**JAVA LAB**

20234014 컴퓨터 공학과 이은정

1. 주사위 굴리기

**package** dice;

**import** java.util.Scanner;

**class** Dice {

**private** **int** face;

**private** **int** num1;

**private** **int** num2;

**private** **int** trynum = 0;

**void** Dice() {

face = 0;

}

**void** setnum1(**int** n) {

num1 = n;

}

**int** getnum1() {

**return** num1;

}

**void** setnum2(**int** n) {

num2 = n;

}

**int** getnum2() {

**return** num2;

}

**void** settrynum(**int** n) {

trynum = n;

}

**int** gettrynum() {

**return** trynum;

}

**boolean** roll() {

num1 = (**int**) (Math.*random*() \* 6) + 1;

num2 = (**int**) (Math.*random*() \* 6) + 1;

System.***out***.println("주사위 1 =" + num1 + " 주사위2 = " + num2);

trynum += 1;

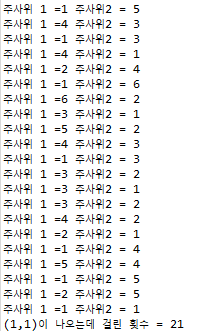
**if** (num1 + num2 == 2) {

**return** **true**;

}

**return** **false**;

}

}

**public** **class** dicegame {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Dice dice;

dice = **new** Dice();

**boolean** result = **false**;

**do** {

result = dice.roll();

**if** (result == **true**) {

System.out.println("(" + dice.getnum1() + "," + dice.getnum2() + ")이 나오는데 걸린 횟수 = " + dice.gettrynum());

**break**;

}

} **while** (**true**);

}

위 코드는 주사위 2개를 던지고, 두 개의 결과 값의 합이 2가 되면 코드를 종료하고, 몇 번 주사위를 던졌는지 반환하는 코드이다. 두 눈의 합이 2가 되려면 두 눈이 (1,1) 결과 값이 나와야만 한다. 코드 구조는 이러하다. MAIN에서 한 번 이상 주사위를 굴릴 수 있도록 do-while문을 통해 주사위를 굴린다. 주사위를 굴려서 나온 결과가 true라면 while문을 벗어나 모든 결과를 보여주고 코드를 끝낸다. 이 때 주사위를 굴리고, 결과를 도출하는 코드는 Dice class에서 실행한다. Dice 클래스는 눈의 수를 보여주는 num1, num2 그리고 시도 횟수를 보여주는 trynum 변수를 가지고 있다. 이제 주사위를 굴려보겠다. 주사위 눈은 1~6이다. 이 값은 랜덤하게 나와야 한다. 따라서 (int)(Math.random()\*6)+1라고 코드를 작성하여서 0~1 사이의 무작위의 수에 6을 곱한 값에 int 처리를 해주고 1을 더해 주어 0~5에서 1~6로 수를 조정한다. 만약 두 눈의 합이 2라면 true를 반환하고 아니라면 false를 반환한다. 이 값은 main으로 반환되는데 main에서는 이 값이 true인지 false인지를 검사한다. 만약 반환 값이 true라면 while문을 빠져 나오고 각각의 눈의 값과 trynum을 출력한다. 이 때 눈의 값과 은 눈의 값, trynum은 클래스를 생성한 이후 getnum1, getnum2과 같은 접근자를 통해 값을 가져와야만 한다. 따라서 접근자를 통해 값을 가져오고 난 이후에 이를 출력하면 위와 같은 결과 값을 얻을 수 있다.

2~3. 강아지

**import** java.util.Scanner;

**class** Dog {

**private** String name;

**private** **int** age;

**private** String color;

**static** **int** *num*;

**public** Dog(String name, **int** age, String color) {

**super**();

**this**.name = name;

**this**.age = age;

**this**.color = color;

*num*++;

}

**void** setname(String na) {

name = na;

}

String getname() {

**return** name;

}

**void** setage(**int** ag) {

age = ag;

}

**int** getage() {

**return** age;

}

**void** setcolor(String col) {

color = col;

}

String getcolor() {

**return** color;

}

**void** num() {

System.***out***.println("현재까지 생성된 강아지의 수: " + *num*);

}

}

**class** Witch {

**void** younger(Dog obj) {

obj.setage(obj.getage() - 5);

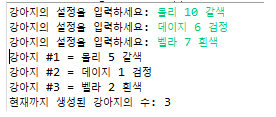
}

}

**public** **class** dognum {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

 String name, color;

**int** age;

Dog dog1, dog2, dog3;

