurseReader reader;
    private PurseWriter writer;
    private Purse purse;

    public PurseController(PurseReader r, PurseWriter w, Purse p) {
        reader = r; writer = w; purse = p;
    }

    public void processTransactions() {
        String message = "+,-,Q 키 중 하나를 누르고 OK 단추를 누르세요.\n  수입 +, 지출 -, 종료 Q";
        char request = reader.readRequest(message);
        message = "금액을 입력하고 OK 단추를 누르세요.";
        int amount;
        if (request == 'Q' || request == 'q' ) {
            writer.showTransaction("서비스를 마칩니다.");
            return;
        }
        else if (request == '+') {
            amount = reader.readAmount(message);
            if (purse.earn(amount))
                writer.showTransaction(amount, "수입");
            else
                writer.showTransaction("수입 실패");
        }
        else if (request == '-') {
            amount = reader.readAmount(message);
            if (purse.spend(amount))
                writer.showTransaction(amount, "지출");
            else
                writer.showTransaction("지출 실패");
        }
        else
            writer.showTransaction("요청 오류");
        this.processTransactions();
    }
}
```

스텝 4. 코드 작성 및 테스트

PurseController 테스트 전용 더미 클래스

```
// dummy class for test
public class Purse {

    private int balance; // no field variables

    public boolean earn(int amount) {
        System.out.println("입금 = " +  + "원")
        return true;
    }

    public boolean spend(int amount) {
        System.out.println("출금 = " +  + "원")
        return true;
    }

    public int getBalance() {
        return 0;
    }

}
```

Starter

애플리케이션 작동을 위한 무대 설치

```
public class PurseManager {

   public static void main(String[] args) {
      PurseReader r = new PurseReader();
      Purse p = new Purse();
      PurseWriter w = new PurseWriter("에리카", 300, 0, p);
      new PurseController(r, w, p).processTransactions();
   }

