[문제 1] 다음과 같은 조건을 만족하는 프로그램을 작성 하시오

MVC패턴을 이용하여 컬렉션/제네릭을 적용하는 문제다. 해당 구현 클래스 다이어그램과 클래스 구조를 참고하여 프로젝트를 완성하시오.

1. 구현 클래스 다이어그램 (Class Diagram)

Farm
- kind:String
+ Farm() + Farm(kind:String) + setter() / getter() + toString() : String + hashCode() : int + equals() : boolean

Fruit
- name:String
+ Fruit() + Fruit(kind:String, name:String) + setter() / getter() + toString() : String + hashCode() : int + equals() : boolean

Vegetable
- name:String
+ Vegetable() + Vegetable(kind:String, name:String) + setter() / getter() + toString() : String + hashCode() : int + equals() : boolean

Nut
- name:String
+ Nut() + Nut(kind:String, name:String) + setter() / getter() + toString() : String + hashCode() : int + equals() : boolean

FarmController
- hMap:HashMap<Farm, Integer> = new HashMap<>(); - list:ArrayList<Farm> = new ArrayList<>();
+ addNewKind(f:Farm, amount:int) : boolean + removeKind(f:Farm) : boolean + changeAmount(f:Farm, amount:int) : boolean + printFarm() : HashMap<Farm, Integer> + buyFarm(f:Farm) : boolean + removeFarm(f:Farm) : boolean + printBuyFarm() : ArrayList<Farm>

FarmMenu
- sc:Scanner = new Scanner(System.in) - fc : FarmController = new FarmController()
+ mainMenu() : void + adminMenu() : void + customerMenu() : void + addNewKind() : void + removeKind() : void + changeAmount() : void + printFarm() : void + buyFarm() : void + removeFarm() : void + printBuyFarm() : void

Run
+ main(args:String[]) : void

2. 구현 클래스 설명

Package명	Class명	Method	설명
com.bs.practice.generics.run	Run	+main(args:String[]) : void	FarmMenu 객체를 생성 후 mainMenu() 실행
com.bs.practice.generics.view	FarmMenu	+ mainMenu() : void	사용자가 직접 메인 메뉴를 선택할 수 있음. 종료 전까지 반복 실행. 각 메뉴를 누를 시 해당 메소드로 이동.
		+ adminMenu() : void	사용자가 직접 직원 메뉴를 선택할 수 있으며 종료 전까지 반복 실행. 각 메뉴를 누를 시 해당 메소드로 이동
		+ customerMenu() : void	사용자가 직접 고객 메뉴를 선택할 수 있으며 종료 전까지 반복 실행. 각 메뉴를 누를 시 해당 메소드로 이동
		+addNewKind() : void	새로운 농산물 추가 성공을 알리는 메소드
		+removeKind() : void	농산물 삭제 성공을 알리는 메소드
		+changeAmount() : void	농산물 수량 정보 수정 성공을 알리는 메소드
		+printFarm():void	농산물을 출력하는 메소드
		+buyFarm():void	구매 성공을 알리는 메소드
		+removeFarm():void	구매 취소 성공을 알리는 메소드
		+printBuyFarm():void	구매 목록 출력하는 메소드
com.bs.practice.generics.controller	FarmController	+addNewKind(f:Farm, amount:int):boolean	맵에 키와 값을 저장하여 농산물 추가하는 메소드
		+removeKind(f:Farm) :boolean	농산물 제거하는 메소드
		+changeAmount(f:Farm, amount:int):boolean	수량 변경하는 메소드
		+printFarm():HashMap<Farm, Integer>	농산물을 저장한 map을 반환하는 메소드
		+buyFarm(f:Farm):boolean	농산물을 사는 메소드
		+removeFarm(f:Farm):boolean	구매한 농산물을 취소하는 메소드

