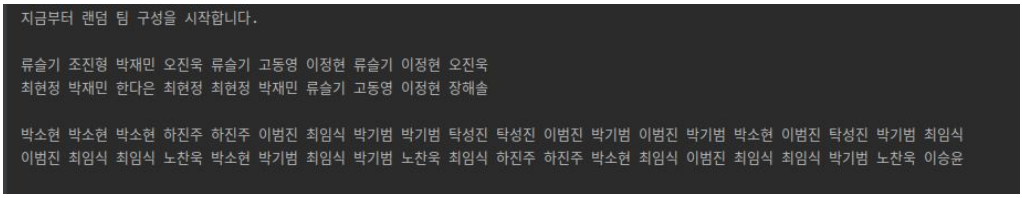
디지털컨버전스 기반 UXUI Front 전문 개발자 양성과정

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

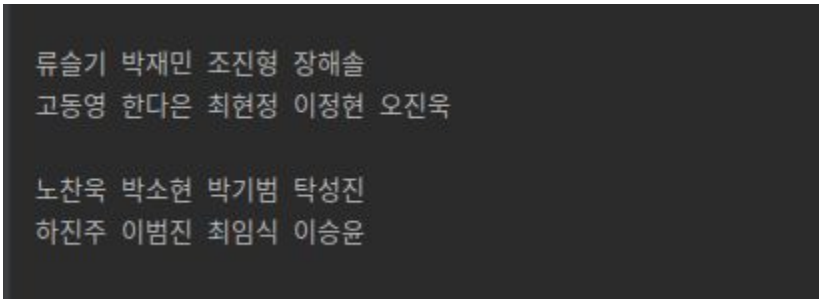
1)CreateTeam 주석 및 궁금점 2, 3 페이지

링크 :
https://github.com/akdl911215/GroupStudy/blob/dc093605a66dd95926ff08f7e4fe4e3c030c99b8/junghyunlee/java_work/src/Fourteenth/CreateTeam.java

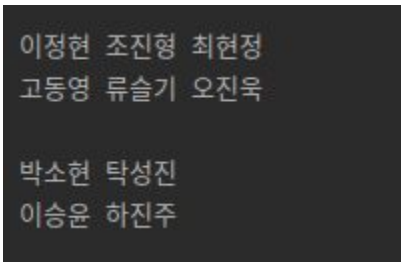
```
// 입력 값은 위의 allocArrayList 2개 케이스를 순서대로 입력받는다.
// 아마 A팀이
// allocArrayList(
//    AteamArrayList, AteamArr, AnumOfPerson
//        ); 정보가 위에 있기때문에 먼저 처리가 될듯하다.
public void allocArrayList(ArrayList<String> al, String[] arr, int loopNum) {

    // is Dup을 false로 셋팅한다.
    boolean isDup = false;

    for(int i = 0; i < loopNum; i++) {
        // 우리는 A 배열, B 배열이 나눠진 케이스를 받으므로
        // 하나의 배열에서 모든 값을 처리하지 않는다.
        // 그러므로 start를 별도로 만들 필요가 없었다.
        // int randNum = (int)(Math.random() * loopNum) + start;
        do {
            // randNum 에는 각 팀의 숫자의 랜덤값을 할당해줄 수 있다.
            // ex) A팀의 경우 loopNum이 AnumOfPerson이 할당되기 때문에
            // 0 ~ 8까지 9가지의 값을 할당해줄 수 있다.
            int randNum = (int)(Math.random() * loopNum);

            // ArrayList<String> al 안에 arr[randNum] 특정 문자열이 포함
            // 되어 있는지 확인하는 함수이다. 그러므로 랜덤으로 뽑힌 배열안에
            // al이 포함되어 있으면 if문 실행 아니면 else 실행하게 된다.
            if(al.contains(arr[randNum])) {
                isDup = true;

                // do ~ while 문은 기본적으로 한번 실행하고 while로 true이면
                // 실행하고 false이면 실행하지 않는다. 그러면 continue가
                // 왜 있을까? continue가 있을때와 주석했을때 의 결과차이가 발생했다.
                // continue가 있는이유는 밑에 있는 al.add(arr[randNum]);이
                // 실행되지 않도록 하기위해서 제대로된 값을 출력해주기 위한 보정이 목적인듯 하다.
                continue;
            } else {
                isDup = false;
            }

            al.add(arr[randNum]);
            // 그래서 do ~ while문은 필요없을것 같았지만, 실제로 결과값이
            // 다르게 나오게 된다. do는 1회를 무조건 반복하고 while에서 결과에 따라
            // 반복을 시작하는 것인데... continue가 있으니 반복을 하게 되니 사실상
            // do ~ while문은 필요가 없는것이 아닐까???
        } while(isDup);
    }
}
```



위에는 continue 를 제거한 결과값.



