디지털컨버전스 기반 UXUI Front 전문 개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - JungHyun LEE(이정현)
akdl911215@naver.com

1번. 13회차(01.15.금) Ticketing 풀이 중 궁금점

링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/Ticketing.java

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

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

        // 위에서 랜덤값 0 ~ 49를 randNum으로 할당하므로
        // personNumberArr[randNum] 배열 자리에 0이 아니면 true
        if (personNumberArr[randNum] != 0) {
            isDup = true;
            // true면 else를 지나서 while(isDup)으로 간다
        } else {
            // 밑에 while문을 끄기 위해 false 값을 준다.
            isDup = false;

            // true가 아니면 랜덤으로 정해진 자리에 값이
            // 없으므로 1값을 넣어줘야 한다.
            // 그러면 0 ~ 49번까지의 모든 배열에 1값이 들어가면
            // 끝나는것이 맞는건가???
            personNumberArr[randNum] = 1;
        }
        // while(isDup) 값이 들어오니 while문이
        // false값이 나올 때 까지 반복한다.
    } while(isDup);
```

2번. 13회차(01.15.금) Ticketing 풀이 중 궁금점

링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/Ticketing.java

```java
public void allocTicket(int personNum) {
    // suDup은 불린 데이터 타입이기 때문에 참 과 거짓을 할당받는데,
    // 거짓으로 셋팅해 놓는다.
    boolean isDup = false;
    int randNum;

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

        // personNumberArr[randNum] 배열 자리에 0이 아니면 true
        // [randNum - 1] 값을 준 이유는 위에서 + 1을 해줬기 때문이다.
        if (ticketNumberArr[randNum - 1] != 0) {
            isDup = true;
        } else {
            isDup = false;
            ticketNumberArr[randNum - 1] = personNum;
        }
        // if 문에서 true 값이 isDup으로 들어감으로써
        // while 문이 실행되고 do로 돌아가서 랜덤값을 생성한다.
    } while(isDup);

    // allocTicket(int personNum) 값은 결국 public int allocRandomPersonNumber()
    // 에서 나온 0 ~ 49 배열 중에서 20개의 중복안되는 값을 만들기 위한 메소드이다.
    // 하지만 personNum을 받는 이유는 무엇이지????
}
```

3번. 13회차(01.15.금) Ticketing 풀이 중 궁금점

링크 :
https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/Ticketing.java

```java
public void ticketingTicket() {
    // 1) 랜덤한 중복되지 않는 0 ~ 49까지의 값으로 사람 번호 매기기
    // 2) 랜덤한 중복되지 않는 0 ~ 19까지의 값으로 예매하기

    int personNum;
    //int cnt = 1;

    // 위의 public Ticketing(int numOfHuman, int numOfTicket) 의
    // 입력받은 int numOfTicket 값만큼 돌것이다.
    // 20명이라는 기준이 있었기에 20이 대입 될것이고,
    // 20회 반복할 것이다.
    for(int i = 0; i < numOfTicket; i++) {

        // allocRandomPersonNumber(); 은 0 ~ 49 배열의 값을
        // 중복되지 않는 값을 가지고 있다.
        // 그렇기 때문에 personNum 에 할당하는 이유는
        // 티켓에 뽑기 위해서라고 추측한다. 그런데, 왜 하나씩만 줄까?
        // 아마 public Ticketing(int numOfHuman, int numOfTicket)
        // 20을 줘서 for문이 20번만 돌아갈 것이다. 그렇기 때문에 20명만
        // 뽑는건데...