}
```

스텝 5. 통합 테스트

실습 #1

지갑 애플리케이션
일본화폐 지갑 추가

일본화폐 지갑을 추가하여 별도로 관리

에리카(원화)

활성
잔액 = 0

에리카(일화)

비활성
잔액 = 0

Input

+, -, K, J, Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 한화 K, 일화 J, 종료 Q

+

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

100000

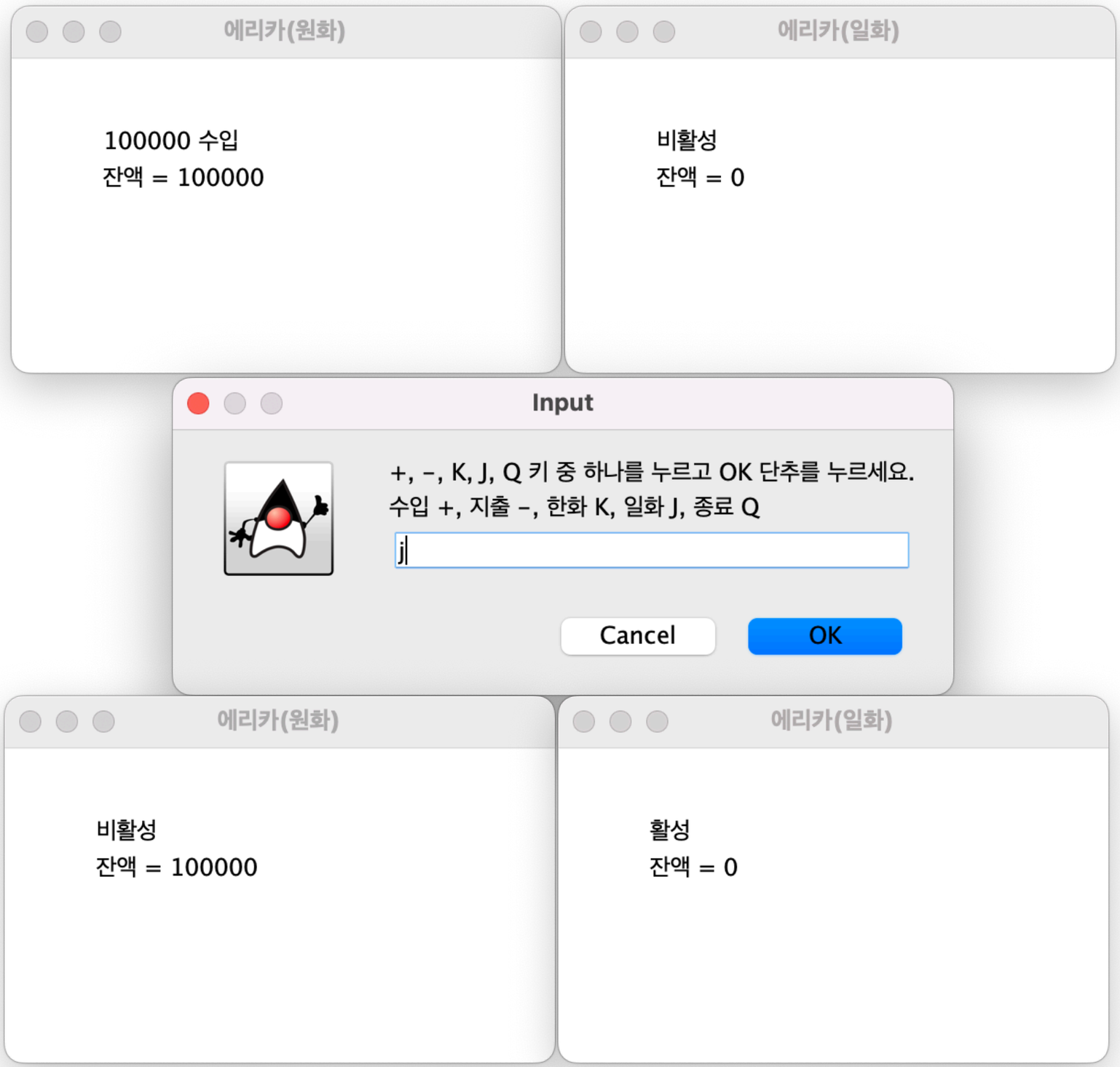
Cancel OK

에리카(원화)

100000 수입
잔액 = 100000

에리카(일화)

비활성
잔액 = 0


에리카(원화)
100000 수입
잔액 = 100000
에리카(일화)
비활성
잔액 = 0
Input
+, -, K, J, Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 한화 K, 일화 J, 종료 Q
j
Cancel
OK
에리카(원화)
비활성
잔액 = 100000
에리카(일화)
활성
잔액 = 0

에리카(원화)

비활성
잔액 = 100000

에리카(일화)

활성
잔액 = 0

Input

+, -, K, J, Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 한화 K, 일화 J, 종료 Q

+

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

20000

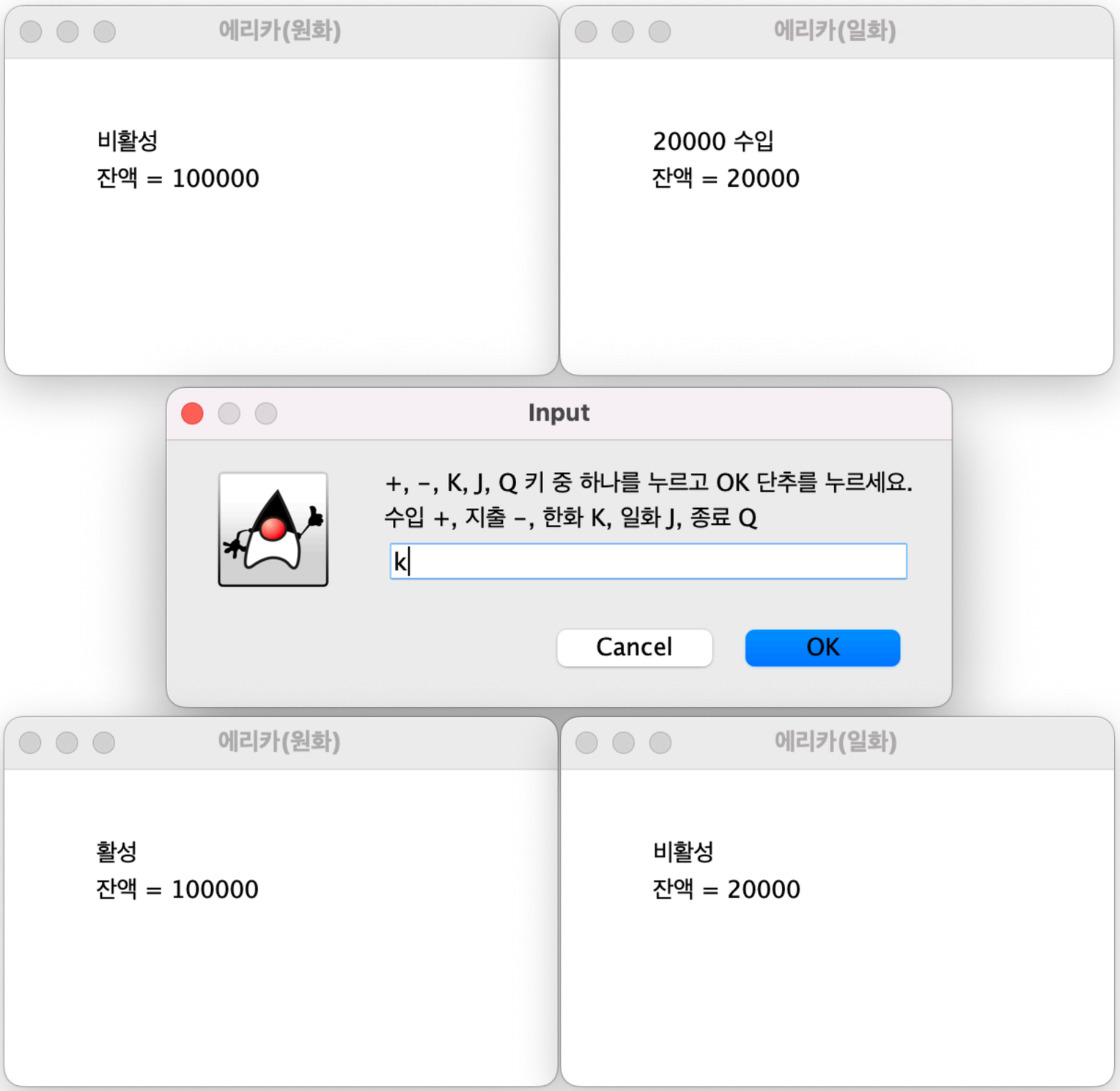
Cancel OK

에리카(원화)

비활성
잔액 = 100000

에리카(일화)

20000 수입
잔액 = 20000


에리카(원화)
비활성
잔액 = 100000
에리카(일화)
20000 수입
잔액 = 20000
Input
+, -, K, J, Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 한화 K, 일화 J, 종료 Q
k
Cancel
OK
에리카(원화)
활성
잔액 = 100000
에리카(일화)
비활성
잔액 = 20000

Starter

PurseManager