위에는 원래 코드 출력값



위에는 원래 do ~ while 문 제거 출력값.

이러한 결과를 비교해보니 내가 생각했을때 do ~ while 문을 넣어야하는이유는 우리가 팀을 두번 돌리기 때문일 것이라고 생각된다. 즉, do 가 없으면 한번 돌고나서 false 값이 되기때문에 돌지 않기때문이다.

1)CreateTeam 주석 및 궁금점 4, 5, 6 페이지

링크 :
https://github.com/akdl911215/GroupStudy/blob/dc093605a66dd95926ff08f7e4fe4e3c030c99b8/junghyunlee/java_work/src/Fourteenth/CreateTeam.java

```
// printArrayList는 A팀이 4, 5 또는 5, 4 이냐를 결정하기 위하기 때문에
// 1회만 사용된다.
public void printArrayList(ArrayList<String> al) {
    String name;
    // ticketArrayList를 순회할 수 있는 정보를 얻음
    // Iterator는 자바의 컬렉션 프레임워크에서 컬렉션에 저장되어 있는
    // 요소들을 읽어오는 방법을 표준화한 것이다.
    // 그러므로 al.iterator 에 해당하는 리스트배열 ( A or B )
    // 값을 순회하면서 Iterator e 정보를 얻는다
    Iterator e = al.iterator();

    int cnt = 1;
    int divNum;
    // al.size()는 al의 총사이즈를 가지고 오는거 같습니다.
    // 하지만, al의 사이즈가 a팀의 경우라 가정하였을 때, 밸류값이 9개라서 9가
    // 되는건지 al의 배열사이즈가 9이기때문에 9가되는지 헷갈립니다.
    int quot = al.size() / numOfTeam;
    int remain = al.size() % numOfTeam;

    // al로 넘어온 al의 사이즈가 numOfTeam로 나눴을 경우
    // 0 보다 큰가? 를 참과 거짓으로 구별하라
    boolean needException =
            (remain > 0) ? true : false;

    int randValue = 0;
```

```
// e.hasNext 사용이유 :
// Iterator 개체는 컬렉션 개체의 iterator 메서드를 호출하여
// 얻어올 수 있습니다. 그리고 hasNext 메서드로 이동이 가능한지
// 확인한 후에 next 메서드로 해당 위치의 보관한 개체를 참조하여
// 원하는 작업을 수행한다고 합니다.

// 그럼으로 Iterator e = al.iterator(); 에서 얻어온 정보들을
// e.hasNext 메소드로 이동이 가능한지를 확인한 후에 이동할 수 있다면
// while문은 실행하게 됩니다. 그리고 while문이 실행된다는 것은
// 곧 Iterator e 의 모든 값들을 할당받은 것입니다.
while(e.hasNext()) {
    // e.next() 에 들어가있는 해당값들을 전부 name 에 할당한다.
    name = (String) e.next();
    System.out.printf("%s ", name);

    // 현재 케이스에서는 무조건 앞에 5명이 나온다.
    // 그러므로 이것도 랜덤하게 4, 5 혹은 5, 4가 나오게 해줘야 한다.

    // needException은 A조에서 4, 5 혹은 5, 4 로 랜덤으로 선택하기
    // 위해서 만들었기 때문에 0 ~ 1 값 한번만 출력하기 하고나서
    // needException을 false 로 만든다.
    if(needException) {
        randValue = (int)(Math.random() * 2);
        needException = false;
    }
```

```
            // 밑에 if 문은 항상 실행된다. 다만, randValue 값이 0 이나오면
            // (quot + randValue) = 4값이 되고 cnt 가 4가 됐을 경우
            // 나머지가 0이되서 A 팀이 4명이 되고, randValue 값이 1 나올 경우에는
            // (quot + randValue) = 5값이 되서 A 팀이 5가 된다.
            if((cnt % (quot + randValue)) == 0) {
                System.out.println("");

                // if 는 A 팀이 4명이 될경우 실행
                if(cnt == 4) {
                    randValue = 1;
                    // else 는 A 팀이 5명이 될경우 실행
                } else {
                    randValue = 0;
                }

                cnt = 0;
            }

            cnt++;
        }

        System.out.println("");
    }

    public ArrayList<String> getAteamArrayList() {
        return AteamArrayList;
    }

    public ArrayList<String> getBteamArrayList() {
        return BteamArrayList;
    }
}
```