9. ITERATOR & COMPOSITE PATTERN

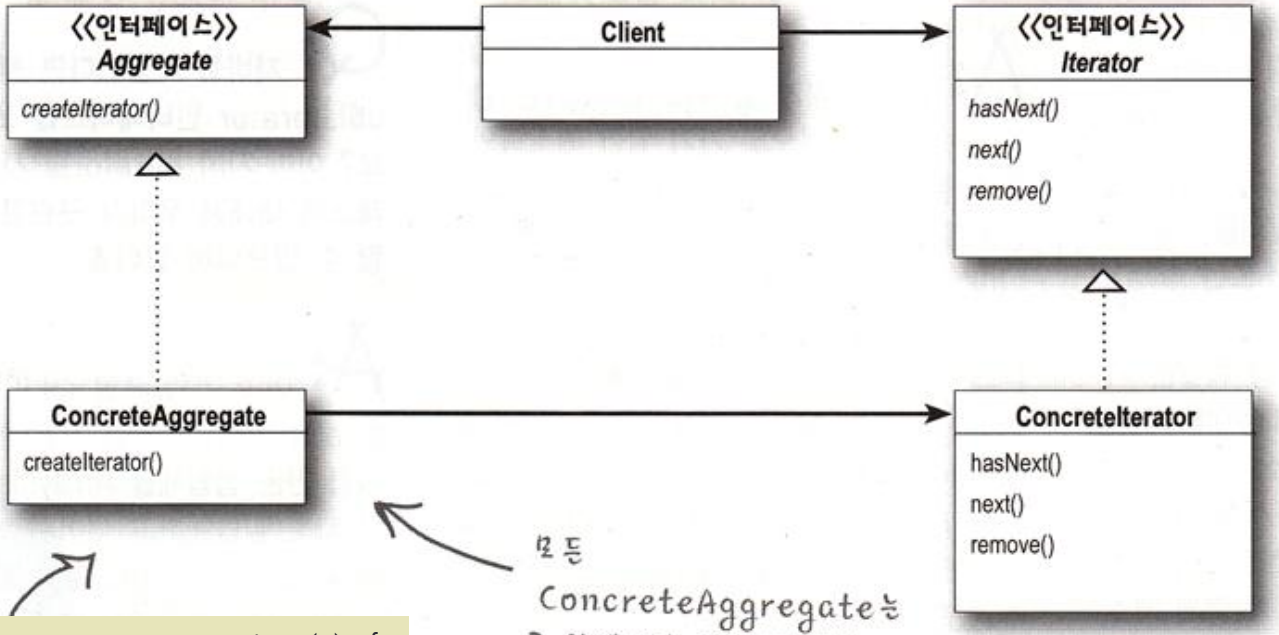
Iterator Pattern

- 컬렉션 구현 방법을 노출시키지 않으면서도 그 **집합체 안에 들어있는 모든 항목에 접근**할 수 있게 해 주는 방법을 제공
  - 집합체 내의 항목 구조를 모르더라도 모든 항목에 대한 반복 작업 수행 가능: 배열, 리스트, 이중 연결 리스트, DEQUE 등
  - 모든 항목에 일일이 접근하는 작업을 컬렉션 객체가 아닌 반복자 (Iterator) 객체에 맡기므로 간단해짐.

➔디자인 원칙: 클래스를 바꾸는 이유는 한 가지 뿐이어야 한다. (**Single Responsibility Principle**)
➔클래스의 역할이 두 개 이상 있으면 바뀔 수 있는 부분이 두 가지 이상이다.

```
interface java.lang.Iterable<T> {
    Iterator<T> iterator();
}
```

공통적인 인터페이스가 있으면 클라이언트 입장에서 매우 편리해집니다. 클라이언트와 객체 컬렉션의 구현이 분리될 수 있으니까요.



Iterator 인터페이스에서는 모든 반복자에서 구현해야 하는 인터페이스를 제공하며 컬렉션에 들어있는 원소들에 돌아가면서 접근할 수 있게 해 주는 메소드들을 제공합니다. 여기에서는 java.util.Iterator를 사용합니다. 자바에서 기본으로 제공하는 Iterator 인터페이스를 사용하고 싶지 않다면 직접 인터페이스를 만들어서 써도 됩니다.

```
interface java.util.List<E> implement Iterable<E> {
    public Iterator<E> iterator() {
        ...
    }
}
```

Iterator를 리턴하는 메소드를 구현합니다.

모든 ConcreteAggregate는 그 안에 있는 객체 컬렉션에 대해 돌아가면서 반복 작업을 처리할 수 있게 해 주는 ConcreteIterator의 인스턴스를 만들 수 있어야 합니다.

ConcreteIterator에서는 반복작업 중에 현재 위치를 관리하는 일을 맡고 있습니다.

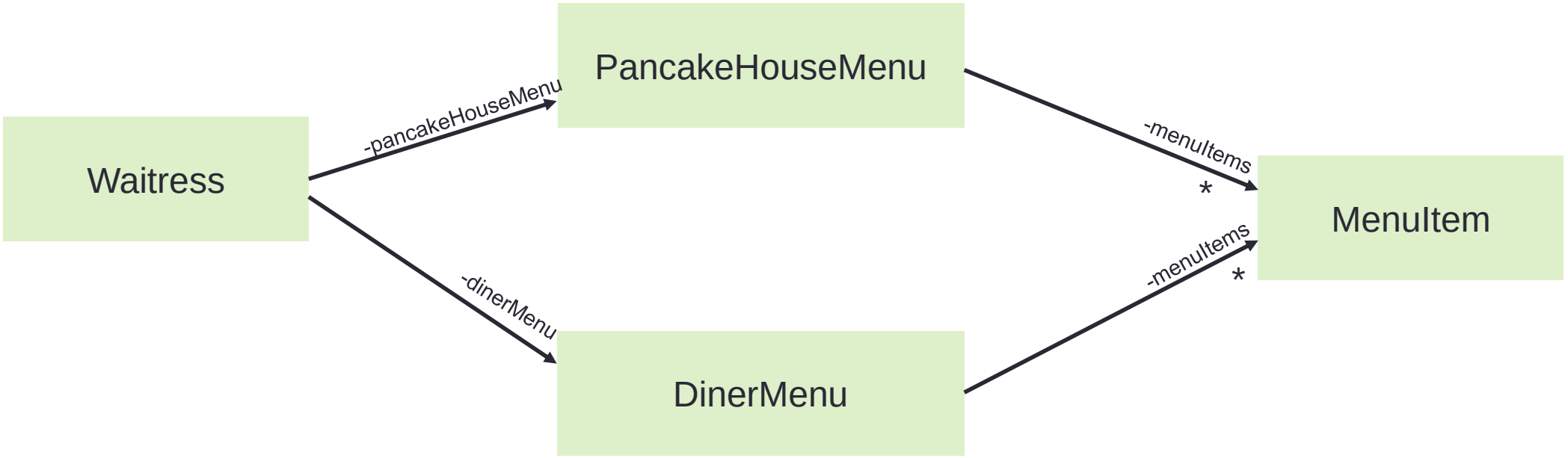
java.util.Iterator<E> 인터페이스

- 컬렉션 객체에 대한 반복적인 일을 담당
- An iterator over a collection
- A member of the Java Collections Framework
  (cf. http://download.oracle.com/javase/6/docs/api/)

Method Summary	
boolean	**hasNext**() Returns `true` if the iteration has more elements.
E	**next**() Returns the next element in the iteration.
void	**remove**() Removes from the underlying collection the last element returned by the iterator (optional operation).

first 모델링

- 구현에 기반하여 코딩
- Waitress 클래스가 PancakeHouseMenu와 DinerMenu 클래스를 직접 사용하여 주문 가능한 메뉴를 출력



MenuItem.java

```java
package cse.design_pattern.ch09.iterator.first;

public class MenuItem {

    private final String name;
    private final String description;
    private final boolean vegetarian;
    private final double price;

    public MenuItem(String name,
            String description,
            boolean vegetarian,
            double price) {
        this.name = name;
        this.description = description;
        this.vegetarian = vegetarian;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

    public double getPrice() {
        return price;
    }

    public boolean isVegetarian() {
        return vegetarian;
    }
}
```

PancakeHouseMenu.java

```
1   package cse.design_pattern.ch09.iterator.first;
2
3   import java.util.ArrayList;
4
5   public class PancakeHouseMenu {
6
7       private final ArrayList<MenuItem> menuItems;
8
9       public PancakeHouseMenu() {
10          menuItems = new ArrayList<>();
11
12          addItem("K&B's Pancake Breakfast",
13                  "Pancakes with scrambled eggs, and toast",
14                  true,
15                  2.99);
16
17          addItem("Regular Pancake Breakfast",
18                  "Pancakes with fried eggs, sausage",
19                  false,
20                  2.99);
```

```
22          addItem("Blueberry Pancakes",
23                  "Pancakes made with fresh blueberries, "
24                          + "and blueberry syrup",
25                  true,
26                  3.49);
27
28          addItem("Waffles",
29                  "Waffles, with your choice of blueberries or strawberries",
30                  true,
31                  3.59);
32      }
33
34      public final void addItem(String name, String description, boolean vegetarian, double price) {
35          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
36          menuItems.add(menuItem);
37      }
38
39      public ArrayList<MenuItem> getMenuItems() {
40          // return menuItems;  // return of Collection field
41          return new ArrayList<>(menuItems);
42      }
43      // 기타 메뉴 관련 메소드
44  }
```

DinerMenu.java

```
1   package cse.design_pattern.ch09.iterator.first;
2
3   public class DinerMenu {
4
5       private static final int MAX_ITEMS = 6;
6       private int numberOfItems = 0;
7       private final MenuItem[] menuItems;
8
9       public DinerMenu() {
10          menuItems = new MenuItem[MAX_ITEMS];
11
12          addItem("Vegetarian BLT",
13                  "(Fakin') Bacon with lettuce & tomato on whole wheat", true, 2.99);
14          addItem("BLT",
15                  "Bacon with lettuce & tomato on whole wheat", false, 2.99);
16          addItem("Soup of the day",
17                  "Soup of the day, with a side of potato salad", false, 3.29);
```

```
18          addItem("Hotdog",
19                  "A hot dog, with saurkraut, relish, onions, topped with cheese",
20                  false, 3.05);
21          addItem("Steamed Veggies and Brown Rice",
22                  "Steamed vegetables over brown rice", true, 3.99);
23          addItem("Pasta",
24                  "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
25                  true, 3.89);
26      }
27
28      public final void addItem(String name, String description, boolean vegetarian, double price) {
29          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
30          if (numberOfItems >= MAX_ITEMS) {
31              System.err.println("Sorry, menu is full!  Can't add item to menu");
32          } else {
33              menuItems[numberOfItems] = menuItem;
34              numberOfItems += 1;
35          }
36      }
37
38      public MenuItem[] getMenuItems() {
39          //return menuItems;  // return of Array Field
40          return menuItems.clone();
41      }
42      // 기타 메뉴 관련 메소드
43  }
```

Waitress.java

```java
package cse.design_pattern.ch09.iterator.first;

import java.util.List;

public class Waitress {

    private final PancakeHouseMenu pancakeHouseMenu;
    private final DinerMenu dinerMenu;

    public Waitress(PancakeHouseMenu pancakeHouseMenu, DinerMenu dinerMenu) {
        this.pancakeHouseMenu = pancakeHouseMenu;
        this.dinerMenu = dinerMenu;
    }
```

```
15      public void printMenu() {
16          List<MenuItem> breakfastItems = pancakeHouseMenu.getMenuItems();
17          System.out.println("메뉴₩n----₩n아침 메뉴");
18          for (int i = 0; i < breakfastItems.size(); i++) {
19              MenuItem m = breakfastItems.get(i);
20              printMenuItem(m);
21          }
22          System.out.println("₩n점심 메뉴");
23          MenuItem[] lunchItems = dinerMenu.getMenuItems();
            for (int i = 0; i < lunchItems.length; i++) {
25              MenuItem m = lunchItems[i];
26              printMenuItem(m);
27          }
28      }
29
30      public void printMenuItem(MenuItem menuItem) {
31          System.out.print(menuItem.getName() + ", ");
32          System.out.print(menuItem.getPrice() + " -- ");
33          System.out.println(menuItem.getDescription());
34      }
35  }
```