		+printBuyFarm():ArrayList<Farm>	구매 정보를 저장한 list를 반환하는 메소드
com.bs.practice.generics.model.vo	Farm	+ Farm()	기본 생성자
		+Farm(kind:String)	매개변수가 있는 생성자
		+ toString() : String	객체의 정보를 리턴하는 메소드
		+ hashCode():int	해시코드를 반환하는 메소드
		+ equals():boolean	데이터가 같은지 여부를 반환하는 메소드
	Fruit	+ Fruit()	기본 생성자
		+Fruit(kind:String, name:String)	매개변수가 있는 생성자
		+ toString() : String	객체의 정보를 리턴하는 메소드
		+ hashCode():int	해시코드를 반환하는 메소드
		+ equals():boolean	데이터가 같은지 여부를 반환하는 메소드
	Vegetable	+ Vegetable()	기본 생성자
		+Vegetable(kind:String, name:String)	매개변수가 있는 생성자
		+ toString() : String	객체의 정보를 리턴하는 메소드
		+ hashCode():int	해시코드를 반환하는 메소드
		+ equals():boolean	데이터가 같은지 여부를 반환하는 메소드
	Nut	+ Nut()	기본 생성자
		+Nut(kind:String, name:String)	매개변수가 있는 생성자
		+ toString() : String	객체의 정보를 리턴하는 메소드
		+ hashCode():int	해시코드를 반환하는 메소드
		+ equals():boolean	데이터가 같은지 여부를 반환하는 메소드