Witch witch;

witch = **new** Witch();

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("강아지의 설정을 입력하세요: ");

name = sc.next();

age = sc.nextInt();

color = sc.next();

dog1 = **new** Dog(name, age, color);

System.***out***.print("강아지의 설정을 입력하세요: ");

name = sc.next();

age = sc.nextInt();

color = sc.next();

dog2 = **new** Dog(name, age, color);

System.***out***.print("강아지의 설정을 입력하세요: ");

name = sc.next();

age = sc.nextInt();

color = sc.next();

dog3 = **new** Dog(name, age, color);

witch.younger(dog1);

witch.younger(dog2);

witch.younger(dog3);

System.***out***.println("강아지 #1 = " + dog1.getname() + " " + dog1.getage() + " " + dog1.getcolor());

System.***out***.println("강아지 #2 = " + dog2.getname() + " " + dog2.getage() + " " + dog2.getcolor());

System.***out***.println("강아지 #3 = " + dog3.getname() + " " + dog3.getage() + " " + dog3.getcolor());

dog1.num();

}

}

2번 3번 문제를 한 코드로 만들어 서술했다. 위 코드는 강아지를 나타내는 클래스 Dog를 만들어 값을 저장하고, 강아지 클래스가 몇 개 생겼는지 출력하는 코드이다. 이 코드는 main에서 매개 변수를 받아 그 때 그 때마다 클래스를 생성함으로써 작동한다. 클래스 매개변수로 name[String], age[int], color[String] 를 넘겨준다. 이렇게 코드를 작성할 시 생성자에 의해 이 값이 초기값으로 해당 클래스의 필드에 저장이 된다. 이후에 Witch 클래스의 younger 메서드를 통해 클래스의 age 값을 조정해야 한다. 이 클래스의 메서드는 매개변수로 클래스를 받는다. 클래스를 넘겨줄 시 클래스의 필드, 메서드를 사용할 수 있다. Witch의 younger 메서드는 Dog 클래스의 age를 받아와 그 값을 5 작게 만들어 원본 값을 변경해야 한다. 이 경우에 클래스를 매개변수로 전달해주면 해당 클래스의 메서드를 자유롭게 쓸 수 있게 된다. 따라서 younger 메서드로 Dog 클래스를 전달해주게 되면 Dog의 클래스의 접근자를 쓸 수 있게 된다. 따라서 해당 Dog 클래스를 사용하는 인스턴스를 받았다면 getage()를 통해 나이를 얻어오고 그 값을 -5 한 다음 setage를 통해 원본 값을 변경해준다. 객체를 매개변수로 전달할 시 해당 객체의 주소가 넘어가기 때문에 메서드를 통해 필드 값을 변경할 시 원본의 값도 바뀐다. Num 여러 객체가 공유하는 하나의 변수이다. 따라서 static 형식으로 써야만 한다. 따라서 아무 객체에서 num을 불러와 출력할 시 해당 변수의 값을 출력할 수 있다.

4. 학생 정보 저장하기

**package** Studioinfo;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**class** Student {

**private** String name;

**private** String grade;

**static** **int** *num*;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Student(String name, String grade) {

**super**();

**this**.name = name;

**this**.grade = grade;

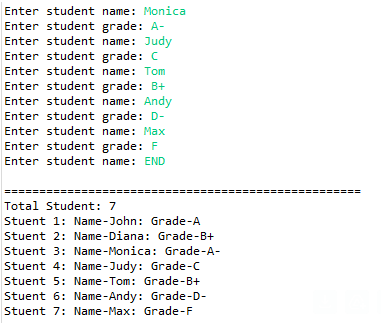
**this**.*num*++;

}

**public** String getGrade() {

**return** grade;

}

 **public** **void** setGrade(String grade) {

**this**.grade = grade;

}

**public** **int** getNum() {

**return** *num*;

}

**public** **void** setNum(**int** num) {

**this**.*num* = num;

}

}

**class** printinfo {

**void** display\_student\_info(ArrayList<Student> students) {

System.***out***.println("\n===================================================");

System.***out***.println("Total Student: " + Student.*num*);

**for** (**int** i = 0; i < students.size(); i++) {

Student currstudent = students.get(i);

System.***out***.println(

"Stuent " + (i + 1) + ": Name-" + currstudent.getName() + ": Grade-" + currstudent.getGrade());

}

}

**public** **class** Studioinfo {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

String name, grade;

Scanner sc = **new** Scanner(System.***in***);

