(디지털컨버전스)스마트 콘텐츠와 웹 융합응용SW개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 –Joongyeon Kim(김중연)
jjjr69@naver.com

2021년 5월 27일 질문노트

[김중연]

```java
public class NumForeachTest {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 5, 6, 7, 8, 9};
        int num;

        // 위와 같이 작성하면 될 수도 있을것 같으나
        // 실질적으로는 foreach에서 사용하는 변수는
        // for 문 내부에서만 처리해야하기 때문에 외부에서 사용할 수 없습니다!
        /*
        for (num : arr) {
            System.out.println("나오나요 ? " + num);
        }
         */
    }
}

// 값은 복제, 객체는 원본,
// String str = "하이"
// == 값을 비교(같은지)
```

```java
import java.util.ArrayList;
import java.util.Arrays;

public class ArrayListPrintTest {
    public static void main(String[] args) {
        String[] names = {"안녕", "하이", "헬로"};
        ArrayList<String> nameLists = new ArrayList<~>(Arrays.asList(names));

        // 방법 1
        System.out.println(nameLists);

        // 방법 2
        for (String name : nameLists) {
            System.out.println(name);
        }
    }
}

// Integer와 int의 핵심적인 차이점
// 향후 Spring(Java)과 Vue(JavaScript)간에 통신을 수행한다고 하면
// 반드시 객체가 전달되어야 하며 값은 전달이 안된다는 문제가 있다.
// 나중에 Spring으로 통신을 수행하게 된다면 값이 아닌 객체를 전달하자
```

```java
/.../


import java.util.ArrayList;
import java.util.Scanner;

class Market {
    private ArrayList<String> userBuyList;
    private ArrayList<Integer> userBuyListStock;

    private String[] marketSellList = {"선풍기", "키보드", "마우스", "모니터"};
    private int[] marketSellListPrice = {380000, 80000, 70000, 400000};  //어차피 중재자 역할하는 메소드랑 생성자 빼고 전부 프라이빗설정 되어있다

    private int myMoney=100;  //결제를 위해 수를 대입함

    private Boolean continueShopping;
    Scanner scan;

    final int DEFAULT_IDX = 1;
    int price;

    public Market () {
        userBuyList = new ArrayList<String>();   //구매한 물건
        userBuyListStock = new ArrayList<Integer>();  //물건을 구매한 수량

        continueShopping = false;  //쇼핑중단을 위해 만든 변수

        scan = new Scanner(System.in);
    }