3. class 구조

```
public class FarmMenu {
        // FarmController 객체 생성
       // Scanner 객체
       public void mainMenu() {
                System.out.println("========== BS 마트 ==========");
               ******* 메인 메뉴 *******
                1. 직원메뉴                  // adminMenu() 실행
                2. 손님 메뉴                 // customerMenu()
                9. 종료                      // "프로그램 종료." 출력 후 main()으로 리턴
                메뉴 번호 선택 : >> 입력 받음        // 메뉴 화면 반복 실행 처리
                // 잘 못 입력 하였을 경우 "잘못 입력하였습니다. 다시 입력해주세요" 출력 후 반복
       }
       public void adminMenu() {
                ******* 직원 메뉴 *******
                1. 새 농산물 추가 // addNewKind() 실행
                2. 종류 삭제                 // removeKind()
                3. 수량 수정                 // changeAmount()
                4. 농산물 목록               // printFarm()
                9. 메인으로 돌아가기         // mainMenu()로 리턴
                메뉴 번호 선택 : >> 입력 받음        // 메뉴 화면 반복 실행 처리
                // 잘 못 입력 하였을 경우 "잘못 입력하였습니다. 다시 입력해주세요" 출력 후 반복
        }
       public void customerMenu() {
        현재 BS마트 농산물 수량
        printFarm();
                ******* 고객 메뉴 *******
                1. 농산물 사기               // buyFarm() 실행
                2. 농산물 빼기               // removeFarm()
                3. 구입한 농산물 보기        // printBuyFarm()
                9. 메인으로 돌아가기         // mainMenu()로 리턴
                메뉴 번호 선택 : >> 입력 받음        // 메뉴 화면 반복 실행 처리
                // 잘 못 입력 하였을 경우 "잘못 입력하였습니다. 다시 입력해주세요" 출력 후 반복
        }
```

```
// 1-1. 새 농산물 추가용 view 메소드
public void addNewKind() {
        '1. 과일 / 2. 채소 / 3. 견과'를 통해 번호로 종류를 받고 추가 농산물의 이름, 수량도
        받음. 이때 없는 번호를 선택하면 "잘못 입력하셨습니다. 다시 입력해주세요."가
        출력되며 다시 번호를 받고, 선택한 종류에 따라 생성되는 객체가 다름.
        객체 안에 종류와 이름을 저장. 데이터를 저장한 객체와 수량을 fc(FarmController)의
        addNewKind()로 넘기고 전달 받은 반환 값이 true면 "새 농산물이 추가되었습니다.",
        false면 "새 농산물 추가에 실패하였습니다. 다시 입력해주세요." 출력
}
// 1-2. 종류 삭제용 view 메소드
public void removeKind() {
        '1. 과일 / 2. 채소 / 3. 견과'를 통해 번호로 종류를 받고 삭제할 농산물의 이름도
        받음. 이때 없는 번호를 선택하면 "잘못 입력하셨습니다. 다시 입력해주세요."가
        출력되며 다시 번호를 받고, 선택한 종류에 따라 생성되는 객체가 다름.
        객체 안에 종류와 이름을 저장. 데이터를 저장한 객체를 fc의 removeKind()로 넘기고
        전달받은 반환 값이 true면 "농산물 삭제에 성공하였습니다.",
        false면 "새 농산물 추가에 실패하였습니다. 다시 입력해주세요." 출력
 }
// 1-3. 수량 수정용 view 메소드
public void changeAmount() {
        '1. 과일 / 2. 채소 / 3. 견과'를 통해 번호로 종류를 받고 수정 농산물의 이름,
        수정할 수량도 받음. 없는 번호 선택 시 "잘못 입력하셨습니다. 다시 입력해주세요."
        출력 후 다시 번호를 받고, 선택한 종류에 따라 생성되는 객체가 다름.
        객체 안에 종류와 이름을 저장. 데이터를 저장한 객체와 수량을 fc의
        changeAmount()로 넘기고 전달 받은 반환 값이 true면
        "농산물 수량이 수정되었습니다.",
        false면 "농산물 수량 수정에 실패하였습니다. 다시 입력해주세요." 출력
}
```

```
    // 1-4. 농산물 목록 출력용 view 메소드
    public void printFarm() {
            fc의 printFarm()의 반환 값을 이용하여 keySet()을 통해
            "종류 : 이름(n개)" 형식으로 출력
     }
    // 2-1. 농산물 구매용 view 메소드
    public void buyFarm() {
            '1. 과일 / 2. 채소 / 3. 견과'를 통해 번호로 종류를 받고 구매할 농산물의 이름도
            받음. 이때 없는 번호를 선택하면 "잘못 입력하셨습니다. 다시 입력해주세요."가
            출력되며 다시 번호를 받고, 선택한 종류에 따라 생성되는 객체가 다름.
            객체 안에 종류와 이름을 저장. 데이터를 저장한 객체를 fc의 buyFarm()로 넘기고
            전달받은 반환 값이 true면 "구매에 성공하였습니다.",
            false면 "마트에 없는 물건이거나 수량이 없습니다. 다시 입력해주세요." 출력
    }
    // 2-2. 농산물 구매 취소용 view 메소드
    public void removeFarm() {
            '1. 과일 / 2. 채소 / 3. 견과'를 통해 번호로 종류를 받고 구매취소할 농산물의 이름도
            받음. 이때 없는 번호를 선택하면 "잘못 입력하셨습니다. 다시 입력해주세요."가
            출력되며 다시 번호를 받고, 선택한 종류에 따라 생성되는 객체가 다름.
            객체 안에 종류와 이름을 저장. 데이터 저장한 객체를 fc의 removeFarm()로 넘기고
            전달받은 반환 값이 true면 "구매 취소에 성공하였습니다.",
            false면 "구매매 목록에 존재하지 않습니다. 다시 입력해주세요." 출력
    }
    // 2-3. 구입한 농산물 출력용 view 메소드
    public void printBuyFarm() {
            fc의 printBuyFarm()의 반환 값을 이용하여 Iterator를 통해 출력
    }
}
```

```
public class FruitController{
	// 마트에서 농산물 저장용 HashMap 객체 생성(hMap)
	// 고객이 구매한 농산물 저장용 ArrayList 객체 생성(list)
	public boolean addNewKind(Farm f, int amount){
		// 전달 받은 f가 hMap 안에 key로 존재하지 않을 때
