UI/UX 전문가 과정비트캠프

2021년 01월 14일 13회차

장 해 솔

이메일 : wkdgothf@gmail.com

github : https://github.com/legossol

목표

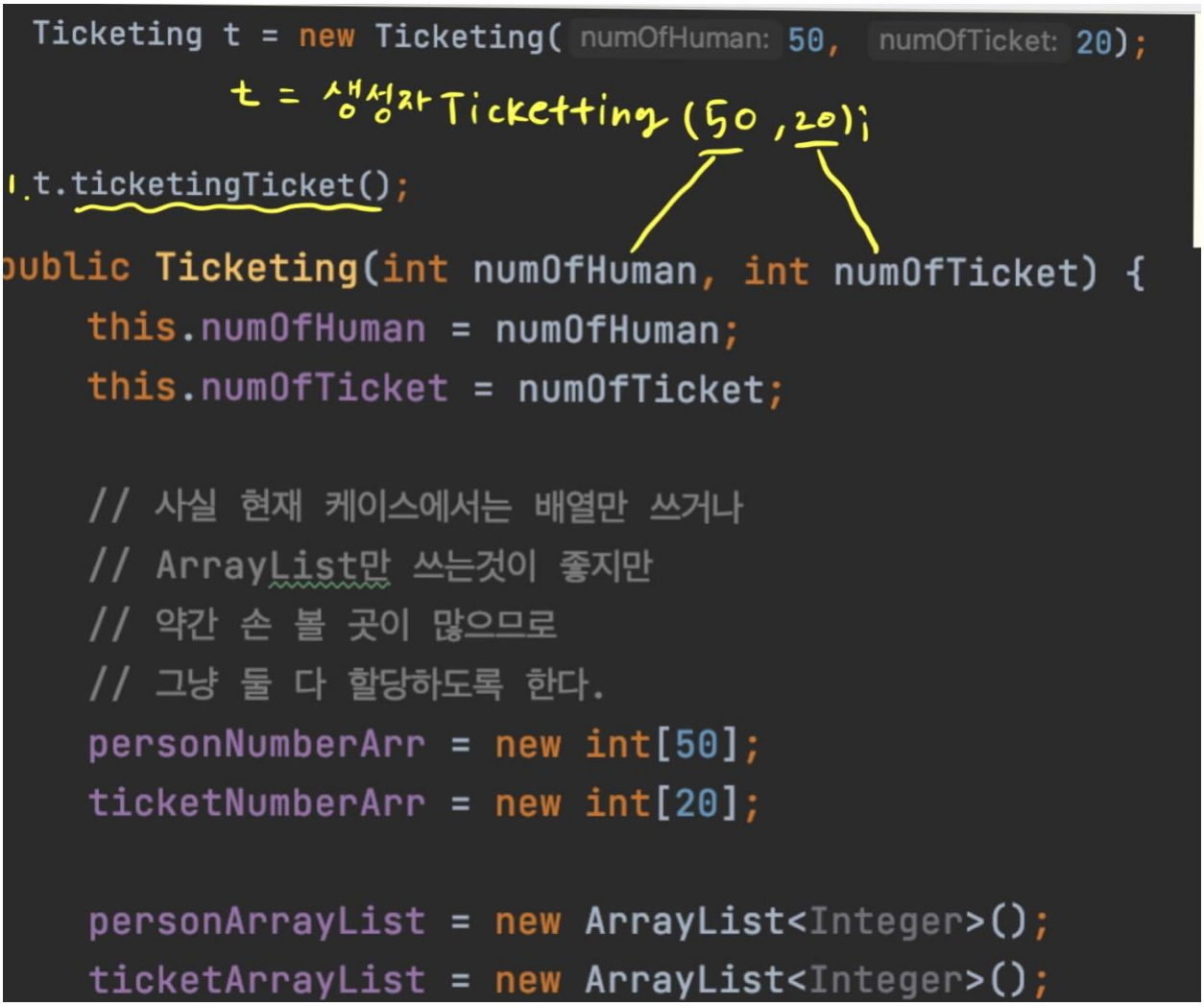
1. 오늘 배운 것들
    - 개인적인 정리 및 해결
    - Arraylist.
    - HashSet.
    - Iterator.
2. 질문 노트 작성
    - Thirrteenth() 질문