```
static void main(String[] args)
```

Controller

PurseController

```
PurseReader reader
PurseWriter writer
Purse purse
```

```
void processTransactions()
```

Model

Purse

```
int balance
```

```
boolean earn(int amount)
boolean spend(int amount)
int getBalance()
```

1 2

1

2

Input View

PurseReader

```
char readRequest(String message)
 int readAmount(String message)
```

Output View

PurseWriter

```
Purse purse
```

```
void showTransaction(int amount, String message)
void showTransaction(String message)
```

Starter

PurseManager

```
static void main(String[] args)
```

Controller

PurseController

```
PurseReader reader
PurseWriter writer_won
PurseWriter writer_yen
PurseWriter writer
Purse purse_won
Purse purse_yen
Purse purse
```

```
void processTransactions()
```

Model

Purse

```
int balance
```

```
boolean earn(int amount)
boolean spend(int amount)
int getBalance()
```

1 2

1

2

Input View

PurseReader

```
char readRequest(String message)
int readAmount(String message)
```

Output View

PurseWriter

```
Purse purse
```

```
void showTransaction(int amount, String message)
void showTransaction(String message)
```

Controller

class	PurseController	

field	`private PurseReader reader;`
	`private PurseWriter writer_won;`
	`private PurseWriter writer_yen;`
	`private PurseWriter writer;`
	`private Purse purse_won;`
	`private Purse purse_yen;`
	`private Purse purse;`
		• 원화와 일화 지갑이 별도 존재
		• 둘 중에서 현재 처리중인 지갑을 writer 와 purse 가 기억

constructor	`PurseController(PurseReader r, PurseWriter w1, PurseWriter w2, Purse p1, Purse p2)`	• 전달받은 인수로 필드 변수를 세팅. • writer와 purse 는 w1과 p1으로 각각 세팅
method	`void processTransactions()`	• 서비스는 수입 +, 지출 -, 한화 <, 일화 >, 종료 Q 중 하나를 선택하여 입력하게 하고, +,-를 선택한 경우 이어서 금액을 입력하게 한다. • <, > 경우, 전자는 원화 지갑을 활성화, 후자는 일화 지갑을 활성화시킨다. • +,- 경우, 주어진 금액을 증감한다. • Q 요청을 할 때까지 고객의 요청을 무한 반복 처리