ArrayList<Student> students = **new** ArrayList<>();

printinfo printstu;

printstu = **new** printinfo();

**do** {

System.***out***.print("Enter student name: ");

name = sc.next();

**if** (name.equals("END"))

**break**;

System.***out***.print("Enter student grade: ");

grade = sc.next();

students.add(**new** Student(name, grade));

} **while** (**true**);

printstu.display\_student\_info(students);

}

}

}

해당 코드는 학생들의 정보를 END를 입력하기 전까지 저장하고, 해당 학생들의 정보를 저장한다. END를 입력하기 전까지 값을 계속 저장해야 하기 때문에 ListArray를 사용한다. main에서 ListArray를 하나 만든다. 이후에 name과 grade 값을 받아 ListArray에 저장하는 것을 반복 하는데 만약 name으로 받은 값이 END라면 반복을 종료한다. 이후에 학생들의 값을 전부 출력하면 해당 코드는 끝이 난다. ListArray는 클래스에 의해 만들어진 인스턴스를 저장해야 하기 때문에 ArrayList<>에서 <>안에 클래스 명을 넣어주어야 한다. 우리가 저장할 클래스는 Student 이므로 ArrayList<Student> students = new ArrayList<>(); 코드를 통해 리스트를 생성한다. ArrayList에 값을 추가 할 때도 <Student> 클래스를 넣을 것이라고 미리 적어두었기 때문에 입력 값도 클래스를 넣어주어야 한다. 따라서 add()안에 new Student(name, age)를 만들어 새로운 객체를 만듦과 동시에 이를 리스트 안에 넣어버린다. 이렇게 반복을 하다가 name에 END를 입력 받으면 해당 반복문을 종료한다. 이후에 리스트에 저장해둔 값을 메서드를 통해 모두 출력하게 된다. 이 때 이 메서드는 외부 클래스에 저장되어 있는 메서드이다. 따라서 리스트의 주소를 매개변수로 보내 해당 리스트의 값을 사용하도록 한다. for문을 사용해 리스트의 크기만큼 반복하며 리스트 값을 출력한다. 리스트 값에는 클래스가 저장되어 있다. 따라서 현재의 값을 출력하기 위해서는 클래스를 임의로 불러와, 그 값을 출력하는 식으로 for문을 돌려야 한다. Students arraylist에서 i번째에 저장된 클래스를 get을 통해 새로운 클래스 변수에 저장한다. 그 클래스 변수는 역시나 Student 클래스이기 때문에 Student 클래스의 메서드들을 사용할 수 있다. 따라서 클래스 변수에서 접근자를 사용해 값을 불러와 출력하면 위와 같은 결과를 얻을 수 있다.

4. 도서관 관리

**package** library;

**import** java.util.ArrayList;

**import** java.util.Scanner;

//Book 클래스: 도서 정보를 나타내는 클래스

**class** Book {

**private** String title;

**private** String author;

**private** String publisher;

**private** **int** price;

**public** Book(String title, String author, String publisher, **int** price) {

**super**();

**this**.title = title;

**this**.author = author;

**this**.publisher = publisher;

**this**.price = price;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getAuthor() {

**return** author;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

**public** String getPublisher() {

**return** publisher;

}

**public** **void** setPublisher(String publisher) {

**this**.publisher = publisher;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

}

// Library 클래스: 도서를 관리하는 클래스

**class** Library {

**private** ArrayList<Book> books;

// 생성자

**public** **void** BookManagement() {

**this**.books = **new** ArrayList<>();

}

// 도서 추가

**public** **void** addBook(Book book) {

books.add(book);

}

// 도서 삭제

**public** **boolean** removeBook(String title) {

**boolean** presence = **false**;

**for** (**int** i = 0; i < books.size(); i++) {

**if** (books.get(i).getTitle().equals(title)) {

System.***out***.println("도서 삭제: " + books.get(i).getTitle());

books.remove(i);

presence = **true**;

**return** presence;

}

}

**return** presence;

}

// 도서 검색

**public** **void** searchBook(String title) {

**for** (Book book : books) {

**if** (book.getTitle().equals(title) == **true**) {

System.***out***.println("\n==================================================");

System.***out***.println("도서 정보를 출력합니다.\n");

System.***out***.println("도서명: " + book.getTitle());

System.***out***.println("저자명: " + book.getAuthor());

System.***out***.println("출판사명: " + book.getPublisher());

System.***out***.println("가격: " + book.getPrice());

} **else**

System.***out***.println("도서가 존재하지 않습니다\n");