MenuTestDrive.java

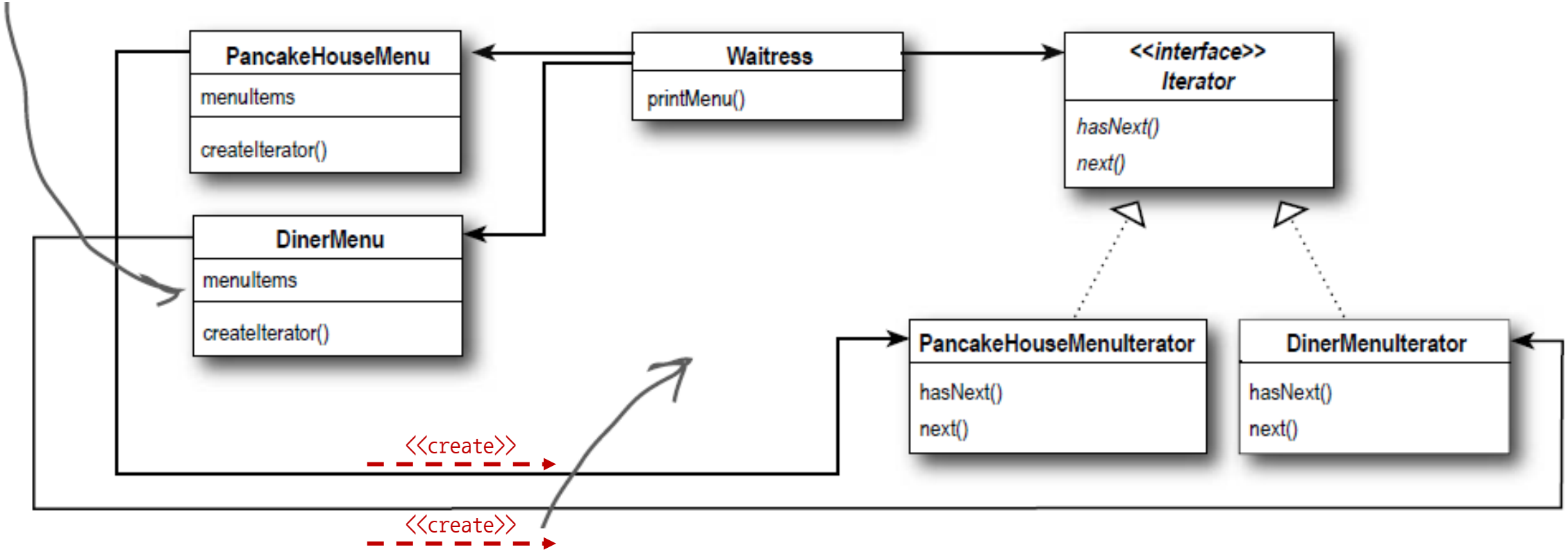
```
1  package cse.design_pattern.ch09.iterator.first;
2
3  public class MenuTestDrive {
4
5      public static void main(String args[]) {
6          PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
7          DinerMenu dinerMenu = new DinerMenu();
8
9          Waitress waitress = new Waitress(pancakeHouseMenu, dinerMenu);
10
11         waitress.printMenu();
12     }
13 }
```

실행 결과

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---
메뉴
----
아침 메뉴
K&B's Pancake Breakfast, 2.99 -- Pancakes with scrambled eggs, and toast
Regular Pancake Breakfast, 2.99 -- Pancakes with fried eggs, sausage
Blueberry Pancakes, 3.49 -- Pancakes made with fresh blueberries, and blueberry syrup
Waffles, 3.59 -- Waffles, with your choice of blueberries or strawberries

점심 메뉴
Vegetarian BLT, 2.99 -- (Fakin') Bacon with lettuce & tomato on whole wheat
BLT, 2.99 -- Bacon with lettuce & tomato on whole wheat
Soup of the day, 3.29 -- Soup of the day, with a side of potato salad
Hotdog, 3.05 -- A hot dog, with saurkraut, relish, onions, topped with cheese
Steamed Veggies and Brown Rice, 3.99 -- Steamed vegetables over brown rice
Pasta, 3.89 -- Spaghetti with Marinara Sauce, and a slice of sourdough bread
```

second 모델링

- (문제점) 접근 방식에 따라서 일하는 방식이 달라짐
  → (해결 방안) 이터레이터를 사용하여 각 메뉴 접근 방식을 통일



Iterator.java

```
1  package cse.design_pattern.ch09.iterator.second;
2
3  public interface Iterator<T> {
4
5      boolean hasNext();
6
7      T next();
8  }
```

← 지네릭(generic) 사용

PancakeHouseMenuIterator.java

```
1   package cse.design_pattern.ch09.iterator.second;
2
3   import java.util.ArrayList;
4   import cse.design_pattern.ch09.iterator.first.MenuItem;
5
6   public class PancakeHouseMenuIterator implements Iterator<MenuItem> {
7
8       private final ArrayList<MenuItem> items;
9       private int position = 0;
10
11      public PancakeHouseMenuIterator(ArrayList<MenuItem> items) {
12          this.items = items;
13      }
```

ArrayList는 이터레이터 반환하는 메서드가 있어 굳이 이렇게 구현할 필요 없음

PanCakeHouseMenu 클래스의 인스턴스 변수 menuItems를 알고 있어야 하므로 PanCakeHouseMenu에서 이터레이터 생성 시 생성자의 인자로 넘겨 줌.

```
15      @Override
        public MenuItem next() {
17          MenuItem item = items.get(position);
18          position += 1;
19          return item;
20      }
21
22      @Override
        public boolean hasNext() {
24          /*
25          if (position >= items.size()) {
26          return false;
27          } else {
28          return true;
29          }
30           */
31          return position < items.size();
32      }
33  }
```

DinerMenuIterator.java

```java
package cse.design_pattern.ch09.iterator.second;

import cse.design_pattern.ch09.iterator.first.MenuItem;

public class DinerMenuIterator implements Iterator<MenuItem> {

    private final MenuItem[] items;
    private int position = 0;

    public DinerMenuIterator(MenuItem[] items) {
        this.items = items;
    }
```

DinerMenu 클래스의
인스턴스 변수 menuItems를
알고 있어야 하므로
DinerMenu에서 이터레이터
생성 시 생성자의 인자로 넘겨 줌.

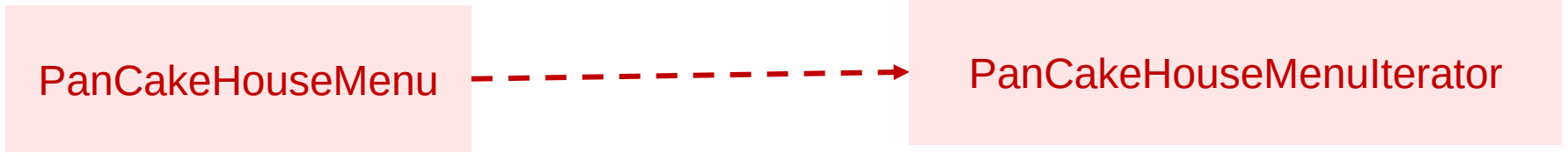
```
14      @Override
        public MenuItem next() {
16          MenuItem menuItem = items[position];
17          position += 1;
18          return menuItem;
19      }
20
21      @Override
        public boolean hasNext() {
23          /*
24          if (position >= items.length || items[position] == null) {
25          return false;
26          } else {
27          return true;
28          }
29           */
30          return !(position >= items.length || items[position] == null);
31      }
32  }
```

PancakeHouseMenu.java

```java
package cse.design_pattern.ch09.iterator.second;

import java.util.ArrayList;
import cse.design_pattern.ch09.iterator.first.MenuItem;

public class PancakeHouseMenu {

    private final ArrayList<MenuItem> menuItems;

    public PancakeHouseMenu() {
        menuItems = new ArrayList<>();

        addItem("K&B's Pancake Breakfast",
                "Pancakes with scrambled eggs, and toast",
                true,
                2.99);
```

```
18          addItem("Regular Pancake Breakfast",
19                  "Pancakes with fried eggs, sausage",
20                  false,
21                  2.99);
22
23          addItem("Blueberry Pancakes",
24                  "Pancakes made with fresh blueberries, and blueberry syrup",
25                  true,
26                  3.49);
27
28          addItem("Waffles",
29                  "Waffles, with your choice of blueberries or strawberries",
30                  true,
31                  3.59);
32      }
33
34      public final void addItem(String name, String description,
35              boolean vegetarian, double price) {
36          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
37          menuItems.add(menuItem);
38      }
```

```
40      public Iterator<MenuItem> createIterator() {
41          return new PancakeHouseMenuIterator(menuItems);
42      }
43
44  }
```



DinerMenu.java

```java
package cse.design_pattern.ch09.iterator.second;

import cse.design_pattern.ch09.iterator.first.MenuItem;

public class DinerMenu {

    private static final int MAX_ITEMS = 6;
    private int numberOfItems = 0;
    private final MenuItem[] menuItems;

    public DinerMenu() {
        menuItems = new MenuItem[MAX_ITEMS];

        addItem("Vegetarian BLT",
                "(Fakin') Bacon with lettuce & tomato on whole wheat", true, 2.99);
        addItem("BLT",
                "Bacon with lettuce & tomato on whole wheat", false, 2.99);
```

```
18          addItem("Soup of the day",
19                  "Soup of the day, with a side of potato salad", false, 3.29);
20          addItem("Hotdog",
21                  "A hot dog, with saurkraut, relish, onions, topped with cheese",
22                  false, 3.05);
23          addItem("Steamed Veggies and Brown Rice",
24                  "Steamed vegetables over brown rice", true, 3.99);
25          addItem("Pasta",
26                  "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
27                  true, 3.89);
28      }
29
30      public final void addItem(String name, String description,
31              boolean vegetarian, double price) {
32          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
33          if (numberOfItems >= MAX_ITEMS) {
34              System.err.println("Sorry, menu is full!  Can't add item to menu");
35          } else {
36              menuItems[numberOfItems] = menuItem;
37              numberOfItems += 1;
38          }
39      }
40
41      public Iterator<MenuItem> createIterator() {
42          return new DinerMenuIterator(menuItems);
43      }
44
45  }
```

DinerMenu - - - - → DinerMenuIterator

Waitress.java

- PancakeHouseMenu와 DinerMenu 구상 클래스와 연관 관계
- 두 구상 클래스의 createIterator() 메소드에 의하여 이터레이터 객체를 반환 받아 일하는 방식은 동일하게 됨.

```
1   package cse.design_pattern.ch09.iterator.second;
2
3   import cse.design_pattern.ch09.iterator.first.MenuItem;
4
5   public class Waitress {
6
7       private final PancakeHouseMenu pancakeHouseMenu;
8       private final DinerMenu dinerMenu;
9
10      public Waitress(PancakeHouseMenu pancakeHouseMenu, DinerMenu dinerMenu) {
11          this.pancakeHouseMenu = pancakeHouseMenu;
12          this.dinerMenu = dinerMenu;
13      }
```

```
15      public void printMenu() {
16          Iterator<MenuItem> pancakeIterator = pancakeHouseMenu.createIterator();
17          Iterator<MenuItem> dinerIterator = dinerMenu.createIterator();
18          System.out.println("메뉴₩n----₩n아침 메뉴");
19          printMenu(pancakeIterator);
20          System.out.println("₩n점심 메뉴");
21          printMenu(dinerIterator);
22      }
23
24      public void printMenu(Iterator<MenuItem> iterator) {
25          while (iterator.hasNext()) {
26              MenuItem menuItem = iterator.next();
27              System.out.print(menuItem.getName() + ", ");
28              System.out.print(menuItem.getPrice() + " -- ");
29              System.out.println(menuItem.getDescription());
30          }
31      }
32  }
```

printMenu()는 오버로딩된 메서드임. 메서드 매개변수가 달라서 구분 가능함.

MenuTestDrive.java

```
1   package cse.design_pattern.ch09.iterator.second;
2
3   public class MenuTestDrive {
4
5       public static void main(String args[]) {
6           PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
7           DinerMenu dinerMenu = new DinerMenu();
8
9           Waitress waitress = new Waitress(pancakeHouseMenu, dinerMenu);
10
11          waitress.printMenu();
12      }
13  }
```

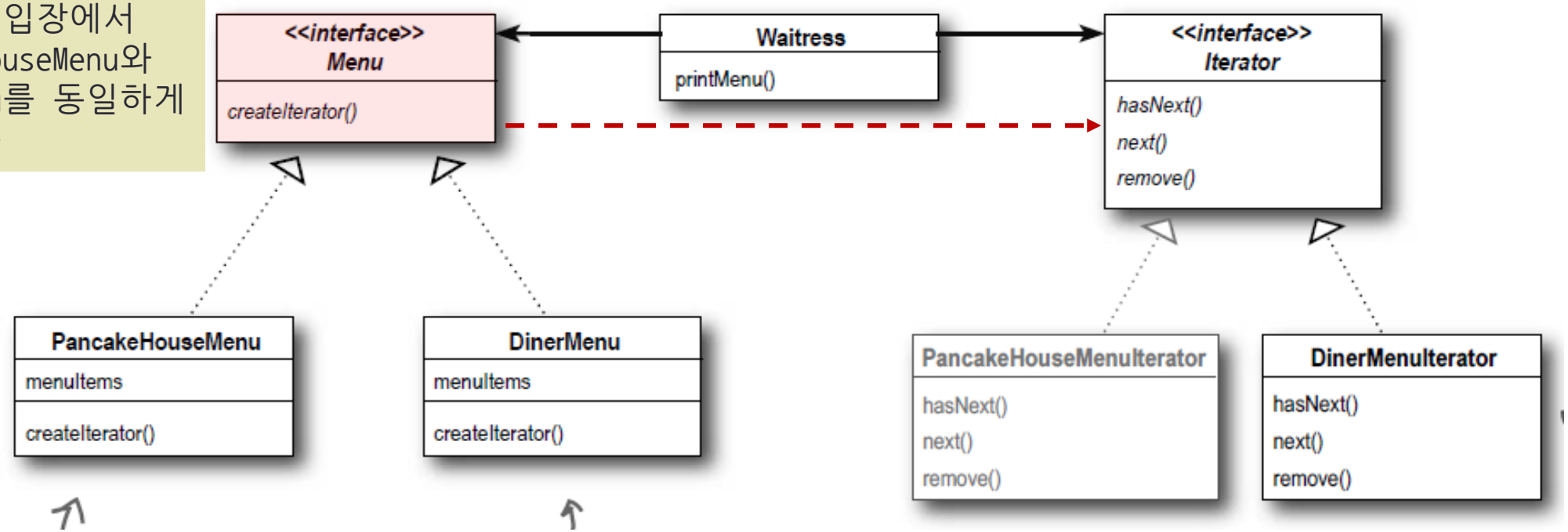
실행 결과

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---
메뉴
----
아침 메뉴
K&B's Pancake Breakfast, 2.99 -- Pancakes with scrambled eggs, and toast
Regular Pancake Breakfast, 2.99 -- Pancakes with fried eggs, sausage
Blueberry Pancakes, 3.49 -- Pancakes made with fresh blueberries, and blueberry syrup
Waffles, 3.59 -- Waffles, with your choice of blueberries or strawberries

점심 메뉴
Vegetarian BLT, 2.99 -- (Fakin') Bacon with lettuce & tomato on whole wheat
BLT, 2.99 -- Bacon with lettuce & tomato on whole wheat
Soup of the day, 3.29 -- Soup of the day, with a side of potato salad
Hotdog, 3.05 -- A hot dog, with saurkraut, relish, onions, topped with cheese
Steamed Veggies and Brown Rice, 3.99 -- Steamed vegetables over brown rice
Pasta, 3.89 -- Spaghetti with Marinara Sauce, and a slice of sourdough bread
```

third 모델링

- 문제점: Waitress 클래스는 PancakeHouseMenu와 DinerMenu 클래스의 구현을 직접 사용
- 해결 방안:
  - PancakeHouseMenu와 DinerMenu 클래스의 createIterator() 메소드는 공통으로 사용
  - Menu 인터페이스을 만들어 대표 타입으로 사용
  - java.util.Iterator 적용