        // allocRandomPersonNumber() 50개의 배열값을 가지고
        // 있기때문에 personNum에 50을 할당할 것이다....왜지?????
        personNum = allocRandomPersonNumber();
        //System.out.printf("%3d", personNum);

        //if(cnt % 5 == 0) {
        //    System.out.println("");
        //}

        //cnt++;

        // personNum을 allocTicket으로 리턴해준다.
        allocTicket(personNum);
    }
}
```

4번. 13회차(01.15.금) Ticketing 풀이 중 궁금점

링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/Ticketing.java

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

    do {
        // randNum 에 0 ~ 49 까지의 임의의 정수를 할당한다.
        randNum = (int) (Math.random() * 50);

        // 현재 ArrayList에 randNum이 있나요 ?

        // contains 는 특정 문자열이 포함되어 있는지 확인하는
        // 기능을 가진 함수다. 그럼으로 personArrayList 안에
        // randNum 가 포함되어 있냐에 대한 참과 거짓을 구별하는
        // if 문이다.

        // 다만 randNum 이 있는지 없는지 구별하기 위해 있는것은
        // 알겠는데 어디서 그걸 볼 수 있는지 모르겠다.....
        if (personArrayList.contains(randNum)) {
            isDup = true;
        } else {
            isDup = false;
            personArrayList.add(randNum);
        }
    } while(isDup);

    return randNum;
}
```

5번. 13회차(01.15.금) Ticketing 풀이 중 궁금점

링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/Ticketing.java

```java
public void ticketingArrayListTicket() {
    int personNum;

    // numOfTicket 은 20이므로 20회 반복한다.
    for(int i = 0; i < numOfTicket; i++) {
        // allocArrayListRandomPersonNumber을 호출하기 때문에
        // 리턴값 randNum 을 personNum에 할당한다.
        personNum = allocArrayListRandomPersonNumber();

        // 할당받은 personNum 값을 ticketArrayList 에
        // personNum 을 할당한다 ??????

        // 예를들어서 ticketArrayList[0] 이라는 배열이 있다고 가정한다.
        // 그렇다면 ticketArrayList[0] = "personNum"; 이라고 표현할 것이다.
        // 결국 personNum 값을 할당하는 표현식이라고 생각이 된다.
        ticketArrayList.add(personNum);
    }
}
```

6번. 13회차(01.15.금) Ticketing 풀이 중 궁금점

링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/Ticketing.java

```java
public void printTicketArrayList() {
    int cnt = 1;

    // ticketNum 은 정수형이다.
    Integer ticketNum;
    // ticketArrayList 를 순회할 수 있는 정보를 얻음
    Iterator e = ticketArrayList.iterator();
    // ArrayList를 탐색할 때는 Iterator를 쓴다. Iterator는 객체지향
    // 프로그래밍에서 주로 사용하는 반복 기법이다. Iterator를 쓰려면
    // 우선 Iterator 객체를 만들어야하기 때문에
    // Iterator e = ticketArrayList.iterator(); 값을 할당받은 듯 하다.

    // 순회할 수 있는가 ?
    // 데이터가 없으면 루프 진행 x
    // 데이터가 하나라도 있으면 루프 진행 o
    // hasNext() 가 순회 할 수 있는지 묻는거다.
    while(e.hasNext()) {
        // 존재하는 값을 가져와서 Integer 형식으로 저장합니다.
        // 정수형 데이터 타입을 ticketNum에다가 할당한다.
        // 검색해보니 메소드는 호출될 때마다 엘리먼트를 순서대로
        // 리턴한다고 하는데 아마 모든 값들을 순서대로 리턴한다고
        // 생각이 된다.

        // public Ticketing(int numOfHuman, int numOfTicket) 안의
        // ticketArrayList = new ArrayList<Integer>(); 을 입력받고
        // ticketArrayList 은 20 입력되니 결국 20이 할당될것이다.
        ticketNum = (Integer) e.next();
        // ticketNum 값을 출력하는데 아마 0 ~ 19가 출력되지 않을까 한다???
        System.out.printf("%3d", ticketNum);

        if(cnt % 5 == 0) {
            System.out.println("");
        }

        cnt++;
    }
}
```

7번. 13회차(01.15.금) RandomTeamSelection 궁금점

본문 링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/RadomTeamSelection.java