}

}

}

**public** **class** LibraryManage {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** input;

String name;

String author;

String publisher;

**int** price;

Book book;

Library library;

library = **new** Library();

library.BookManagement();

Scanner sc = **new** Scanner(System.***in***);

**do** {

System.***out***.println("\n순천향 도서관 서비스에 접속합니다. 원하는 서비스를 선택해 주십시오");

System.***out***.println("1.도서 추가 2.도서 삭제 3.도서 검색 4.서비스 종료\n");

System.***out***.println("==================================================");

System.***out***.print("입력: ");

input = sc.nextInt();

**if** (input == 1) {

System.***out***.println("\n추가할 도서의 정보를 입력해 주십시오");

System.***out***.print("도서명: ");

name = sc.next();

System.***out***.print("저자명: ");

author = sc.next();

System.***out***.print("출판사명: ");

publisher = sc.next();

System.***out***.print("가격: ");

price = sc.nextInt();

sc.nextLine(); // 버퍼 비우기

library.addBook(**new** Book(name, author, publisher, price));

System.***out***.println("\n==================================================");

System.***out***.println("도서가 정상적으로 등록되었습니다.");

}

**else** **if** (input == 2) {

**boolean** result;

System.***out***.println("\n삭제할 도서의 정보를 입력해 주십시오");

System.***out***.print("도서명: ");

name = sc.next();

result = library.removeBook(name);

**if** (result == **false**) {

System.***out***.println("해당 도서가 존재하지 않습니다.");

} **else** {

System.***out***.println("\n==================================================");

System.***out***.println("도서가 정상적으로 삭제되었습니다.");

}

}

**else** **if** (input == 3) {

System.***out***.println("\n검색할 도서의 정보를 입력해 주십시오");

System.***out***.print("도서명: ");

name = sc.next();

library.searchBook(name);

}

**else** {

System.***out***.println("\n==================================================");

System.***out***.println("\n서비스를 이용해주셔서 감사합니다. 서비스를 종료합니다.");

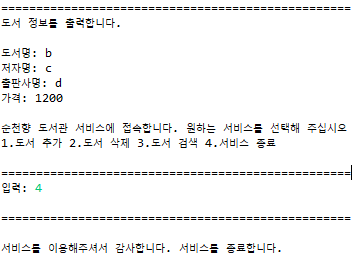
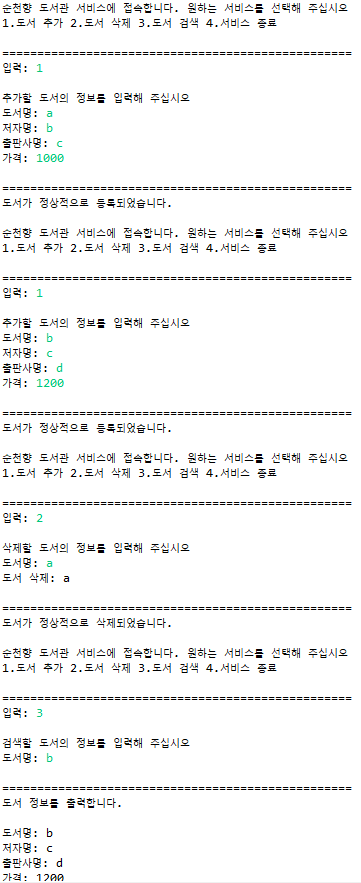
**break**;

}

} **while** (**true**);