Waitress 입장에서 PancakeHouseMenu와 DinerMenu를 동일하게 처리 가능



Menu.java

- p.372

```
1   package cse.design_pattern.ch09.iterator.third;
2
3   import cse.design_pattern.ch09.iterator.first.MenuItem;
4   import java.util.Iterator;
5
6
7   public interface Menu {
8
9       public Iterator<MenuItem> createIterator();
10  }
```

PancakeHouseMenu.java

```
1   package cse.design_pattern.ch09.iterator.third;
2
3   import java.util.ArrayList;
4   import java.util.Iterator;
5   import cse.design_pattern.ch09.iterator.first.MenuItem;
6
7   public class PancakeHouseMenu implements Menu {
8
9       private final ArrayList<MenuItem> menuItems;
10
11      public PancakeHouseMenu() {
12          menuItems = new ArrayList<>();
13
14          addItem("K&B's Pancake Breakfast",
15                  "Pancakes with scrambled eggs, and toast",
16                  true,
17                  2.99);
18
19          addItem("Regular Pancake Breakfast",
20                  "Pancakes with fried eggs, sausage",
21                  false,
22                  2.99);
```

```
23          addItem("Blueberry Pancakes",
24                  "Pancakes made with fresh blueberries, and blueberry syrup",
25                  true,
26                  3.49);
27
28          addItem("Waffles",
29                  "Waffles, with your choice of blueberries or strawberries",
30                  true,
31                  3.59);
32      }
33
34      public final void addItem(String name, String description,
35              boolean vegetarian, double price) {
36          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
37          menuItems.add(menuItem);
38      }
39
40      @Override
        public Iterator<MenuItem> createIterator() {
42          return menuItems.iterator();
43      }
44
45  }
```

ArrayList에 있는 iterator() 메서드 사용함.

DinerMenu.java

```java
package cse.design_pattern.ch09.iterator.third;

import cse.design_pattern.ch09.iterator.first.MenuItem;
import java.util.Iterator;


public class DinerMenu implements Menu {

    private static final int MAX_ITEMS = 6;
    private int numberOfItems = 0;
    private final MenuItem[] menuItems;

    public DinerMenu() {
        menuItems = new MenuItem[MAX_ITEMS];

        addItem("Vegetarian BLT",
                "(Fakin') Bacon with lettuce & tomato on whole wheat", true, 2.99);
        addItem("BLT",
                "Bacon with lettuce & tomato on whole wheat", false, 2.99);
        addItem("Soup of the day",
                "Soup of the day, with a side of potato salad", false, 3.29);
```

```
22          addItem("Hotdog",
23                  "A hot dog, with saurkraut, relish, onions, topped with cheese",
24                  false, 3.05);
25          addItem("Steamed Veggies and Brown Rice",
26                  "Steamed vegetables over brown rice", true, 3.99);
27          addItem("Pasta",
28                  "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
29                  true, 3.89);
30      }
31
32      public final void addItem(String name, String description,
33              boolean vegetarian, double price) {
34          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
35          if (numberOfItems >= MAX_ITEMS) {
36              System.err.println("Sorry, menu is full!  Can't add item to menu");
37          } else {
38              menuItems[numberOfItems] = menuItem;
39              numberOfItems += 1;
40          }
41      }
```

```
43      @Override
        public Iterator<MenuItem> createIterator() {
45          return new DinerMenuIterator(menuItems);
46      }
47
48  }
```

DinerMenuIterator.java

```java
package cse.design_pattern.ch09.iterator.third;

import java.util.Iterator;
import cse.design_pattern.ch09.iterator.first.MenuItem;

public class DinerMenuIterator implements Iterator<MenuItem> {

    private final MenuItem[] items;
    private int position = 0;

    public DinerMenuIterator(MenuItem[] items) {
        this.items = items;
    }
```

```
15      @Override
        public MenuItem next() {
17          MenuItem menuItem = items[position];
18          position += 1;
19          return menuItem;
20      }
21
22      @Override
        public boolean hasNext() {
24          return !(position >= items.length || items[position] == null);
25      }
26
27      // remove()는 default method이므로 재정의하지 않아도 됨.
28  }
```

Waitress.java

```java
package cse.design_pattern.ch09.iterator.third;

import java.util.Iterator;
import cse.design_pattern.ch09.iterator.first.MenuItem;

public class Waitress {

    private final Menu pancakeHouseMenu;
    private final Menu dinerMenu;

    public Waitress(Menu pancakeHouseMenu, Menu dinerMenu) {
        this.pancakeHouseMenu = pancakeHouseMenu;
        this.dinerMenu = dinerMenu;
    }
```

```
16      public void printMenu() {
17          Iterator<MenuItem> pancakeIterator = pancakeHouseMenu.createIterator();
18          Iterator<MenuItem> dinerIterator = dinerMenu.createIterator();
19          System.out.println("메뉴₩n----₩n아침 메뉴");
20          printMenu(pancakeIterator);
21          System.out.println("₩n점심 메뉴");
22          printMenu(dinerIterator);
23      }
24
25      public void printMenu(Iterator<MenuItem> iterator) {
26          while (iterator.hasNext()) {
27              MenuItem menuItem = iterator.next();
28              System.out.print(menuItem.getName() + ", ");
29              System.out.print(menuItem.getPrice() + " -- ");
30              System.out.println(menuItem.getDescription());
31          }
32      }
33  }
```

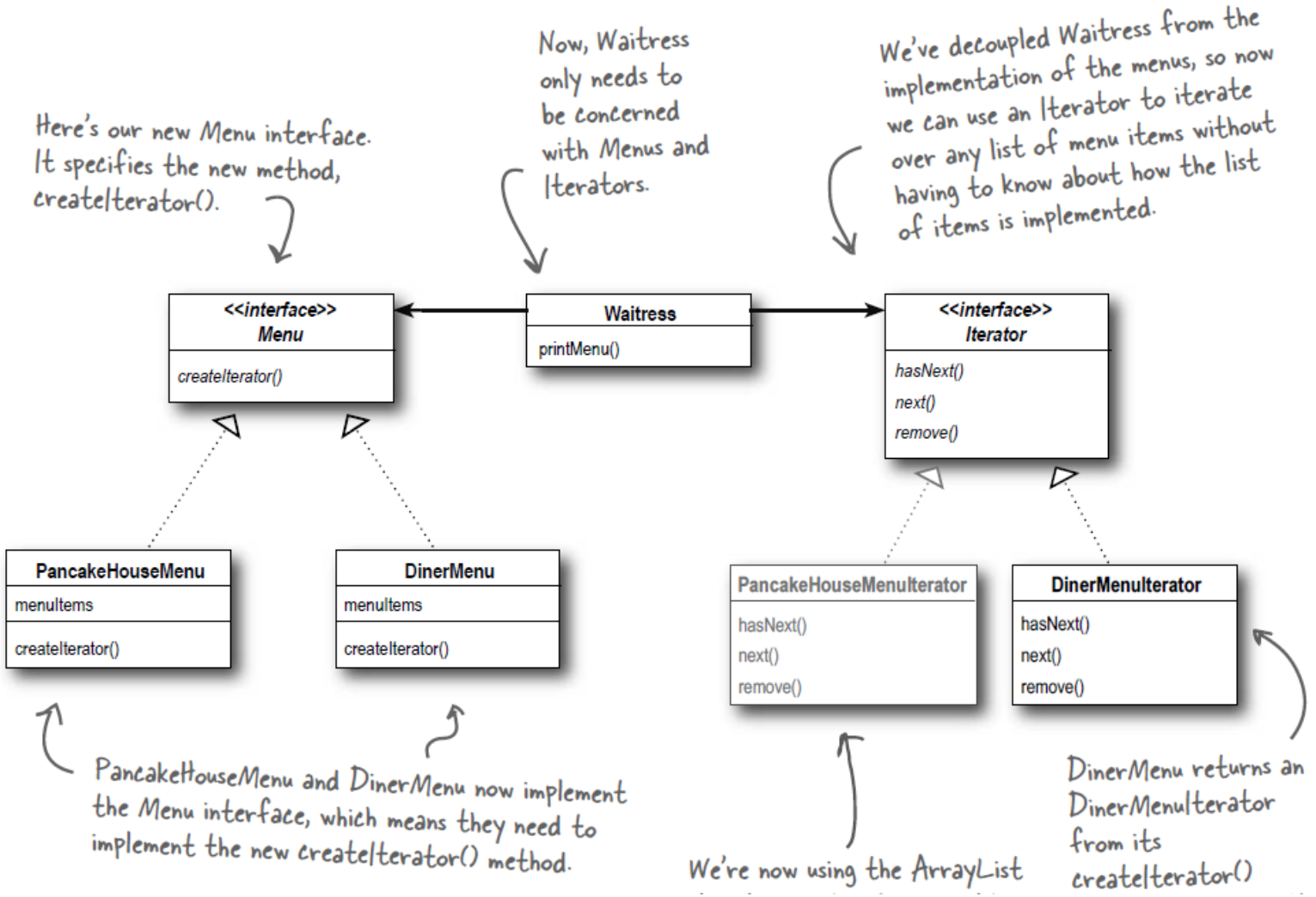
MenuTestDrive.java

```
1   package cse.design_pattern.ch09.iterator.third;
2
3   public class MenuTestDrive {
4
5       public static void main(String args[]) {
6           PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
7           DinerMenu dinerMenu = new DinerMenu();
8
9           Waitress waitress = new Waitress(pancakeHouseMenu, dinerMenu);
10
11          waitress.printMenu();
12      }
13  }
```

실행 결과

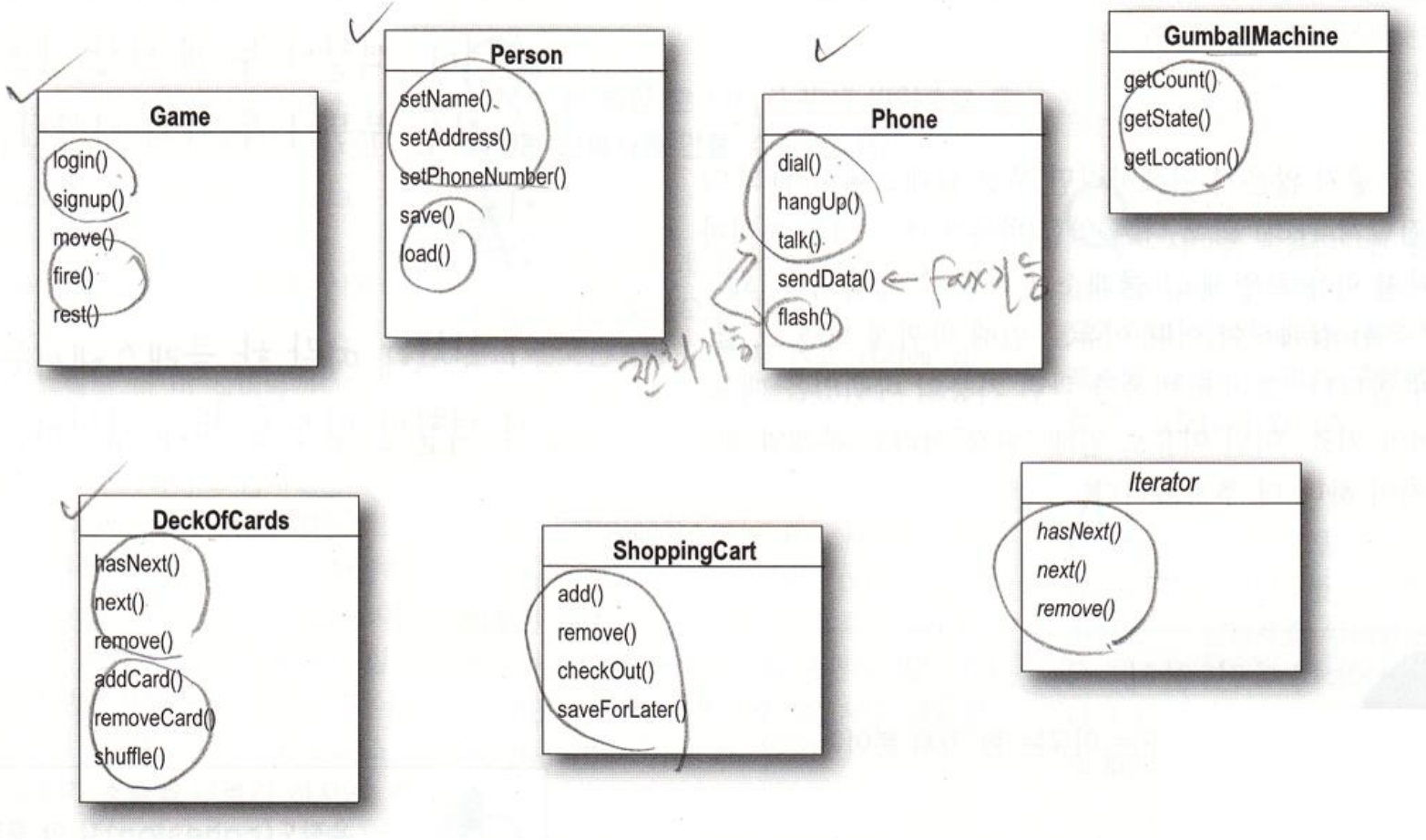
```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---
메뉴
----
아침 메뉴
K&B's Pancake Breakfast, 2.99 -- Pancakes with scrambled eggs, and toast
Regular Pancake Breakfast, 2.99 -- Pancakes with fried eggs, sausage
Blueberry Pancakes, 3.49 -- Pancakes made with fresh blueberries, and blueberry syrup
Waffles, 3.59 -- Waffles, with your choice of blueberries or strawberries

점심 메뉴
Vegetarian BLT, 2.99 -- (Fakin') Bacon with lettuce & tomato on whole wheat
BLT, 2.99 -- Bacon with lettuce & tomato on whole wheat
Soup of the day, 3.29 -- Soup of the day, with a side of potato salad
Hotdog, 3.05 -- A hot dog, with saurkraut, relish, onions, topped with cheese
Steamed Veggies and Brown Rice, 3.99 -- Steamed vegetables over brown rice
Pasta, 3.89 -- Spaghetti with Marinara Sauce, and a slice of sourdough bread
```