*상속자

:

*개인적인 정리 및 해결

- Ticketing문제에서 나온 50명의 사람들을 20개의 티켓에 뽑는 문제이다



우선 생성자를 입력받았으니 ticketingTicket매서드로 가보자

```
public void ticketingTicket() {
    int personNum;

    for(int i = 0; i < numOfTicket; i++) {
        1.personNum = allocRandomPersonNumber();
        2.allocTicket(personNum);
    }
}
```

```
public int allocRandomPersonNumber() {
    boolean isDup = false;
    int randNum;

    do {
        randNum = (int) (Math.random() * 50);

        if (personNumberArr[randNum] != 0) {
            isDup = true;
        } else {
            isDup = false;
            personNumberArr[randNum] = 1;
        }
    } while(isDup);

    return randNum;
```

ticketing매서드를 보면 티켓의 갯수 즉 Maximum 20개(0~19)까지 for문을 돌릴거라는것을 말하고잇다. 첫번째 사람들을 20명 뽑기위해 allocRandompersonNumber을 해야한다.

근데 당최 그 식이 어떻게 동작하는지 이해가 안된다.

그렇기에 이렇식으로 코드를 변경 하여 동작시켜봤다.(오른쪽은 결과값)

```
public int allocRandomPersonNumber() {
    boolean isDup = false;
    int randNum;

    do {
        randNum = (int) (Math.random() * 50);
        System.out.println(randNum);
        System.out.println("input");

        if (personNumberArr[randNum] != 0 ) {

            isDup = true;
            System.out.println("ifififififif");
        } else {
            personNumberArr[randNum] = 1;
            System.out.println(randNum);
            isDup = false;
            System.out.println("elseelseelse");
        }
        personNumberArr[randNum] = 1;
        System.out.println("outoutout");
    } while(isDup);
    System.out.println("return");
    return randNum;
```

```
23
input
23
elseelseelse
outoutout
return
44
input
44
elseelseelse
outoutout
return
49
input
49
elseelseelse
outoutout
return
49
input
ifififififif
outoutout
```

첫 시작은 이랬다. isDup을 false로 한 이유는 모르겠고 일단 숫자를 따라가 보자 23이라는 번호가 랜덤으로 뽑혀서 if문이 아닌 else문으로 갔다(왜 if 문이 아닌 else문으로 갔는지는 생각하지 말아보자) 23이 personNumberArr[23]에 대입되어 =1과 같다는 판정을 빤았다. 그러므로 isDup=false가 되었고 else문을 나온뒤에도 이 personNumberArr[23]=1과 같다는 판정을 얻고 return되어 randNum이 되어 나갔다.

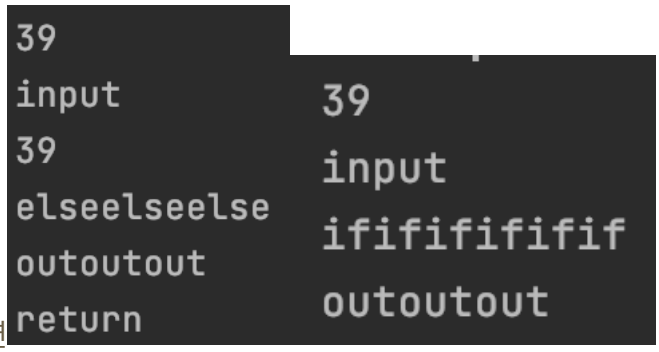
이렇게 생각을 끈낸뒤 결과표를 자세히 보니 여러개의 같은 숫자가 있는것이 보였다.

바로 49결과표 중간부터 보면 49는 3번 등장하고 else문에서도 있고 if 문에서도 있는 것이 보인다 그럼 이걸 봐야겠구나 생각하고 49를 따라가보자

기존 23과 동일하게 처음으로 뽑힌 49는 else문을 거쳐 리턴 randNum이 되어 나갔다. 그 뒤 49가 또 랜덤값으로 뽑혔는데 이번엔 else문이 아닌 if 문으로 들어갔다 그렇다면 여기서 유추해 낼 수 있다.

if를 돈다는 것은 이미 나온 값이 또 나왔다는 것을 의미하고

else문을 돈다는 것은 처음 나온 랜덤값이라는 거구나



확인을 한번 해보면 이렇게 다른 결과들에서 먼저나온 39는 else를 돌고 나중에 39가 나왔을 때는 if문을 돌았다. 이것이 맞다.