}

}



위 코드는 도서관 시스템을 java로 표현한 것이다. main에서는 내가 어떤 행동을 할 것인지 정하기만 하고, 행동을 실제로 하는 것은 메서드로 매개변수를 전달하면서 동작을 실행한다. 먼저 Book 클래스는 책에 대한 정보를 저장한다. 제목, 저자, 출판사, 가격 변수를 접근자와 설정자를 통해 값을 저장하고 반환한다. 그 다음 실제로 도서관에서 하는 일들을 행하는 곳은 Library 클래스이다. 이 곳에서는 책을 저장할 리스트를 만들고, 도서를 추가하고 삭제하거나 검색하는 등의 일을 한다. 먼저 최소 한 번 이상의 동작을 해야 하므로 do-while문을 사용해 반복해준다. 그리고 우리가 해야 할 일이 무엇인지 숫자로 입력 받는다. 만약 우리가 입력 받은 값이 1이라면 도서를 추가 해야 한다. ArrayList에 저장을 해야 하므로 main에서 객체를 만들어 넘겨주는 방식으로 책을 추가한다. 도서명, 저자명, 출판사명, 가격을 전부 입력 받은 후 libraray.addBook을 통해 책을 추가한다. 이 때 넘겨주는 객체는 Book 클래스에 의해 만들어진 새 객체이다. 새 객체는 생성자에 의해 이름, 저자명, 출판사명, 가격을 넘겨 받아 저장한다. 따라서 코드는 library.addBook(**new** Book(name, author, publisher, price)); 라고 만들어진다. 제대로 코드가 실행되었다면 이를 안내하는 문장을 출력한다. 도서를 삭제하고 검색하는 것은 비슷하다. 우선 두 개 모두 리스트의 모든 값을 조사해보아야 한다는 공통점을 가진다. for문을 사용해 ArrayList의 크기만큼 검사를 진행한다. 이 때 책이 저장된 리스트인 books.get(i)를 통해 값을 가져올 수도 있고 Book book : books 를 통해 each문을 사용할 수도 있다. 어쨌든 Book 클래스로 만들어진 인스턴스를 가져와 검사를 한다. 만약 .getTitme() 접근자를 통해 얻어온 제목과 내가 찾는 도서가 같다면 각각의 행동을 취하면 된다. 만약 없을 시 도서가 존재하지 않는다고 안내한다. 마지막으로 4를 누르면 while문을 벗어나게 만들어 위 과정을 종료한다.

5. 쇼핑

**package** library;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**class** Item {

**private** String title;

**private** **int** price;

**public** Item(String title, **int** price) {

**super**();

**this**.title = title;

**this**.price = price;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

}

**class** ShoppingCart {

**private** ArrayList<Item> items;

// 생성자

**public** **void** BookManagement() {

**this**.items = **new** ArrayList<>();

}

// 물품 추가

**public** **void** addItem(Item item) {

items.add(item);

}

// 물품 삭제

**public** **boolean** removeItem(String title) {

**boolean** presence = **false**;

**for** (**int** i = 0; i < items.size(); i++) {

**if** (items.get(i).getTitle().equals(title)) {

System.***out***.println("도서 삭제: " + items.get(i).getTitle());

items.remove(i);

presence = **true**;

**return** presence;

}

}

**return** presence;

}

**public** **int** calprice() {

**int** price = 0;

**for** (**int** i = 0; i < items.size(); i++) {

price += items.get(i).getPrice();

}

**return** price;

}

**public** **void** deleteItem() {

items.clear();

}

}

**public** **class** Shopping {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** input;

String name;

**int** price;

Item item;

ShoppingCart shoppingcart;

shoppingcart = **new** ShoppingCart();

shoppingcart.BookManagement();

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("순천향 쇼핑센터입니다.");

**do** {

System.***out***.println("원하는 서비스를 선택해 주십시오");

System.***out***.println("1.상품 추가 2.상품 삭제 3.총 가격 계산 4.장바구니 비우기 5.서비스 종료\n");

System.***out***.println("==================================================");

System.***out***.print("입력: ");

input = sc.nextInt();

**if** (input == 1) {

System.***out***.println("\n추가할 상품의 정보를 입력해 주십시오");

System.***out***.print("품명: ");

name = sc.next();

System.***out***.print("가격: ");

price = sc.nextInt();

sc.nextLine(); // 버퍼 비우기

shoppingcart.addItem(**new** Item(name, price));

System.***out***.println("\n==================================================");

System.***out***.println("물품이 장바구니에 담겼습니다.");

}

**else** **if** (input == 2) {

**boolean** result;

System.***out***.println("\n삭제할 도서의 정보를 입력해 주십시오");

System.***out***.print("품명: ");

name = sc.next();

result = shoppingcart.removeItem(name);

**if** (result == **false**) {

System.***out***.println("해당 물품이 존재하지 않습니다.");

} **else** {

System.***out***.println("\n==================================================");

System.***out***.println("물품이 정상적으로 삭제되었습니다.\n");

}

}

**else** **if** (input == 3) {

price = shoppingcart.calprice();

System.***out***.println("\n총 물건의 가격은 " + price + "입니다.\n");

}

**else** **if** (input == 4) {

shoppingcart.deleteItem();

System.***out***.println("장바구니를 전부 비웠습니다.\n");

}

**else** {

System.***out***.println("\n==================================================");

System.***out***.println("\n서비스를 이용해주셔서 감사합니다. 서비스를 종료합니다.");