이터레이터 패턴 적용 완성

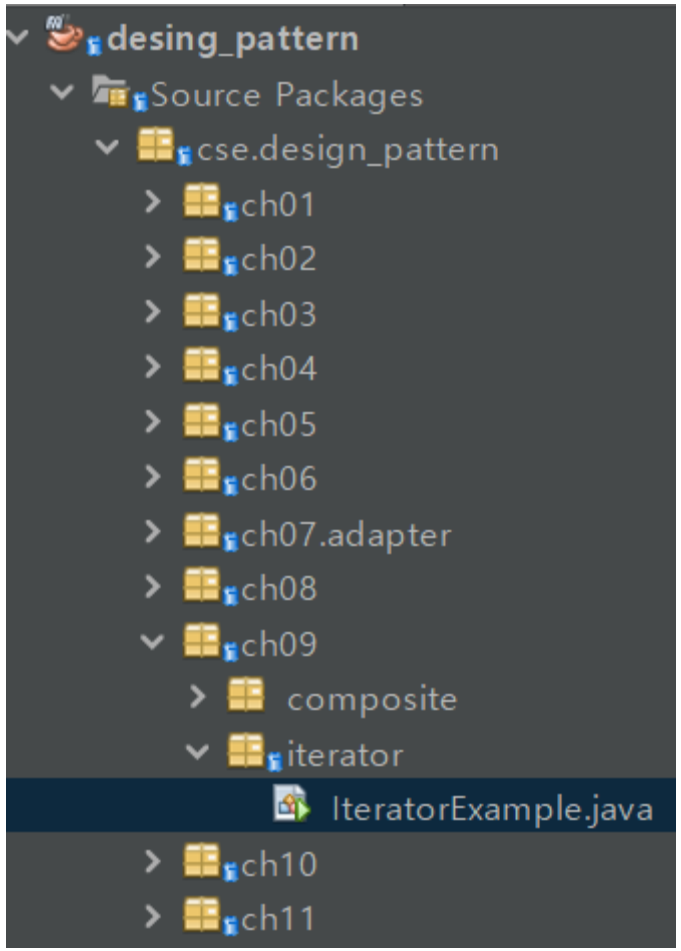


클래스의 역할 찾기

다음 클래스들을 살펴보고 여러 역할을 맡고 있는 클래스를 찾아 봅시다. p.378



IteratorExample 프로젝트



IteratorExample.java

```
6   package cse.design_pattern.ch09.iterator.example;
7
8   import java.util.ArrayList;
9   import java.util.Comparator;
0   import java.util.Iterator;
1   import java.util.List;
2
3   /**
7   public class IteratorExample {
8       private static List<String> names;
9
0       private static void initialize() {
1           names = new ArrayList<>();
2           names.add("Gildong");
3           names.add("Woochi");
4           names.add("Chulsoo");
5       }
```

```
27      public static void run01() {
28          System.out.println("===========");
29          for (int i=0; i<names.size(); i++) {
30              System.out.printf("Hello, %s%n", names.get(i));
31          }
32          System.out.println("===========");
33          for (String name: names) {
34              System.out.printf("Hello, %s%n", name);
35          }
36      }
37
38      public static void run02() {
39          System.out.println("===========");
40          Iterator<String> it = names.iterator();
41          while (it.hasNext()) {
42              String name = it.next();
43              System.out.printf("Hello, %s%n", name);
44          }
45      }
```

```
47      public static void run03() {
48          System.out.println("===========");
49          names.forEach((e) -> {
50                  System.out.printf("Hello, %s%n", e);
51          });
52      }
53
54      public static void doSort() {
55          System.out.println("===== doSort() =====");
56          names.sort(new MyComparator());
57          names.forEach((e) -> {
58                  System.out.printf("Hello, %s%n", e);
59          });
60      }
```

```
65      public static void main(String[] args) {
66          initialize();
67          run01();
68          run02();
69          run03();
70          doSort();
71      }
72  }
73
74
75  class MyComparator implements Comparator<String> {
76
77      @Override
78      public int compare(String o1, String o2) {
79          return o1.compareTo(o2);
80      }
81
82  }
```

실행 결과

```
==========
Hello, Gildong
Hello, Woochi
Hello, Chulsoo
==========
Hello, Gildong
Hello, Woochi
Hello, Chulsoo
==========
Hello, Gildong
Hello, Woochi
Hello, Chulsoo
==========
Hello, Gildong
Hello, Woochi
Hello, Chulsoo
===== doSort() =====
Hello, Chulsoo
Hello, Gildong
Hello, Woochi
```

C++ 버전 1/2

cpp_iterator_example.cpp

```cpp
#include <iostream>
#include <string>
#include <list>


int main()
{
    std::list<std::string> names;

    std::string n1("Gildong");
    std::string n2("Woochi");
    std::string n3("Chulsoo");

    names.push_back(n1);
    names.push_back(n2);
    names.push_back(n3);

    for (std::list<std::string>::iterator it=names.begin(); it!=names.end(); it++) {
        std::cout << "Hello, " << (*it).c_str() << std::endl;
    }
}
```

C++ 버전 2/2

cpp_iterator_example2.cpp

```cpp
#include <iostream>
#include <string>
#include <list>

using namespace std;

int main()
{
    list<string> names;

    string n1("Gildong");
    string n2("Woochi");
    string n3("Chulsoo");

    names.push_back(n1);
    names.push_back(n2);
    names.push_back(n3);

    for (list<string>::iterator it=names.begin(); it!=names.end(); it++) {
        cout << "Hello, " << (*it).c_str() << endl;
    }
}
```

cafe1 모델링

- 새 메뉴인 카페 메뉴 CafeMenu 추가
  - Menu 인터페이스를 구현하여 기존의 PancakeHouseMenu와 DinerMenu 클래스와 동일하게 사용할 수 있도록 하려고 함.
  - **CafeMenu 클래스는 HashMap<String, MenuItem>을 사용하여 구현**

- 기존 시스템에 새로운 기능을 추가하거나 삭제 시 쉽게 할 수 있어야 좋은 구조!

CafeMenu.java

```
1   package cse.design_pattern.ch09.iterator.cafe1;
2
3   import cse.design_pattern.ch09.iterator.third.Menu;
4   import cse.design_pattern.ch09.iterator.first.MenuItem;
5   import java.util.*;
6
7   public class CafeMenu implements Menu {
8
9       private final HashMap<String, MenuItem> menuItems = new HashMap<>();
10
11      public CafeMenu() {
12          addItem("Veggie Burger and Air Fries",
13                  "Veggie burger on a whole wheat bun, lettuce, tomato, and fries",
14                  true, 3.99);
15          addItem("Soup of the day",
16                  "A cup of the soup of the day, with a side salad",
17                  false, 3.69);
18          addItem("Burrito",
19                  "A large burrito, with whole pinto beans, salsa, guacamole",
20                  true, 4.29);
21      }
```

```
23      public final void addItem(String name, String description,
24              boolean vegetarian, double price) {
25          MenuItem menuItem = new MenuItem(name, description, vegetarian, price);
26          menuItems.put(menuItem.getName(), menuItem);
27      }
28
29      public Map<String, MenuItem> getItems() {
30          return Collections.unmodifiableMap(menuItems);
31      }
32
33      @Override
        public Iterator<MenuItem> createIterator() {
35          return menuItems.values().iterator();
36      }
37  }
```

```
Collections.unmodifiableList(…)
Collections.unmodifiableSet(…)
…
```

→ 컬렉션 수정 시 **UnsupportedOperationException** 발생

Waitress.java

```java
package cse.design_pattern.ch09.iterator.cafe1;

import cse.design_pattern.ch09.iterator.first.MenuItem;
import cse.design_pattern.ch09.iterator.third.*;
import java.util.Iterator;

public class Waitress {

    private final Menu pancakeHouseMenu;
    private final Menu dinerMenu;
    private final Menu cafeMenu;

    public Waitress(Menu pancakeHouseMenu, Menu dinerMenu, Menu cafeMenu) {
        this.pancakeHouseMenu = pancakeHouseMenu;
        this.dinerMenu = dinerMenu;
        this.cafeMenu = cafeMenu; // 추가
    }
```

```
19      public void printMenu() {
20          Iterator<MenuItem> pancakeIterator = pancakeHouseMenu.createIterator();
21          Iterator<MenuItem> dinerIterator = dinerMenu.createIterator();
22          Iterator<MenuItem> cafeIterator = cafeMenu.createIterator(); // 추가
23
24          System.out.println("메뉴₩n----₩n아침 메뉴");
25          printMenu(pancakeIterator);
26          System.out.println("₩n점심 메뉴");
27          printMenu(dinerIterator);
28          // 추가
29          System.out.println("₩n저녁 메뉴");
30          printMenu(cafeIterator);
31      }
32
33      public void printMenu(Iterator<MenuItem> iterator) {
34          while (iterator.hasNext()) {
35              MenuItem menuItem = iterator.next();
36              System.out.print(menuItem.getName() + ", ");
37              System.out.print(menuItem.getPrice() + " -- ");
38              System.out.println(menuItem.getDescription());
39          }
40      }
41  }
```

MenuTestDrive.java

```java
package cse.design_pattern.ch09.iterator.cafe1;

import cse.design_pattern.ch09.iterator.third.*;

public class MenuTestDrive {

    public static void main(String args[]) {
        PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
        DinerMenu dinerMenu = new DinerMenu();
        CafeMenu cafeMenu = new CafeMenu();

        Waitress waitress = new Waitress(pancakeHouseMenu, dinerMenu, cafeMenu);

        waitress.printMenu();
    }
}
```

실행 결과

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---
메뉴
----
아침 메뉴
K&B's Pancake Breakfast, 2.99 -- Pancakes with scrambled eggs, and toast
Regular Pancake Breakfast, 2.99 -- Pancakes with fried eggs, sausage
Blueberry Pancakes, 3.49 -- Pancakes made with fresh blueberries, and blueberry syrup
Waffles, 3.59 -- Waffles, with your choice of blueberries or strawberries

점심 메뉴
Vegetarian BLT, 2.99 -- (Fakin') Bacon with lettuce & tomato on whole wheat
BLT, 2.99 -- Bacon with lettuce & tomato on whole wheat
Soup of the day, 3.29 -- Soup of the day, with a side of potato salad
Hotdog, 3.05 -- A hot dog, with saurkraut, relish, onions, topped with cheese
Steamed Veggies and Brown Rice, 3.99 -- Steamed vegetables over brown rice
Pasta, 3.89 -- Spaghetti with Marinara Sauce, and a slice of sourdough bread

저녁 메뉴
Soup of the day, 3.69 -- A cup of the soup of the day, with a side salad
Veggie Burger and Air Fries, 3.99 -- Veggie burger on a whole wheat bun, lettuce, tomato, and fries
Burrito, 4.29 -- A large burrito, with whole pinto beans, salsa, guacamole
```

cafe2 모델링

- 문제점
  - **새 메뉴 추가 시 Waitress 클래스의 printMenu()에 코드 추가 필요 → OCP 위배**

- 해결 방안
  - **메뉴를 리스트로 관리**

Waitress.java

```
1   package cse.design_pattern.ch09.iterator.cafe2;
2
3   import cse.design_pattern.ch09.iterator.cafe1.CafeMenu;
4   import cse.design_pattern.ch09.iterator.first.MenuItem;
5   import cse.design_pattern.ch09.iterator.third.*;
6   import java.util.Iterator;
7   import java.util.List;
8
9   public class Waitress {
10
11      private final List<Menu> menus;
12
13      public Waitress(List<Menu> menus) {
14          this.menus = menus;
15      }
```

```
17      public void printMenu() {
18          Iterator<Menu> menuIterator =menus.iterator();
19          while (menuIterator.hasNext()) {
20              Menu menu = menuIterator.next();
21              printMenuName(menu);  // 추가
22              printMenu(menu.createIterator());
23          }
24      }
```