```

```java
public void doShopping () {
    do {
        // 1. 마켓에서 판매하는 물품을 보여줌
        showMarketSellList();
        // 2. 구매할 물건을 선택하세요.
        //    어떻게 선택하지 ?
        //    키보드 입력이나 뭔가가 필요한거 같네 ?
        //    가만 보니 초기에 Scanner를 안만들었구나 추가!
        selectBuyItem();
        // 3. 구매리스트가 작성되었다면 비용 산정 진행
        doPayment();
        // 4. 계속 구매할 것인지 여부 판단
        checkContinueShopping();
        //5. 결제가 가능한지 불가능한지 알아야함
        payMoney();
        //6. 결제를 입력하면 메뉴 화면으로 돌아와야하지만 Y를 한번 더 눌러야 정상적으로 돌아간다(버그?)
        checkPayMoney();
    } while (continueShopping);
}

private void checkContinueShopping () {
    Boolean isOK = false;

    do {
        System.out.print("쇼핑을 계속하시겠습니까 ? Y/N");

        String res = scan.nextLine();


        if (res.equals("Y")) {
            isOK = false;
            continueShopping = true;
        } else if (res.equals("N")) {
            isOK = false;
            continueShopping = false;
        } else {
            isOK = true;
            continue;
        }
    } while (isOK);

}

private void doPayment () {  //총 결제금액을 구하는 메소드는 만들었다
    // userBuyList, userBuyListStock에
    // 구매 물품과 구매 물량이 기록되어 있음
    // 물건 가격 정보는 marketSellList, marketSellListPrice 를 통해 알 수 있음
    // userBuyList와 userBuyListStock을 활용하여 어떤 물건을 몇 개 구하는지 체크하고 이것을 이용해
    // 그리고 지갑에다가 적용한다.(지갑 설정기능이 빠져있음 현재)

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

        if(userBuyList.get(i).equals("선풍기")){  //이퀄스를 각 문자열이 맞으면 가격이 더해지게끔 만듬
            price += userBuyListStock.get(i) * marketSellListPrice[0];
        }else if(userBuyList.get(i).equals("키보드")){
            price += userBuyListStock.get(i) * marketSellListPrice[1];
        }else if(userBuyList.get(i).equals("마우스")){
            price += userBuyListStock.get(i) * marketSellListPrice[2];
        }else if(userBuyList.get(i).equals("모니터")){
            price += userBuyListStock.get(i) * marketSellListPrice[3];
```

```
    }
    System.out.printf("총 결제 금액은 %s원 입니다.\n", price);

}

private void payMoney(){ //현금 액수에 따라 결제여부가 결정되는 메소드도 만들었고...
    System.out.printf("지갑에 있는 현금 : %d, 당신이 지불해야할 현금 : %d\n",myMoney, price);
    System.out.printf("잔액 %d\n", (myMoney-price));
    int totalAmount= myMoney-price;
    if(totalAmount<=0){
        System.out.println("결제가 불가능합니다!");
    }else{
        System.out.println("결제가 완료되었습니다");
    }
}

private void checkPayMoney(){  //결제의 진행여부를 확인하는 메소드
    Boolean payEnd = true;
    String st=scan.nextLine();

    do{
        System.out.print("그럼 결제 진행하겠습니까?");

        if (st.equals("Y")){
            payEnd = false;

        } if(st.equals("N")){
            payEnd = false;
```

```
            continueShopping = true;
        } else {
            continue;
        }

    }while(payEnd);
}

private void selectBuyItemStock (String selectItem) {
    Boolean isntErrorAmount = true;
    int amount;

    do {
        System.out.print("구매할 수량을 선택하세요: ");

        amount = scan.nextInt();

        if (amount <= 0) {
            System.out.println("잘못된 수량이니 다시 입력해주세요!");
            continue;
        }

        isntErrorAmount = false;
    } while (isntErrorAmount);

    createNonDuplicateBuyList(selectItem, amount);
}
```

```java
// 어떤 물건을 구할지 결정하는 매서드
private void selectBuyItem () {
    Boolean continueBuying = true;

    do {
        System.out.print("구매할 물건의 번호를 누르세요(결제진행: 0): ");

        int itemNum = scan.nextInt();

        if (itemNum > 4) {
            System.out.println("잘못된 물품을 선택하셨습니다!");
            continue;
        } else if (itemNum < 0) {
            System.out.println("잘못된 물품을 선택하셨습니다!");
            continue;
        } else if (itemNum == 0) {
            continueBuying = false;
            continue;
        }

        // 실제 물건의 구매 수량을 결정하기 전에 해당 물품을 구매하므로 ArrayList 설정이 필요하다.
        // 이제 해당 작업을 여기에 추가해봅시다 ~

        // 현재 케이스에서는 중복에 대한 대처가 진행되지 않고 있음
        // 그러므로 중복을 감지하여 리스팅을 할 수 있는 매서드를 만들 필요가 있다!
        //userBuyList.add(marketSellList[itemNum - DEFAULT_IDX]);
        //System.out.println(userBuyList);

        // 현재 createNonDuplicateBuyList()도 stock을 처리하고
        // 아래쪽의 selectBuyItemStock()도 stock을 처리한다.
        // 이렇게 혼동이 발생하는 경우에는 누가 더 우선권을 가져야 하는지 분석이 필요하다.
        // cNDBL(줄여서)은 실제 물건의 구매에 있어서 중복이 있는지 검사한다.
        // createNonDuplicateBuyList(marketSellList[itemNum - DEFAULT_IDX]);

        // 물품을 모두 선택하고 몇 개 구할지 결정하는 매서드
        selectBuyItemStock(marketSellList[itemNum - DEFAULT_IDX]);

        System.out.println(userBuyList);
        System.out.println(userBuyListStock);
    } while (continueBuying);
}

private void createNonDuplicateBuyList (String target, int amount) {
    // 실제 중복이 되었다면 인덱스 값이 나올 것이고
    // 중복이 없으면 -1이 나오게 될 것이다.
    int idx = userBuyList.indexOf(target);

    if (idx == -1) {    // 중복 없음
        userBuyList.add(target);
        userBuyListStock.add(amount);
    } else {            // idx가 중복된 요소를 알려줌
        // set(idx, 데이터)는 특정 인덱스의 값을 update(갱신)함
        // add(idx, 데이터) + remove(idx + 1)과 동일한 역할을 함
        userBuyListStock.set(idx, userBuyListStock.get(idx) + amount);
```

```
        }
    }

    private void showMarketSellList () {
        int length = marketSellList.length;

        System.out.println("우리 마켓에서 판매하는 물품을 리스팅 합니다!");

        for (int i = 0; i < length; i++) {
            System.out.printf("%d. %s: %d\n", i + 1, marketSellList[i], marketSellListPrice[i]);
        }
    }
}

public class Prob51 {
    public static void main(String[] args) {
        Market m = new Market();

        m.doShopping();
    }
}
```