		// f와 amount를 각각 키와 값으로 저장 후 true 반환, 존재할 경우 false 반환.
	}
	public boolean removeKind(Farm f){
		// 전달 받은 f가 hMap 안에 key로 존재할 때 hMap에 f 삭제 후 true 반환
		// 존재하지 않을 경우 false 반환
	}
	public boolean changeAmount(Farm f, int amount){
		// 전달 받은 f가 hMap 안에 key로 존재할 때 f와 amount 저장 후 true 반환
		// 존재하지 않을 경우 false 반환
	}
	public HashMap<Farm, Integer> printFarm() {
		// 농산물 데이터가 들어가있는 컬렉션 반환
	}
	public boolean buyFarm(Farm f){
		// 전달 받은 f가 hMap 안에 존재하면서 그 f의 수량이 0개 이상일 때
		// list에 f 추가, 그리고 hMap에 f의 수량 1 감소 후 true 반환
		// 존재하지 않으면 false 반환
	}
	public boolean removeFarm(Farm f){
		// 전달 받은 f가 list에 존재할 때 list에 f 삭제, 그리고 hMap에 f 수량 1 증가
		// 위 경우일 때 true 반환, 아니면 false 반환
	}
	public ArrayList<Farm> printBuyFarm(){
		// 농산물 구매 데이터가 들어가있는 컬렉션 반환
	}
}
```

```
========== BS 마트 ==========
=====***** 메인 메뉴 *****=====
1. 직원 메뉴
2. 손님 메뉴
9. 종료

메뉴 번호 입력 : 1
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 1
이름 : 사과
수량 : 20
새 농산물이 추가되었습니다.
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 2
이름 : 양배추
수량 : 10
새 농산물이 추가되었습니다.
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 2
이름 : 무
수량 : 50
새 농산물이 추가되었습니다.
```

```
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 3
이름 : 땅콩
수량 : 50
새 농산물이 추가되었습니다.
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 1
이름 : 배
수량 : 30
새 농산물이 추가되었습니다.
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 1
이름 : 귤
수량 : 500
새 농산물이 추가되었습니다.
```

```
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 4
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(500개)
과일: 사과(20개)
채소: 무(50개)
채소: 양배추(10개)
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 2
1. 과일 / 2. 채소 / 3. 견과
삭제할 종류 번호 : 2
이름 : 무
농산물 삭제에 성공하였습니다.
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 4
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(500개)
과일: 사과(20개)
채소: 양배추(10개)
```

```
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 1
이름 : 사과
수량 : 80
새 농산물 추가에 실패하였습니다. 다시 입력해주세요.
1. 과일 / 2. 채소 / 3. 견과
추가할 종류 번호 : 3
이름 : 호두
수량 : 4
새 농산물이 추가되었습니다.
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 4
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(500개)
과일: 사과(20개)
채소: 양배추(10개)
견과: 호두(4개)
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 3
1. 과일 / 2. 채소 / 3. 견과
수정할 종류 번호 : 1
이름 : 사과
수정할 수량 : 80
농산물 수량이 수정되었습니다.
```

```
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 4
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(500개)
과일: 사과(80개)
채소: 양배추(10개)
견과: 호두(4개)
=====***** 직원 메뉴 *****=====
1. 새 농산물 추가
2. 종류 삭제
3. 수량 수정
4. 농산물 목록
9. 메인으로 돌아가기

메뉴 번호 입력 : 9
=====***** 메인 메뉴 *****=====
1. 직원 메뉴
2. 손님 메뉴
9. 종료

메뉴 번호 입력 : 2
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(500개)
과일: 사과(80개)
채소: 양배추(10개)
견과: 호두(4개)

=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
구매 종류 번호 : 1
구매할 것 : 귤
구매에 성공하였습니다.
```

```
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(499개)
과일: 사과(80개)
채소: 양배추(10개)
견과: 호두(4개)

=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
구매 종류 번호 : 2
구매할 것 : 양배추
구매에 성공하였습니다.
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(499개)
과일: 사과(80개)
채소: 양배추(9개)
견과: 호두(4개)

=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 1
1. 과일 / 2. 채소 / 3. 견과
구매 종류 번호 : 3
구매할 것 : 호두
구매에 성공하였습니다.
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(499개)
과일: 사과(80개)
채소: 양배추(9개)
견과: 호두(3개)
```

```
=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 3
과일: 귤
채소: 양배추
견과: 호두
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(499개)
과일: 사과(80개)
채소: 양배추(9개)
견과: 호두(3개)

=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 2
1. 과일 / 2. 채소 / 3. 견과
취소 종류 번호 : 2
구매 취소할 것 : 양배추
구매 취소에 성공하였습니다.
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(499개)
과일: 사과(80개)
채소: 양배추(10개)
견과: 호두(3개)

=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 3
과일: 귤
견과: 호두
```

```
현재 BS마트 농산물 수량
과일: 배(30개)
견과: 땅콩(50개)
과일: 귤(499개)
과일: 사과(80개)
채소: 양배추(10개)
견과: 호두(3개)

=====***** 고객 메뉴 *****=====
1. 농산물 사기
2. 농산물 빼기
3. 구입한 농산물 보기
9. 메인으로 돌아가기

메뉴 번호 입력 : 9
=====***** 메인 메뉴 *****=====
1. 직원 메뉴
2. 손님 메뉴
9. 종료

메뉴 번호 입력 : 9
프로그램 종료
```