```
26      public void printMenu(Iterator<MenuItem> iterator) {
27          while (iterator.hasNext()) {
28              MenuItem menuItem = iterator.next();
29              System.out.print(menuItem.getName() + ", ");
30              System.out.print(menuItem.getPrice() + " -- ");
31              System.out.println(menuItem.getDescription());
32          }
33          //System.out.println();
34      }
35
36      private void printMenuName(Menu menu) {  // 추가
37          if (menu instanceof PancakeHouseMenu) {
38              System.out.println("메뉴₩n----₩n아침 메뉴");
39          } else if (menu instanceof DinerMenu) {
40              System.out.println("₩n점심 메뉴");
41          } else if (menu instanceof CafeMenu) {
42              System.out.println("₩n저녁 메뉴");
43          } else {
44              System.out.println("₩n모르는 메뉴");
45          }
46      }
47  }
```

MenuTestDrive.java

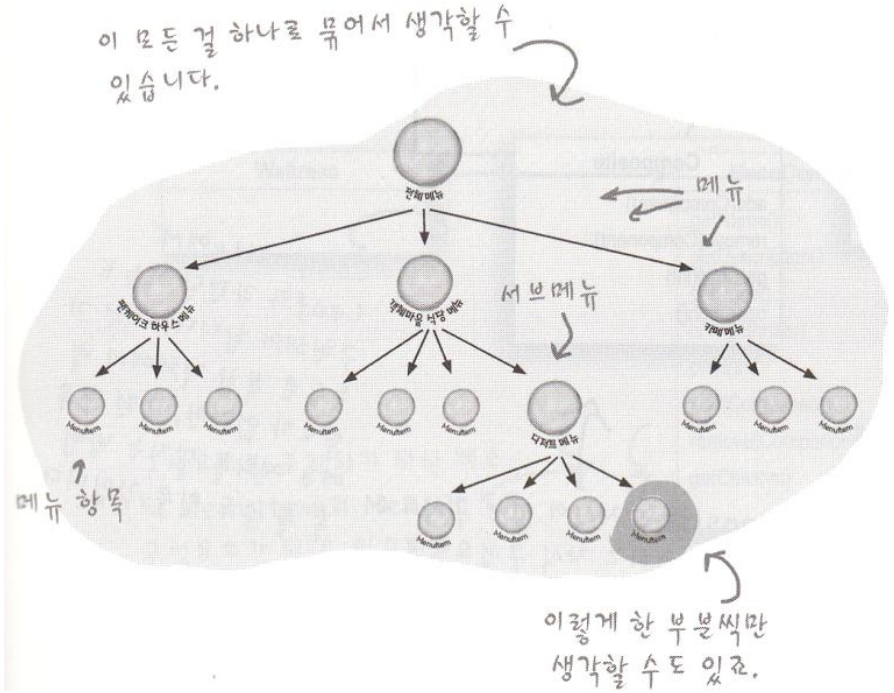
```
1   package cse.design_pattern.ch09.iterator.cafe2;
2
3   import cse.design_pattern.ch09.iterator.cafe1.*;
4   import cse.design_pattern.ch09.iterator.third.*;
5   import java.util.ArrayList;
6   import java.util.List;
7
8
9   public class MenuTestDrive {
10
11      public static void main(String args[]) {
12          List<Menu> menus = new ArrayList<>();
13          menus.add(new PancakeHouseMenu());
14          menus.add(new DinerMenu());
15          menus.add(new CafeMenu());
16
17          Waitress waitress = new Waitress(menus);
18          waitress.printMenu();
19      }
20  }
```

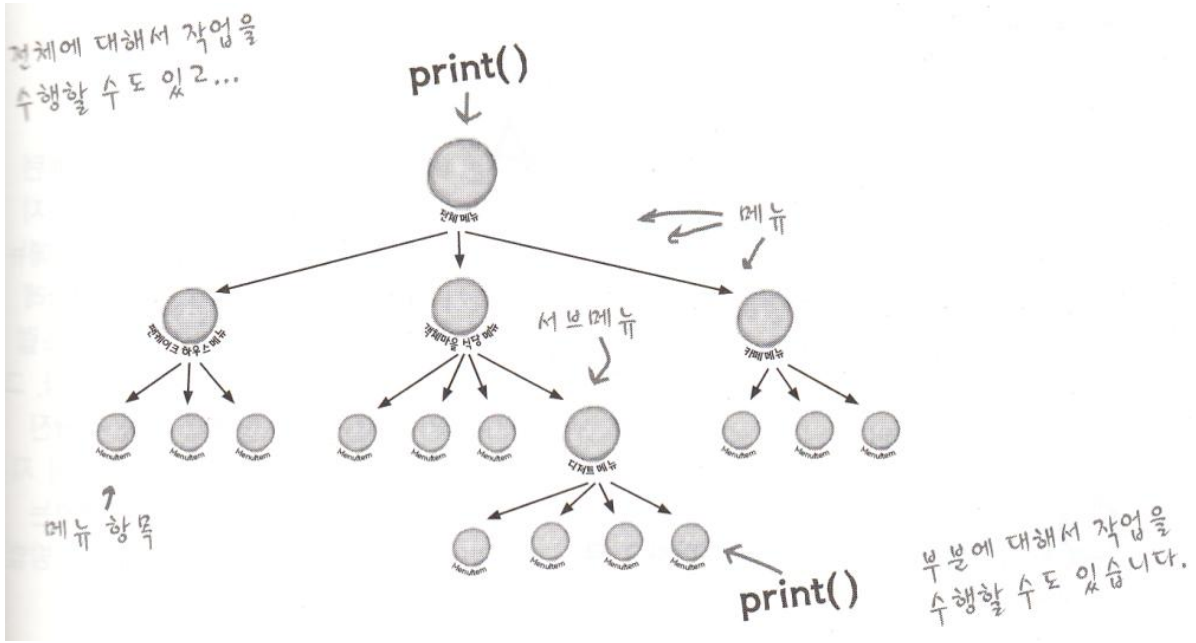
실행 결과

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---
메뉴
----
아침 메뉴
K&B's Pancake Breakfast, 2.99 -- Pancakes with scrambled eggs, and toast
Regular Pancake Breakfast, 2.99 -- Pancakes with fried eggs, sausage
Blueberry Pancakes, 3.49 -- Pancakes made with fresh blueberries, and blueberry syrup
Waffles, 3.59 -- Waffles, with your choice of blueberries or strawberries

점심 메뉴
Vegetarian BLT, 2.99 -- (Fakin') Bacon with lettuce & tomato on whole wheat
BLT, 2.99 -- Bacon with lettuce & tomato on whole wheat
Soup of the day, 3.29 -- Soup of the day, with a side of potato salad
Hotdog, 3.05 -- A hot dog, with saurkraut, relish, onions, topped with cheese
Steamed Veggies and Brown Rice, 3.99 -- Steamed vegetables over brown rice
Pasta, 3.89 -- Spaghetti with Marinara Sauce, and a slice of sourdough bread

저녁 메뉴
Soup of the day, 3.69 -- A cup of the soup of the day, with a side salad
Veggie Burger and Air Fries, 3.99 -- Veggie burger on a whole wheat bun, lettuce, tomato, and fries
Burrito, 4.29 -- A large burrito, with whole pinto beans, salsa, guacamole
```

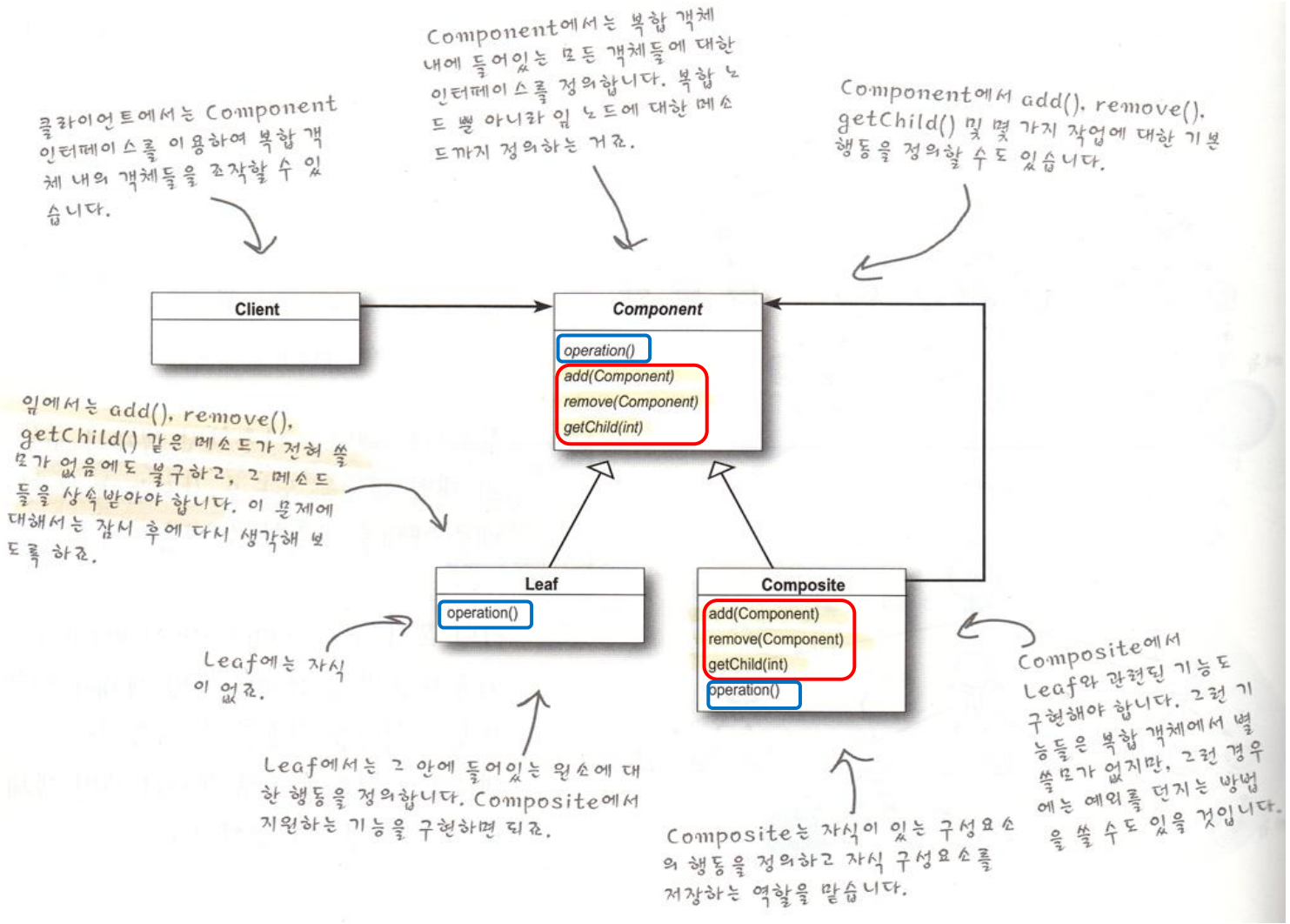
Composite Pattern

- 객체들을 트리 구조로 구성하여 부분과 전체를 나타내는 계층 구조로 만들 수 있다.
  - 복합 구조(composite structure)
  - 복합 객체와 개별 객체에 대하여 똑같은 작업 적용 가능