자 그럼이제 **if문의 =1과 else문의 =0**은 무엇인가.

나는 이렇게 이해했다. personNumberArr[randNum]은 항렬의 형태이다 즉 공간을 가지고 있다는 뜻인데 공간에 무엇인가 들어 있습니까? 없습니까?이렇게 이해를 하였다

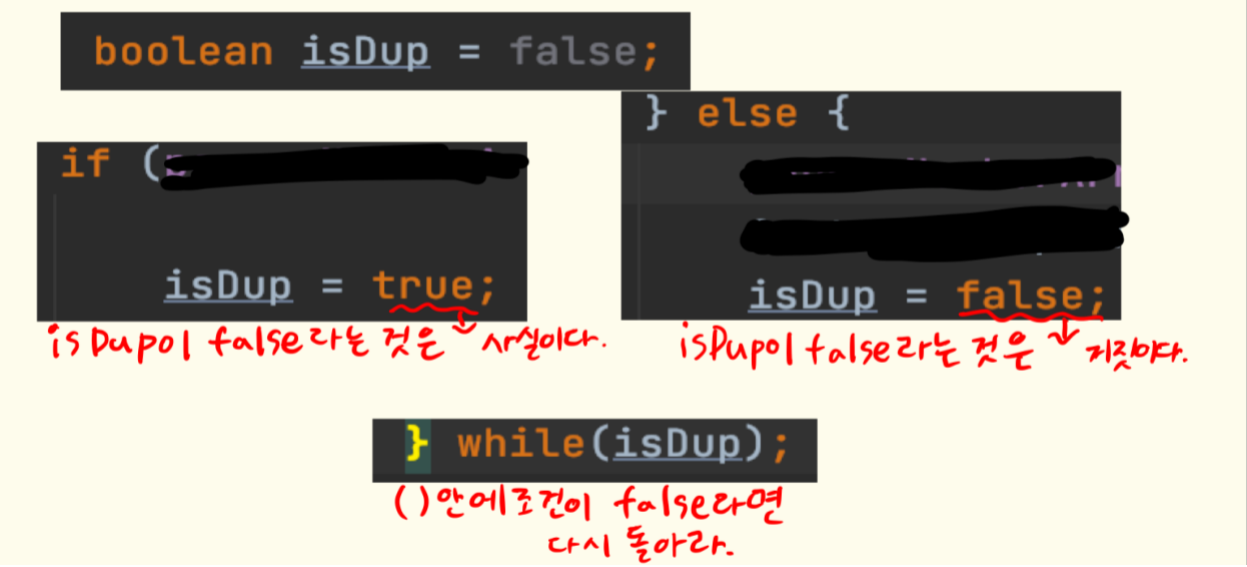
else문에서는 랜덤값을 부여받은 personNumberArr[randNum]이라는

공간에 공간이(0) 비어있다. = 즉 공간에 아무것이 없기때문에 들어갈 수 있습니다.

if문에서는 랜덤값을 부여받은 personNumberArr[randNum]이라는

공간에 공간이(1) 차있습니다. = 즉 공간에 무엇이 있어서 들어갈 수 없습니다.

이제 마지막으로 isDup= false; 이부분이 약간 뇌가 아프다.