```java
public class PurseController {

    private PurseReader reader;
    private PurseWriter writer_won;
    private PurseWriter writer_yen;
    private PurseWriter writer; // remembers active writer
    private Purse purse_won;
    private Purse purse_yen;
    private Purse purse; // remembers active purse

    public PurseController(PurseReader r, PurseWriter w1, PurseWriter w2,
                           Purse p1, Purse p2) {
        reader = r;
        writer_won = w1;
        writer_yen = w2;
        purse_won = p1;
        purse_yen = p2;
        writer = w1;
        purse = p1;
        writer_won.showTransaction("활성");
        writer_yen.showTransaction("비활성");
    }

    public void processTransactions() {
        String message = "+, -, K, J, Q 키 중 하나를 누르고 OK 단추를 누르세요.\n";
        message += "수입 +, 지출 -, 한화 K, 일화 J, 종료 Q";
        char request = reader.readRequest(message);
        message = "금액을 입력하고 OK 단추를 누르세요.";
        int amount;
        if (request == 'Q' || request == 'q' ) {
            writer_won.showTransaction("서비스를 마칩니다.");
            writer_yen.showTransaction("서비스를 마칩니다.");
            return;
        }
```

```
public void processTransactions() {
    String message = "+, -, K, J, Q 키 중 하나를 누르고 OK 단추를 누르세요.\n";
    message += "수입 +, 지출 -, 한화 K, 일화 J, 종료 Q";
    char request = reader.readRequest(message);
    message = "금액을 입력하고 OK 단추를 누르세요.";
    int amount;
    if (request == 'Q' || request == 'q' ) {
        writer_won.showTransaction("서비스를 마칩니다.");
        writer_yen.showTransaction("서비스를 마칩니다.");
        return;
    }
    else if (request == '+') {
        amount = reader.readAmount(message);
        if (purse.earn(amount))
            writer.showTransaction(amount, "수입");
        else
            writer.showTransaction("수입 실패");
    }
    else if (request == '-') {
        amount = reader.readAmount(message);
        if (purse.spend(amount))
            writer.showTransaction(amount, "지출");
        else
            writer.showTransaction("지출 실패");
    }
    else if (request == 'K' || request == 'k')
        switchPurse(writer_won, purse_won);
    else if (request == 'J' || request == 'j')
        switchPurse(writer_yen, purse_yen);
    else
        writer.showTransaction("요청 오류");
    this.processTransactions();
}

private void switchPurse(PurseWriter w, Purse p) {
```

```
            writer_won.showTransaction("서비스를 마칩니다.");
            writer_yen.showTransaction("서비스를 마칩니다.");
            return;
        }
        else if (request == '+') {
            amount = reader.readAmount(message);
            if (purse.earn(amount))
                writer.showTransaction(amount, "수입");
            else
                writer.showTransaction("수입 실패");
        }
        else if (request == '-') {
            amount = reader.readAmount(message);
            if (purse.spend(amount))
                writer.showTransaction(amount, "지출");
            else
                writer.showTransaction("지출 실패");
        }
        else if (request == 'K' || request == 'k')
            switchPurse(writer_won, purse_won);
        else if (request == 'J' || request == 'j')
            switchPurse(writer_yen, purse_yen);
        else
            writer.showTransaction("요청 오류");
        this.processTransactions();
    }

    private void switchPurse(PurseWriter w, Purse p) {
        writer.showTransaction("비활성");
        writer = w;
        purse = p;
        writer.showTransaction("활성");
    }
}
```

```java
public class PurseManager {

   public static void main(String[] args) {
      PurseReader r = new PurseReader();
      Purse p1 = new Purse();
      Purse p2 = new Purse();
      PurseWriter w1 = new PurseWriter("에리카(원화)", 300, 0,  p1);
      PurseWriter w2 = new PurseWriter("에리카(일화)", 600, 0, p2);
      new PurseController(r, w1, w2, p1, p2).processTransactions();
   }

}
```

실습 #2

지갑 애플리케이션
환전 기능 추가

환전하여 다른 지갑으로 옮기는 기능

가상 고정 환율 : 10원 = 1엔

에리카(원화)

비활성
잔액 = 250000

에리카(일화)

활성
잔액 = 15000

Input

+, -, K, J, >, <, Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 한화 K, 일화 J, 환전 (한일 >, 일한 <), 종료 Q

>

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

50000

Cancel OK

에리카(원화)

50000 환전 지출
잔액 = 200000

에리카(일화)

5000 환전 수입
잔액 = 20000

에리카(원화)

50000 환전 지출
잔액 = 200000

에리카(일화)

5000 환전 수입
잔액 = 20000

Input

+, -, K, J, >, <, Q 키 중 하나를 누르고 OK 단추를 누르세요.
수입 +, 지출 -, 한화 K, 일화 J, 환전 (한일 >, 일한 <), 종료 Q

<

Cancel OK

Input

금액을 입력하고 OK 단추를 누르세요.

1000

Cancel OK

에리카(원화)

10000 환전 수입
잔액 = 210000

에리카(일화)

1000 환전 지출
잔액 = 19000