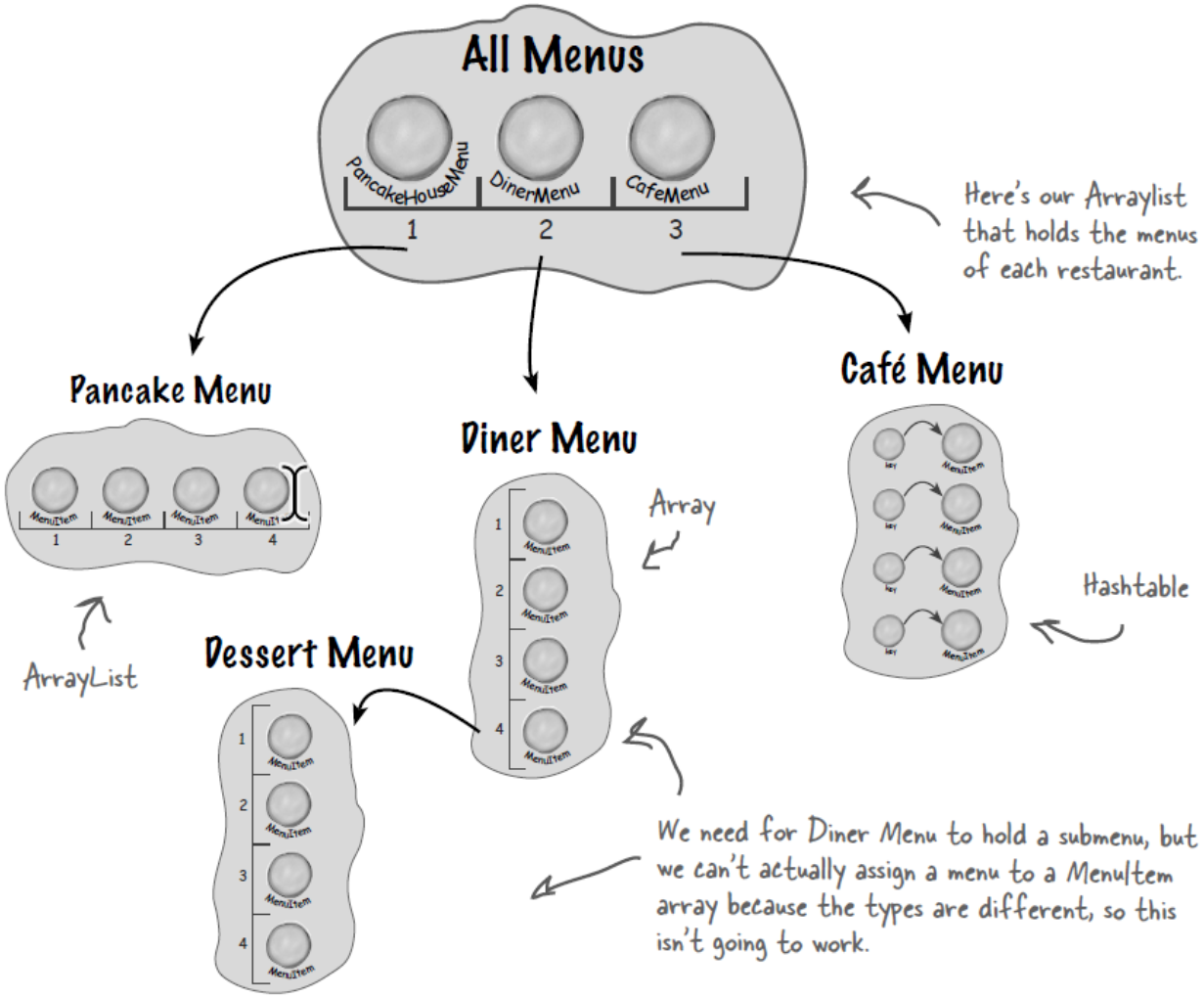


Composite 패턴 클래스 다이어그램

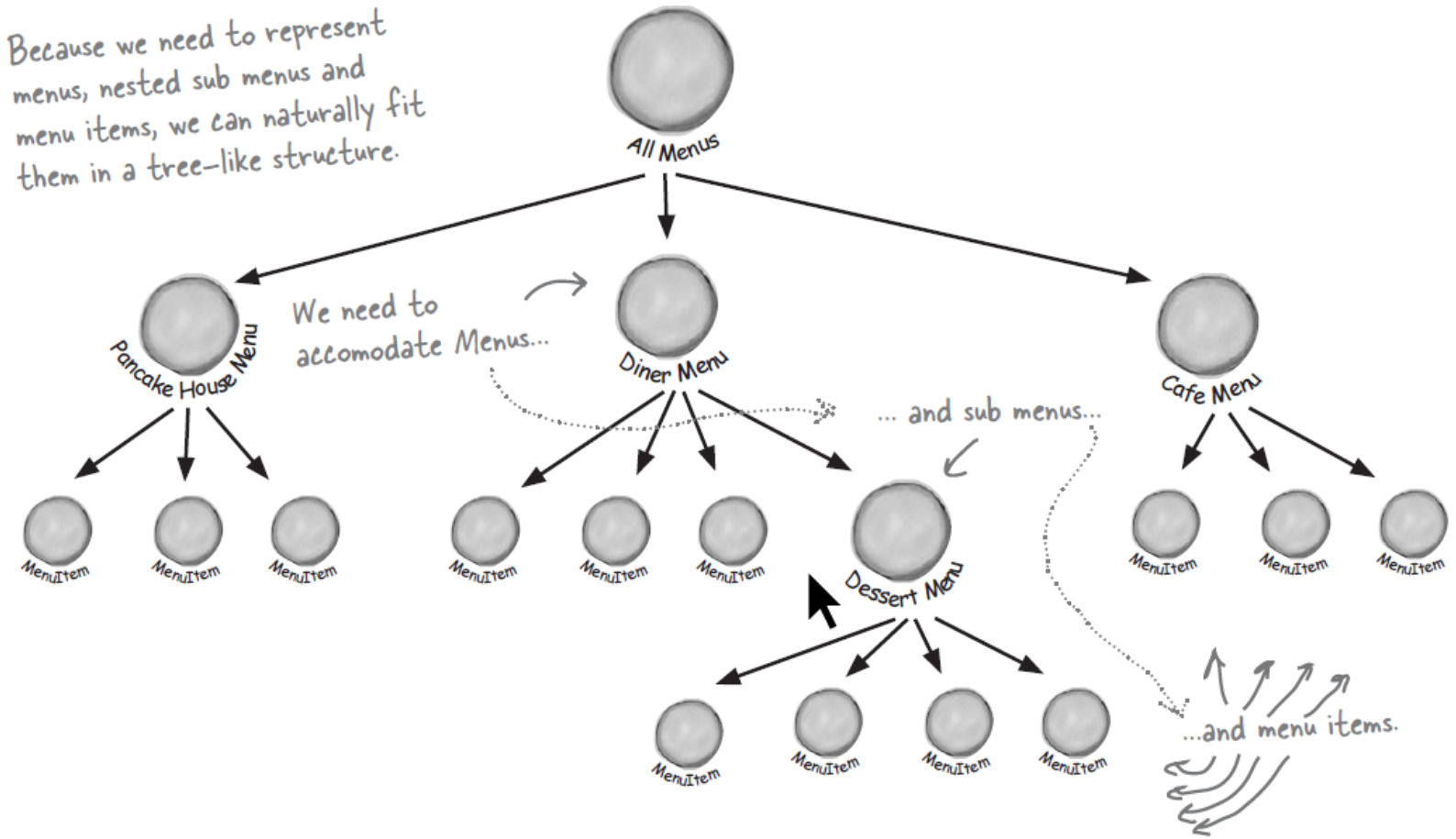


composite.menus1 모델링

- 요구사항 변경
  - Diner Menu 안에 Dessert Menu를 추가
- 해결 방안
  - composite pattern 적용



객체 관점 메뉴 구성


Because we need to represent menus, nested sub menus and menu items, we can naturally fit them in a tree-like structure.
All Menus
Pancake House Menu
We need to accomodate Menus...
Diner Menu
Cafe Menu
... and sub menus...
Dessert Menu
MenuItem
...and menu items.

MenuComponent.java

(주의) 인터페이스로 정의하면 메서드를 선택적으로 사용할 수 없음!

```
1   package cse.design_pattern.ch09.composite.menus1;
2
3   public abstract class MenuComponent {
4
5       public void add(MenuComponent menuComponent) {
6           throw new UnsupportedOperationException();
7       }
8
9       public void remove(MenuComponent menuComponent) {
10          throw new UnsupportedOperationException();
11      }
12
13      public MenuComponent getChild(int i) {
14          throw new UnsupportedOperationException();
15      }
```

```
    public String getName() {
        throw new UnsupportedOperationException();
    }

    public String getDescription() {
        throw new UnsupportedOperationException();
    }

    public double getPrice() {
        throw new UnsupportedOperationException();
    }

    public boolean isVegetarian() {
        throw new UnsupportedOperationException();
    }

    public void print() {
        throw new UnsupportedOperationException();
    }
}
```

MenuItem.java

```java
package cse.design_pattern.ch09.composite.menus1;

public class MenuItem extends MenuComponent {

    private final String name;
    private final String description;
    private final boolean vegetarian;
    private final double price;

    public MenuItem(String name,
            String description,
            boolean vegetarian,
            double price) {
        this.name = name;
        this.description = description;
        this.vegetarian = vegetarian;
        this.price = price;
    }
```

```
20      @Override
        public String getName() {
22          return name;
23      }
24
25      @Override
        public String getDescription() {
27          return description;
28      }
29
30      @Override
        public double getPrice() {
32          return price;
33      }
34
35      @Override
        public boolean isVegetarian() {
37          return vegetarian;
38      }
```

```
40      @Override
        public void print() {
42          System.out.print("  " + getName());
43          if (isVegetarian()) {
44              System.out.print("(v)");
45          }
46          System.out.println(", " + getPrice());
47          System.out.println("     -- " + getDescription());
48      }
49  }
```

Menu.java

```java
package cse.design_pattern.ch09.composite.menus1;

import java.util.Iterator;
import java.util.ArrayList;

public class Menu extends MenuComponent {

    private final ArrayList<MenuComponent> menuComponents;
    private final String name;
    private final String description;

    public Menu(String name, String description) {
        this.menuComponents = new ArrayList<>();
        this.name = name;
        this.description = description;
    }
```

```
18      @Override
        public void add(MenuComponent menuComponent) {
20          menuComponents.add(menuComponent);
21      }
22
23      @Override
        public void remove(MenuComponent menuComponent) {
25          menuComponents.remove(menuComponent);
26      }
27
28      @Override
        public MenuComponent getChild(int i) {
30          return menuComponents.get(i);
31      }
32
33      @Override
        public String getName() {
35          return name;
36      }
```

```
38      @Override
        public String getDescription() {
40          return description;
41      }
42
43      @Override
        public void print() {
45          System.out.print("₩n" + getName());
46          System.out.println(", " + getDescription());
47          System.out.println("---------------------");
48
49          Iterator<MenuComponent> iterator = menuComponents.iterator();
50          while (iterator.hasNext()) {
51              MenuComponent menuComponent = iterator.next();
52              menuComponent.print();
53          }
54      }
55  }
```

Waitress.java

```java
package cse.design_pattern.ch09.composite.menus1;

public class Waitress {

    private final MenuComponent allMenus;

    public Waitress(MenuComponent allMenus) {
        this.allMenus = allMenus;
    }

    public void printMenu() {
        allMenus.print();
    }
}
```

MenuTestDrive.java

```
1   package cse.design_pattern.ch09.composite.menus1;
2
3   public class MenuTestDrive {
4
5       public static void main(String args[]) {
6           MenuComponent pancakeHouseMenu = new Menu("PANCAKE HOUSE MENU", "Breakfast");
7           MenuComponent dinerMenu = new Menu("DINER MENU", "Lunch");
8           MenuComponent cafeMenu = new Menu("CAFE MENU", "Dinner");
9           MenuComponent dessertMenu = new Menu("DESSERT MENU", "Dessert of course!");
10          MenuComponent coffeeMenu = new Menu("COFFEE MENU", "Stuff to go with your afternoon coffee");
11
12          MenuComponent allMenus = new Menu("ALL MENUS", "All menus combined");
```

```
14          allMenus.add(pancakeHouseMenu);
15          allMenus.add(dinerMenu);
16          allMenus.add(cafeMenu);
17
18          pancakeHouseMenu.add(new MenuItem(
19                  "K&B's Pancake Breakfast",
20                  "Pancakes with scrambled eggs, and toast",
21                  true,
22                  2.99));
23          pancakeHouseMenu.add(new MenuItem(
24                  "Regular Pancake Breakfast",
25                  "Pancakes with fried eggs, sausage",
26                  false,
27                  2.99));
28          pancakeHouseMenu.add(new MenuItem(
29                  "Blueberry Pancakes",
30                  "Pancakes made with fresh blueberries, and blueberry syrup",
31                  true,
32                  3.49));
```

```
33          pancakeHouseMenu.add(new MenuItem(
34                  "Waffles",
35                  "Waffles, with your choice of blueberries or strawberries",
36                  true,
37                  3.59));
38
39          dinerMenu.add(new MenuItem(
40                  "Vegetarian BLT",
41                  "(Fakin') Bacon with lettuce & tomato on whole wheat",
42                  true,
43                  2.99));
44          dinerMenu.add(new MenuItem(
45                  "BLT",
46                  "Bacon with lettuce & tomato on whole wheat",
47                  false,
48                  2.99));
49          dinerMenu.add(new MenuItem(
50                  "Soup of the day",
51                  "A bowl of the soup of the day, with a side of potato salad",
52                  false,
53                  3.29));
```