이게 약간 말로하면 이상하기도 한데

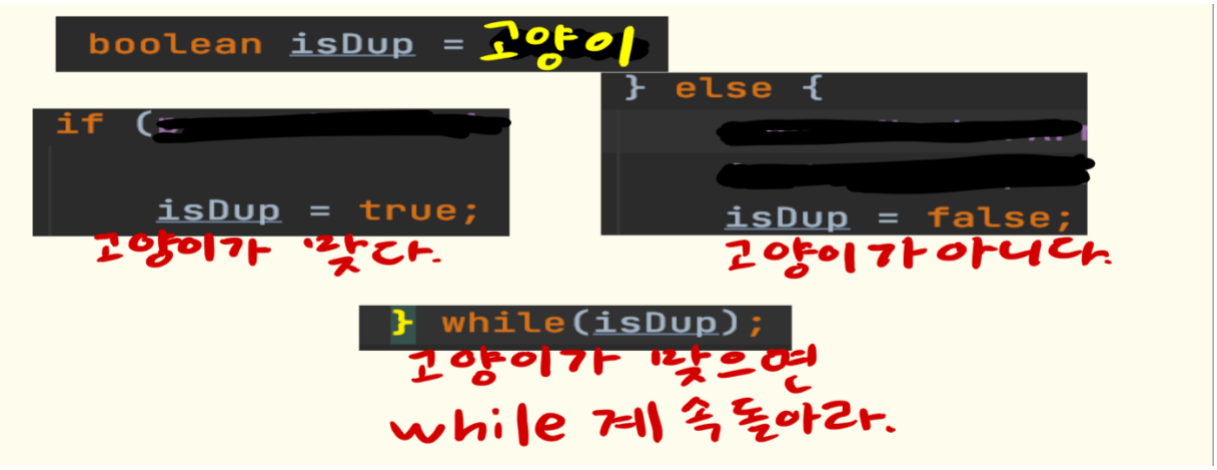
위에서 알아보았듯이 if 안을 동작한다는 것인 else로 이미 먼저 빠져나간 값이 있다는 것이다.

그렇다면 while조건에 맞아서 계속 돌게 되는건 if문이다 이걸 이해하고 보면

isDup자체가 false이니까 이 false가 맞으니까 while문을 돈다. 이렇게 생각하고 else문에서는

isDup자체가 false이니까 이 false는 틀렸으니까 while로 다시 돌지 않는다.

쉽게 이해하려면 isDup=고양이로 생각해보자



이렇게보면 왜 고양이가 맞다면 while을 돌게되니까 **false라는 isDup이 true면 while을 돈다.**

위까지의 결론

```
do {
    randNum = (int) (Math.random() * 50);
    System.out.println(randNum);
    System.out.println("input");

    if (personNumberArr[randNum] != 0 ) {
    // personNumberArr[randNum]에 공간에 이미 무엇인가 있어서
    // (!= 0)공간이 없다면isDup이 true 니까 while(isDup)을 돌아 다시
    //randNum = (int) (Math.random() * 50);값을 받는다.

    //personNumberArr[27] != 0
    //즉 만약 personNumberArr[27] ! = 0이라면
    //isDup은 false로 주어져있고 if문에서 isDup 즉 false=ture다 그러
    // while(isDup(false)) 와일문에서 isDup이라는 boolean은 true가 맞으니 다시
    // 값을 할당받으러 do 로 가야한다.
    //반대로 else가 문에서는
    //diDup이 false로 주어져 있지만 else문 연산처리하며 isDup이 false = false즉
    //isDup이라는 boolean(isDup=false)이 거짓이 되었다.
    //그러므로 while문으로 가서는 isDup이라는 boolean false가 거짓이기 때문에
    //do문으로 가지않고 빠져나간다.
```

```
    } else {
        personNumberArr[randNum] = 1;
    //personNumberArr[randNum]에 공간이 하나 있다면
    //randNum을 넣는다.
```

```
public void allocTicket(int personNum) {
    boolean isDup = false;
    int randNum;

    do {
        randNum = (int) (Math.random() * 20) + 1;

        if (ticketNumberArr[randNum - 1] != 0) {
            isDup = true;
        } else {
            isDup = false;
            ticketNumberArr[randNum - 1] = personNum;
                                          //=randNum
```

이제 사람을 뽑았으니 티켓을 뽑는다.

위와같은 방식으로 20개를 뽑는다. 주의해야할 것은 personNum은 어디서 왔냐인데

```
personNum = allocRandomPersonNumber();
allocTicket(personNum);
```

allocTicket을 돌기 전 allocRandomPersonNumber을 돌아서 나온 값 (20개)가 personNum이고 그 personNum을 뜻하는 것이다.