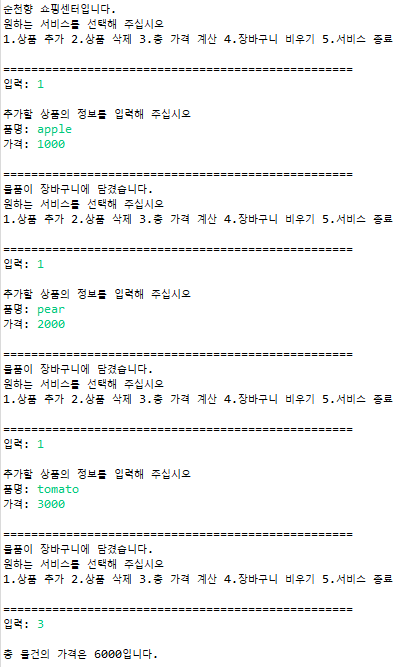
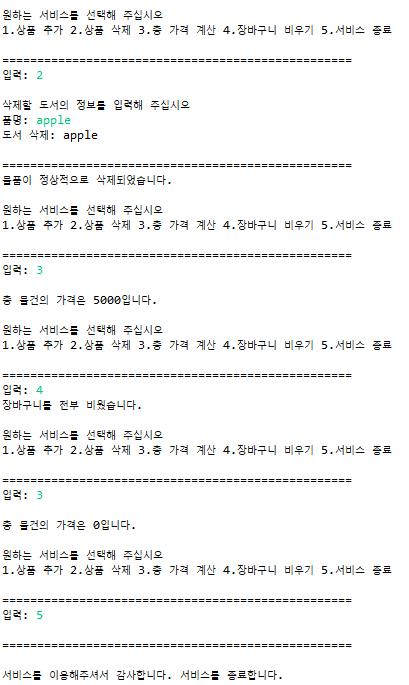
**break**;

}

} **while** (**true**);

}

}



쇼핑은 위에서 작성한 도서관과 거의 비슷한 코드 양식을 가진다. 내가 장바구니에 담는 모든 물건은 ArrayList에저장되며 이 ArrayList에는 Item 클래스에 의해 만들어진 객체가 담긴다. main에서는 내가 어떤 행동을 할 것인지 정하기만 하고, 행동을 실제로 하는 것은 메서드로 매개변수를 전달하면서 동작을 실행한다. 한 번 이상은 동작을 실행해야 하므로 do-while(true)를 사용한다. 내가 무슨 행동을 하려는지 input 값을 통해 값을 받는다. 5번을 누르면 서비스 종료를 해야 하기 때문에 break문을 통해 while문을 벗어나고 코드를 종료한다. 만약 내가 하려는 행동이 상품 추가라면, 품명과 가격을 받는다. 내가 몇 개의 상품을 ArrayList에 저장할 지 모르므로 매개변수 입력 창에서 새 객체를 만들어 넘긴다. 우리가 필요한 속성은 이름과 가격이므로, 위 값들만 입력 받아 이를 Item의 생성자로 넘긴다. Item은 그 즉시 이 값들을 본인 속성값으로 저장한 후 새 객체를 만든다. 그리고 addItem 메서드에 의해 이 객체는 ArrayList에 저장된다. 그리고 2~4번 선택지에 의한 동작은 모두 이 저장 값을 이용해 이루어 진다. Item을 통해 만들어진 값들은 ShoppingCart 클래스에서 사용된다. 2번 상품 삭제를 누를 시, 이름을 main에서 입력 받는다. 그리고 이름을 ShoppingCart의 removeItem의 매개변수로 넘긴다. 해당 매개변수가 넘어오면, 함수에서는 ArrayList에 저장된 인스턴스에 해당 이름을 속성으로 가진 인스턴스가 존재하는지 검사한다. items.get(i).getTitle().equals(title)를 통해 items라는 이름을 가진 ArrayList에서 각각의 인스턴스를 불러와 getTitle을 통해 그 인스턴스의 이름을 가져와 매개변수로 받은 제목과 같은지 검사를 한다. 검사 결과가 같다면 해당 물품이 존재하는 것이므로 remove 메서드를 사용해 그 물품을 제거한다. 3번째로 가격을 계산하는 메서드는 2번과 비슷하다. 모든 인스턴스의 price를 가져와 이를 메서드 내의 price 변수에 더한다. 최종적으로는 이 값을 return해서 총 가격을 계산한다. 4번은 모든 장바구니 물품을 지우는 문제이다. 따라서 clear()메서드를 통해 ArrayList를 모두 비운다.