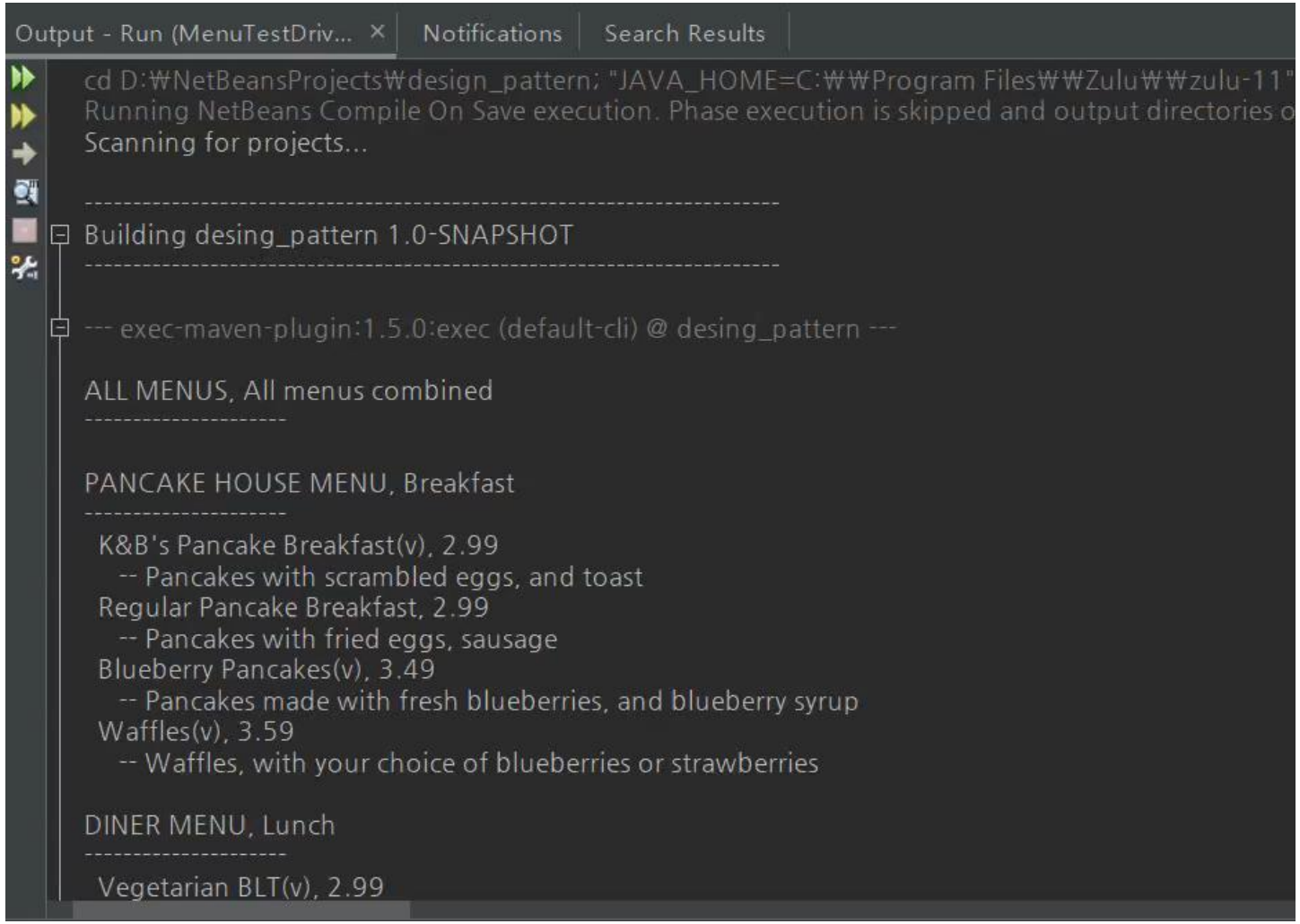
```
54          dinerMenu.add(new MenuItem(
55                  "Hotdog",
56                  "A hot dog, with saurkraut, relish, onions, topped with cheese",
57                  false,
58                  3.05));
59          dinerMenu.add(new MenuItem(
60                  "Steamed Veggies and Brown Rice",
61                  "Steamed vegetables over brown rice",
62                  true,
63                  3.99));
64
65          dinerMenu.add(new MenuItem(
66                  "Pasta",
67                  "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
68                  true,
69                  3.89));
70
71          dinerMenu.add(dessertMenu);
```

```
73          dessertMenu.add(new MenuItem(
74                  "Apple Pie",
75                  "Apple pie with a flakey crust, topped with vanilla icecream",
76                  true,
77                  1.59));
78
79          dessertMenu.add(new MenuItem(
80                  "Cheesecake",
81                  "Creamy New York cheesecake, with a chocolate graham crust",
82                  true,
83                  1.99));
84          dessertMenu.add(new MenuItem(
85                  "Sorbet",
86                  "A scoop of raspberry and a scoop of lime",
87                  true,
88                  1.89));
89
90          cafeMenu.add(new MenuItem(
91                  "Veggie Burger and Air Fries",
92                  "Veggie burger on a whole wheat bun, lettuce, tomato, and fries",
93                  true,
94                  3.99));
```

```
95          cafeMenu.add(new MenuItem(
96                  "Soup of the day",
97                  "A cup of the soup of the day, with a side salad",
98                  false,
99                  3.69));
100         cafeMenu.add(new MenuItem(
101                 "Burrito",
102                 "A large burrito, with whole pinto beans, salsa, guacamole",
103                 true,
104                 4.29));
105
106         cafeMenu.add(coffeeMenu);
107
108         coffeeMenu.add(new MenuItem(
109                 "Coffee Cake",
110                 "Crumbly cake topped with cinnamon and walnuts",
111                 true,
112                 1.59));
```

```
113             coffeeMenu.add(new MenuItem(
114                     "Bagel",
115                     "Flavors include sesame, poppyseed, cinnamon raisin, pumpkin",
116                     false,
117                     0.69));
118             coffeeMenu.add(new MenuItem(
119                     "Biscotti",
120                     "Three almond or hazelnut biscotti cookies",
121                     true,
122                     0.89));
123
124             Waitress waitress = new Waitress(allMenus);
125
126             waitress.printMenu();
127         }
128     }
```

실행 결과


Output - Run (MenuTestDriv...
Notifications
Search Results
cd D:₩NetBeansProjects₩design_pattern; "JAVA_HOME=C:₩₩Program Files₩₩Zulu₩₩zulu-11"
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories o
Scanning for projects...
Building desing_pattern 1.0-SNAPSHOT
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---
ALL MENUS, All menus combined
PANCAKE HOUSE MENU, Breakfast
K&B's Pancake Breakfast(v), 2.99
-- Pancakes with scrambled eggs, and toast
Regular Pancake Breakfast, 2.99
-- Pancakes with fried eggs, sausage
Blueberry Pancakes(v), 3.49
-- Pancakes made with fresh blueberries, and blueberry syrup
Waffles(v), 3.59
-- Waffles, with your choice of blueberries or strawberries
DINER MENU, Lunch
Vegetarian BLT(v), 2.99

composite.menus2

- 컴포지트 패턴 내에서 이터레이터 패턴 활용
- MenuComponent 추상 클래스에 createIterator() 메소드 추가
  - Menu & menuItem 클래스에 createIterator() 메소드 정의

MenuComponent.java

```
1   package cse.design_pattern.ch09.composite.menus2;
2
3   import java.util.Iterator;
4
5   public abstract class MenuComponent {
6
7       public void add(MenuComponent menuComponent) {
8           throw new UnsupportedOperationException();
9       }
10
11      public void remove(MenuComponent menuComponent) {
12          throw new UnsupportedOperationException();
13      }
14
15      public MenuComponent getChild(int i) {
16          throw new UnsupportedOperationException();
17      }
```

```
    public String getName() {
        throw new UnsupportedOperationException();
    }

    public String getDescription() {
        throw new UnsupportedOperationException();
    }

    public double getPrice() {
        throw new UnsupportedOperationException();
    }

    public boolean isVegetarian() {
        throw new UnsupportedOperationException();
    }

    // 추가된 추상 메소드
    public abstract Iterator<MenuComponent> createIterator();

    public void print() {
        throw new UnsupportedOperationException();
    }
}
```

MenuItem.java

```java
package cse.design_pattern.ch09.composite.menus2;

import java.util.Iterator;

public class MenuItem extends MenuComponent {

    private final String name;
    private String description;
    private boolean vegetarian;
    private double price;

    public MenuItem(String name,
            String description,
            boolean vegetarian,
            double price) {
        this.name = name;
        this.description = description;
        this.vegetarian = vegetarian;
        this.price = price;
    }
```

```
22      @Override
        public String getName() {
24          return name;
25      }
26
27      @Override
        public String getDescription() {
29          return description;
30      }
31
32      @Override
        public double getPrice() {
34          return price;
35      }
36
37      @Override
        public boolean isVegetarian() {
39          return vegetarian;
40      }
```

```
42      @Override
        public Iterator<MenuComponent> createIterator() {
44          return new NullIterator();
45      }
46
47      @Override
        public void print() {
49          System.out.print("  " + getName());
50          if (isVegetarian()) {
51              System.out.print("(v)");
52          }
53          System.out.println(", " + getPrice());
54          System.out.println("     -- " + getDescription());
55      }
56
57  }
```

Leaf 노드 임을 의미

Menu.java

```
1   package cse.design_pattern.ch09.composite.menus2;
2
3   import java.util.Iterator;
4   import java.util.ArrayList;
5
6   public class Menu extends MenuComponent {
7
8       private Iterator<MenuComponent> iterator = null;
9       private final ArrayList<MenuComponent> menuComponents =
10              new ArrayList<>();
11      private final String name;
12      private final String description;
13
14      public Menu(String name, String description) {
15          this.name = name;
16          this.description = description;
17      }
```

```
19      @Override
        public void add(MenuComponent menuComponent) {
21          menuComponents.add(menuComponent);
22      }
23
24      @Override
        public void remove(MenuComponent menuComponent) {
26          menuComponents.remove(menuComponent);
27      }
28
29      @Override
        public MenuComponent getChild(int i) {
31          return menuComponents.get(i);
32      }
33
34      @Override
        public String getName() {
36          return name;
37      }
38
39      @Override
        public String getDescription() {
41          return description;
42      }
```

```
43      // 새로 추가된 메소드
44      @Override
        public Iterator<MenuComponent> createIterator() {
46          if (iterator == null) {
47              iterator = new CompositeIterator(menuComponents.iterator());
48          }
49          return iterator;
50      }
51
52      @Override
        public void print() {
54          System.out.print("₩n" + getName());
55          System.out.println(", " + getDescription());
56          System.out.println("---------------------");
57
58          Iterator<MenuComponent> it = menuComponents.iterator();
59          while (it.hasNext()) {
60              MenuComponent menuComponent = it.next();
61              menuComponent.print();
62          }
63      }
64  }
```

NullIterator.java

```java
package cse.design_pattern.ch09.composite.menus2;

import java.util.Iterator;

public class NullIterator implements Iterator<MenuComponent> {

    @Override
    public MenuComponent next() {
        return null;
    }

    @Override
    public boolean hasNext() {
        return false;
    }
```

```
17        /*
18                * No longer needed as of Java 8
19                *
20                * (non-Javadoc)
21                * @see java.util.Iterator#remove()
22                *
23               public void remove() {
24                        throw new UnsupportedOperationException();
25               }
26         */
27    }
```

CompositeIterator.java

```java
package cse.design_pattern.ch09.composite.menus2;

import java.util.Iterator;
import java.util.Stack;

public class CompositeIterator implements Iterator<MenuComponent> {

    // Stack<Iterator<MenuComponent>> stack = new Stack<Iterator<MenuComponent>>();
    private final Stack<Iterator<MenuComponent>> stack = new Stack<>();

    public CompositeIterator(Iterator<MenuComponent> iterator) {
        stack.push(iterator);
    }
```

```
15      @Override
        public MenuComponent next() {
17          if (hasNext()) {
18              Iterator<MenuComponent> iterator = stack.peek();
19              MenuComponent component = iterator.next();
20              stack.push(component.createIterator());
21              return component;
22          } else {
23              return null;
24          }
25      }
```

```
27      @Override
        public boolean hasNext() {
29          if (stack.empty()) {
30              return false;
31          } else {
32              Iterator<MenuComponent> iterator = stack.peek();
33              if (!iterator.hasNext()) {
34                  stack.pop();
35                  return hasNext();
36              } else {
37                  return true;
38              }
39          }
40      }
41
42  }
```

Waitress.java

```java
package cse.design_pattern.ch09.composite.menus2;

import java.util.Iterator;

public class Waitress {

    private MenuComponent allMenus;

    public Waitress(MenuComponent allMenus) {
        this.allMenus = allMenus;
    }

    public void printMenu() {
        allMenus.print();
    }
```

```
17      public void printVegetarianMenu() {
18          Iterator<MenuComponent> iterator = allMenus.createIterator();
19
20          System.out.println("₩nVEGETARIAN MENU₩n----");
21          while (iterator.hasNext()) {
22              MenuComponent menuComponent = iterator.next();
23              try {
24                  if (menuComponent.isVegetarian()) {
25                      menuComponent.print();
26                  }
27              } catch (UnsupportedOperationException e) {
28              }
29          }
30      }
31  }
```

← hook 개념 적용 가능?

MenuTestDrive.java

```
1   package cse.design_pattern.ch09.composite.menus2;
2
3   public class MenuTestDrive {
4
5       public static void main(String args[]) {
6
7           MenuComponent pancakeHouseMenu = new Menu("PANCAKE HOUSE MENU", "Breakfast");
8           MenuComponent dinerMenu = new Menu("DINER MENU", "Lunch");
9           MenuComponent cafeMenu = new Menu("CAFE MENU", "Dinner");
10          MenuComponent dessertMenu = new Menu("DESSERT MENU", "Dessert of course!");
11
12          MenuComponent allMenus = new Menu("ALL MENUS", "All menus combined");
13
14          allMenus.add(pancakeHouseMenu);
15          allMenus.add(dinerMenu);
16          allMenus.add(cafeMenu);
```

```
18          pancakeHouseMenu.add(new MenuItem(
19                  "K&B's Pancake Breakfast",
20                  "Pancakes with scrambled eggs, and toast",
21                  true,
22                  2.99));
23          pancakeHouseMenu.add(new MenuItem(
24                  "Regular Pancake Breakfast",
25                  "Pancakes with fried eggs, sausage",
26                  false,
27                  2.99));
28          pancakeHouseMenu.add(new MenuItem(
29                  "Blueberry Pancakes",
30                  "Pancakes made with fresh blueberries, and blueberry syrup",
31                  true,
32                  3.49));
33          pancakeHouseMenu.add(new MenuItem(
34                  "Waffles",
35                  "Waffles, with your choice of blueberries or strawberries",
36                  true,
37                  3.59));
```

```
39          dinerMenu.add(new MenuItem(
40                  "Vegetarian BLT",
41                  "(Fakin') Bacon with lettuce & tomato on whole wheat",
42                  true,
43                  2.99));
44          dinerMenu.add(new MenuItem(
45                  "BLT",
46                  "Bacon with lettuce & tomato on whole wheat",
47                  false,
48                  2.99));
49          dinerMenu.add(new MenuItem(
50                  "Soup of the day",
51                  "A bowl of the soup of the day, with a side of potato salad",
52                  false,
53                  3.29));
54          dinerMenu.add(new MenuItem(
55                  "Hotdog",
56                  "A hot dog, with saurkraut, relish, onions, topped with cheese",
57                  false,
58                  3.05));
```

```
59          dinerMenu.add(new MenuItem(
60                  "Steamed Veggies and Brown Rice",
61                  "A medly of steamed vegetables over brown rice",
62                  true,
63                  3.99));
64
65          dinerMenu.add(new MenuItem(
66                  "Pasta",
67                  "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
68                  true,
69                  3.89));
70
71          dinerMenu.add(dessertMenu);
72
73          dessertMenu.add(new MenuItem(
74                  "Apple Pie",
75                  "Apple pie with a flakey crust, topped with vanilla icecream",
76                  true,
77                  1.59));
78          dessertMenu.add(new MenuItem(
79                  "Cheesecake",
80                  "Creamy New York cheesecake, with a chocolate graham crust",
81                  true,
82                  1.99));
```

```
83          dessertMenu.add(new MenuItem(
84                  "Sorbet",
85                  "A scoop of raspberry and a scoop of lime",
86                  true,
87                  1.89));
88
89          cafeMenu.add(new MenuItem(
90                  "Veggie Burger and Air Fries",
91                  "Veggie burger on a whole wheat bun, lettuce, tomato, and fries",
92                  true,
93                  3.99));
94          cafeMenu.add(new MenuItem(
95                  "Soup of the day",
96                  "A cup of the soup of the day, with a side salad",
97                  false,
98                  3.69));
99          cafeMenu.add(new MenuItem(
100                 "Burrito",
101                 "A large burrito, with whole pinto beans, salsa, guacamole",
102                 true,
103                 4.29));
```