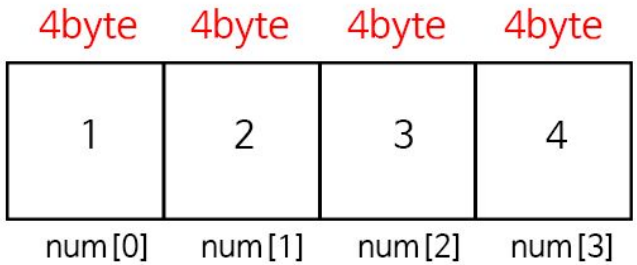
출력 링크 : https://github.com/akdl911215/GroupStudy/blob/main/2/junghyunlee/java_work/src/Thirteenth/RandomTeamSelectionHomeWork.java

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 17 out of bounds for length 17
    at Thirteenth.RadomTeamSelection.allocArrTeamChoice(RadomTeamSelection.java:81)
    at Thirteenth.RandomTeamSelectionHomeWork.main(RandomTeamSelectionHomeWork.java:11)
```

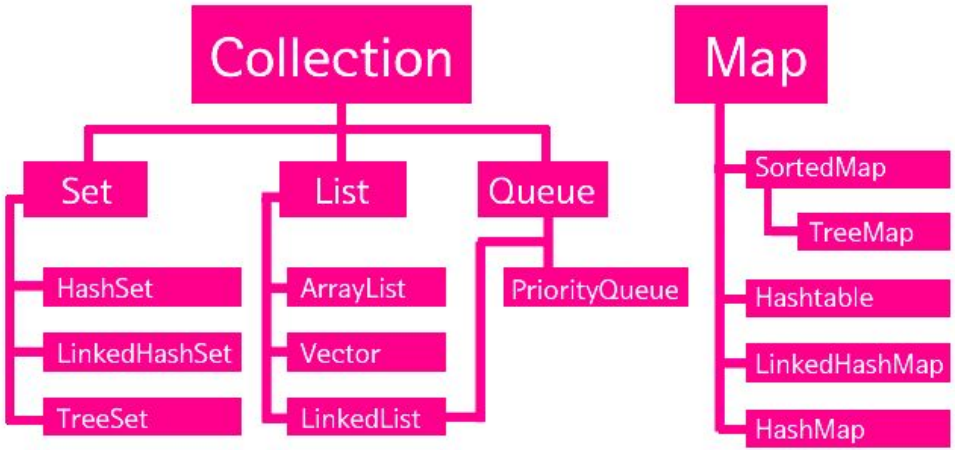
RandomTeamSelection 의 코드와 RandomTeamSelectionHomeWork를 실행했을 경우 나오는 에러입니다.
링크에서 들어가서 보면 `RadomTeamSelection rt = new RadomTeamSelection (17);`
주고, 배열의 길이도 17을 줌으로써 배열길이의 17을 넘어가지 않을것이라는 생각을 하는데 계속 이런 오류가 뜹니다.

컬렉션즈 프레임워크 [Collections Framework] 장점과 종류

배열의 체감상 느낄수 있는 단점은 배열은 그 배열을 선언할 때 그 배열이 몇개의 배열의 값을 가질 수 있는지 지정할 수 있습니다. 그리고 그 값보다 더 많은 배열의 값을 입력하게 되면 오류가 발생하게 됩니다. 하지만, 배열에서 느껴지는 불편함을 컬렉션즈 프레임워크를 사용하면 불편함을 줄일 수 있습니다.



실제로 비교해보자면 배열은 끝을 정해놓고 사용할 수 밖에 없다는 것입니다. 하지만 컬렉션즈 프레임워크는 정해놓지않고 끊임없이 붙여서 사용할 수 있는 장점이 있습니다.



컬렉션 프레임워크에는 옆의 그림을 보다시피 여러가지 종류가 존재합니다.

List의 경우는 중복이 가능합니다. 예를들어서 3개의 리스트안에 1, 1, 2 이런식의 중복이 가능합니다.

하지만, Set의 경우 중복되지 않은숫자만 가능합니다. 즉, 1, 2, 3 이런식으로 저장을 해야합니다.

또는 키밸류형식의 컨테이너가 필요하다면 Map을 선택해야합니다. 각각의 카테고리 마다 데이터를 가져오는 형식이 다르기 때문에 상황에 맞춰 사용하면 됩니다.