즉 allocTicket의 마지막 personNum은 allocTicket안에 int된 randNum이 아닌 allocRandomPersonNumber을 통해 나온 randNum이라는 뜻

*Arraylist

```
public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<~>();

    list.add("MILK");
    list.add("BREAD");
    list.add("BUTTER");

    list.add( index: 1,  element: "APPLE");
    list.add( index: 2,  element: "GRAPE");

    for(int i = 0; i < list.size(); i++) {
        System.out.println(list.get(i));
    }

    System.out.println("**********************");
    list.remove( index: 3);

    for(int i = 0; i < list.size(); i++) {
        System.out.println(list.get(i));
    }
```

위와같이 리스트 안에 들어갈 재료들을 추가하고(list.add) 만약 대규모일때 새로운 것을 추가할 경우 list.add{숫자(자리), 추가할 것}을 써넣는다.

이것의 범위를 정하는 경우 행렬 같이 기차같은 길이 list.length가 아니라 주머니에 담겨있다고 생각하여 list.size를 써야 한다.

출력을 할 시에는 .get( )을 이용한다.

*HashSet:집합클래스

```java
public class HashSetTest {
    public static void main(String[] args) {
        Set<String> s1 = new HashSet<~>();
        Set<String> s2 = new HashSet<~>();

        s1.add("A");
        s1.add("B");
        s1.add("C");
        s1.add("D");

        s2.add("A");
        s2.add("C");
        s2.add("E");

        // Set은 집합 클래스이며
        // HashSet은 상대적으로 속도가 더 빠른 집합 클래스다.
        Set<String> union = new HashSet<~>(s1);
        // addAll의 경우 모든 내용을 합함(중복 허용 x)
        union.addAll(s2);

        Set<String> intersection = new HashSet<~>(s1);
        // retainAll의 경우 모든 내용을 합하는데 교집합 형식으로 해석
        intersection.retainAll(s2);

        System.out.println("합집합 = " + union);
        System.out.println("교집합 = " + intersection);
```

```java
// Set은 순서대로 정렬을 한다.
// HashSet은 순서를 신경쓰지 않는다.
HashSet<String> hs = new HashSet<~>();
String[] sample = {
        "안녕", "하하", "호호", "크크", "키키", "켈켈"
};

// sample에 있는 값을 순서대로 하나 하나 가져와서 s에 배치함
for(String s : sample) {
    // Set은 값이 중복되면 add가 되지 않고 false를 리턴한다.
    if(!hs.add(s)) {
        System.out.println("중복!");
    }

    System.out.println(hs.size() + "출력: " + hs);
```

*Iterator

```java
// ArrayList<>:
// 이 녀석은 다양한 데이터 타입을 수용할 수 있게해준다.
// <String> 문자열, <Integer> 정수, <Float> float 타입 등등
ArrayList<String> list = new ArrayList<~>();
ArrayList<Integer> list2 = new ArrayList<~>();


list.add("one");
list.add("two");
list.add("three");
list.add("four");
list.add("five");

String s;
// ArrayList 를 순회할 수 있는 정보를 획득함
Iterator e = list.iterator();

// e 를 통해 순회할 수 있는 정보가 있다면
while(e.hasNext()) {
    // 해당 위치의 정보를 s에 저장하세요.
    s = (String) e.next();
    System.out.println(s);
```

이해를 돕기위한 설명 아래그림 위는 행렬, 아래는 Iterator *연산속도는 위가 더 빠름

```
// ----------------------------------------
// | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
// ----------------------------------------
// | 1 | 2 |   | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
// ----------------------------------------
// | 1 | 2 |   |   | 5 |   | 7 |   | 9 | 10 |
// ----------------------------------------
// | 1 | 2 | 5 | 7 | 9 | 10 | xxxxxxxxxxxxxxxxxxxxxxx
// ----------------------------------------

// -----     -----     -----
// | 1 | -> | 2 | -> | 4 | -> ... ->     ooooooooooooo
// -----     -----     -----
```

*궁금한점.

Q1 isDup을 false로 줘서 판단을 하는 이유가 인터넷에서 알아본 바로는 연산 속도가 더 빠르다, 메모리 절약 이런 면이던데 맞나요?

```
public int allocRandomPersonNumber() {
    boolean isDup = false;
    int randNum;

    do {
        randNum = (int) (Math.random() * 50);

        if (personNumberArr[randNum] != 0) {
            isDup = true;
        } else {
            isDup = false;
            personNumberArr[randNum] = 1;
        }
    } while(isDup);

    return randNum;
```