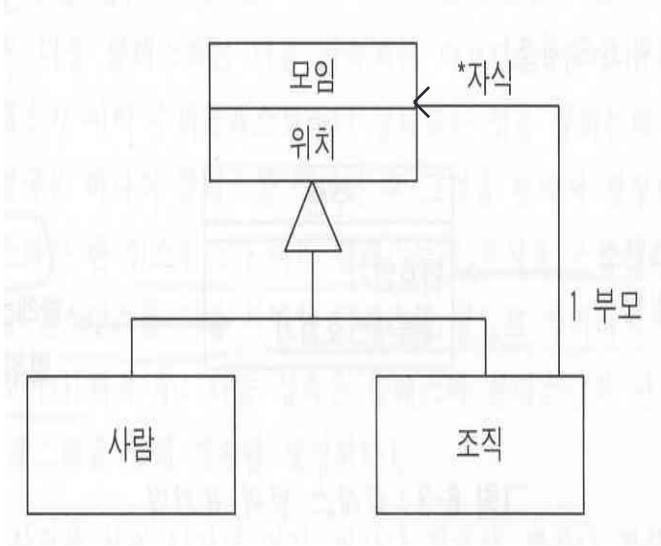
```
105             Waitress waitress = new Waitress(allMenus);
106
107             waitress.printVegetarianMenu();
108             //waitress.printMenu();
109         }
110     }
```

실행 결과

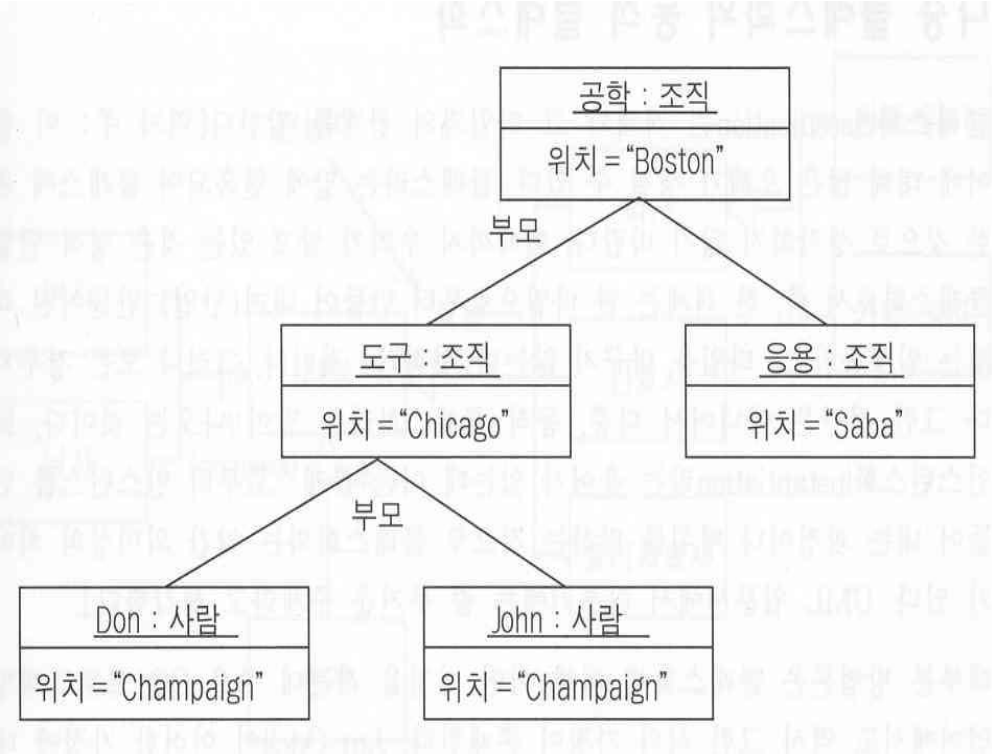
```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ desing_pattern ---

VEGETARIAN MENU
----
 K&B's Pancake Breakfast(v), 2.99
   -- Pancakes with scrambled eggs, and toast
 Blueberry Pancakes(v), 3.49
   -- Pancakes made with fresh blueberries, and blueberry syrup
 Waffles(v), 3.59
   -- Waffles, with your choice of blueberries or strawberries
 Vegetarian BLT(v), 2.99
   -- (Fakin') Bacon with lettuce & tomato on whole wheat
 Steamed Veggies and Brown Rice(v), 3.99
   -- A medly of steamed vegetables over brown rice
 Pasta(v), 3.89
   -- Spaghetti with Marinara Sauce, and a slice of sourdough bread
 Apple Pie(v), 1.59
   -- Apple pie with a flakey crust, topped with vanilla icecream
 Cheesecake(v), 1.99
   -- Creamy New York cheesecake, with a chocolate graham crust
 Sorbet(v), 1.89
   -- A scoop of raspberry and a scoop of lime
 Veggie Burger and Air Fries(v), 3.99
   -- Veggie burger on a whole wheat bun, lettuce, tomato, and fries
 Burrito(v), 4.29
   -- A large burrito, with whole pinto beans, salsa, guacamole
```

참고 자료: Composite Pattern

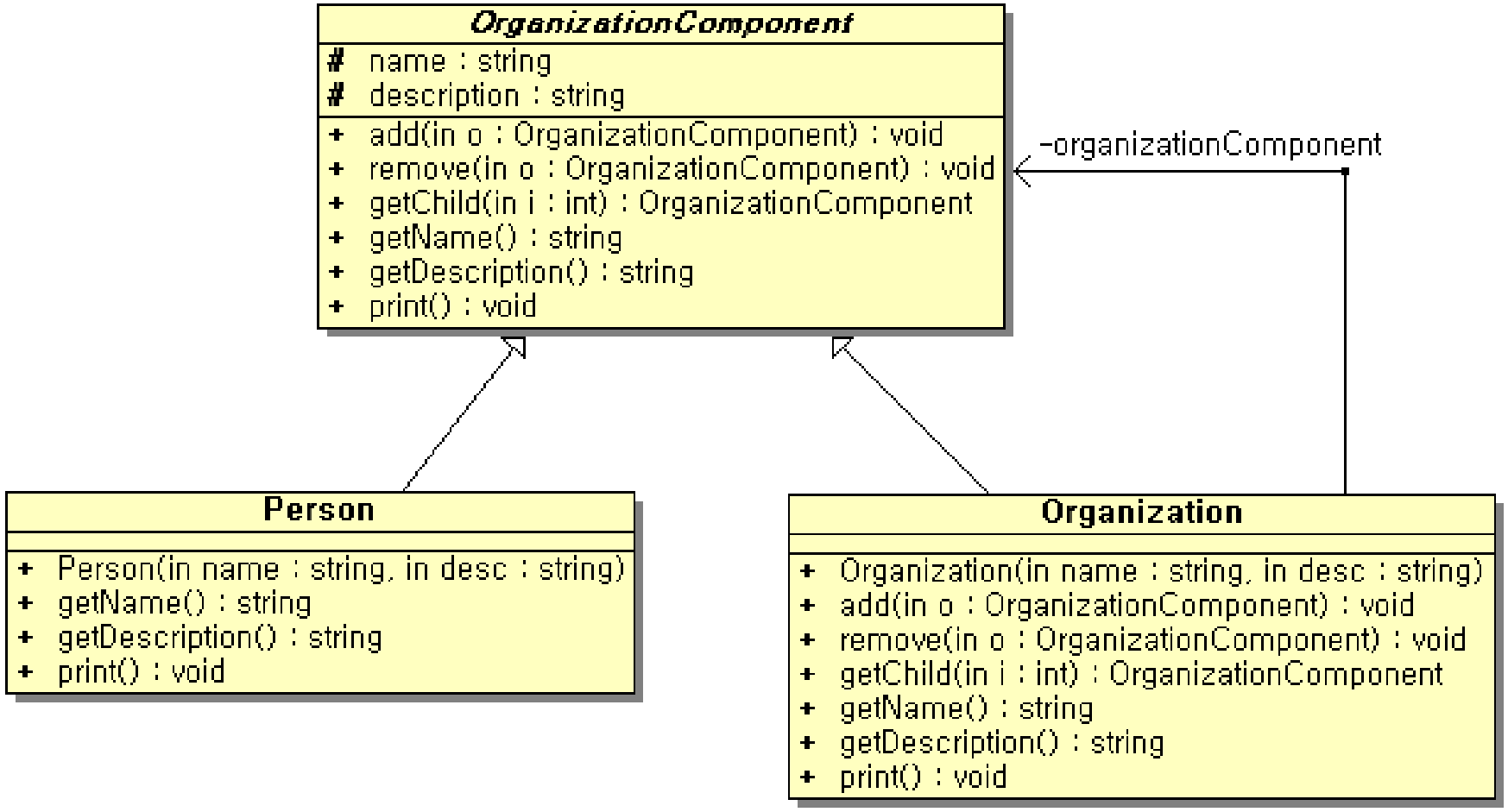
조직 구성 구조에 대한 클래스도



조직 인스턴스들에 대한 사례를 보여주는 객체도



"조직" 클래스 다이어그램


OrganizationComponent
name : string
description : string
+ add(in o : OrganizationComponent) : void
+ remove(in o : OrganizationComponent) : void
+ getChild(in i : int) : OrganizationComponent
+ getName() : string
+ getDescription() : string
+ print() : void
-organizationComponent
Person
+ Person(in name : string, in desc : string)
+ getName() : string
+ getDescription() : string
+ print() : void
Organization
+ Organization(in name : string, in desc : string)
+ add(in o : OrganizationComponent) : void
+ remove(in o : OrganizationComponent) : void
+ getChild(in i : int) : OrganizationComponent
+ getName() : string
+ getDescription() : string
+ print() : void

OrganizationComponent 클래스

```
abstract class OrganizationComponent {

    protected String name;
    protected String description;

    public void add(OrganizationComponent o) {
        throw new UnsupportedOperationException();
    }

    public void remove(OrganizationComponent o) {
        throw new UnsupportedOperationException();
    }

    public OrganizationComponent getChild(int i) {
        throw new UnsupportedOperationException();
    }

    public String getName() {
        throw new UnsupportedOperationException();
    }

    public String getDescription() {
        throw new UnsupportedOperationException();
    }

    public void print() {
        throw new UnsupportedOperationException();
    }
}
```

Organization에서 사용

둘다 사용

Person 클래스

```
class Person extends OrganizationComponent {

    public Person(String name, String desc) {
        this.name = name;
        this.description = desc;

    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String getDescription() {
        return description;
    }

    @Override
    public void print() {
        System.out.print("Name: " + getName());
        System.out.println("  Descripton: " + getDescription());
    }
}
```

Organization 클래스

```
import java.util.ArrayList;
import java.util.List;

class Organization extends OrganizationComponent {

    private List<OrganizationComponent> organizationComponent =
            new ArrayList<OrganizationComponent>();

    public Organization(String name, String desc) {
        this.name = name;
        this.description = desc;
    }

    @Override
    public void add(OrganizationComponent o) {
        organizationComponent.add(o);
    }

    @Override
    public void remove(OrganizationComponent o) {
        if (organizationComponent.contains(o))
            organizationComponent.remove(o);
    }

    @Override
    public OrganizationComponent getChild(int i) {
        return organizationComponent.get(i);
    }
```

```
    @Override
    public String getName() {
        return name;
    }

    @Override
    public String getDescription() {
        return description;
    }

    @Override
    public void print() {
        System.out.print("\nOrganization: " + getName());
        System.out.println("  Description: " + getDescription());

        for (OrganizationComponent comp : organizationComponent) {
            comp.print();
        }
    }
}
```

TestDriver 클래스

```
public class TestDriver {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        OrganizationComponent allOrg = new Organization("공학", "Boston");

        OrganizationComponent tool = new Organization("도구", "Chicago");
        OrganizationComponent app = new Organization("응용", "Saba");

        OrganizationComponent don = new Person("Don", "Champaign");
        OrganizationComponent john = new Person("John", "Champaign");

        allOrg.add(tool);
        allOrg.add(app);

        tool.add(don);
        tool.add(john);

        allOrg.print();
    }
}
```

실행 결과

```
run:

Organization: 공학  Description: Boston

Organization: 도구  Description: Chicago
Name: Don  Descripton: Champaign
Name: John  Descripton: Champaign

Organization: 응용  Description: Saba
```