# 디지털컨버전스 기반 UI/UX Front 전문 개발자 양성과정

A조 박재민

# 2021-01-15(ArrayList) 복습

```java
public class ArryaListTest {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<~>();
        // 기본길이(용량, capacity)가 string인(또는 Integer 정수) ArrayList를 생성
        // <> --> 타입 변수 자리 : int, String 등
        // 순서가 있는 데이터 집합, 데이터 중복을 허용, 저장순서
        // 데이터의 저장공간으로 배열을 사용한다(배열기반)
        // ArrayList에는 객체만 저장가능, 원래는 list.add(new String("MILK"));로 작성해야지만
        // autoboxing에 의해 컴파일러가 기본형에서 참조형으로 자동 변환
        list.add("MILK");
        list.add("BRAED");
        list.add("BUTTER");
        // 기존 목록, Object배열

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

        // 추가 목록 몇번째에 위치할지 index(저장위치) 및 요소(타입에 따라 다름)
        // - void add(int index, Object element)
```

```java
        for(int i = 0; i <list.size(); i++){ // size값 위 list.add값 확인 = 5
            System.out.println(list.get(i));

            // 추가 목록 포함한걸 for문으로 list.get(i)값 출력
        }

        System.out.println("**********");
        list.remove( index: 3);
        //인덱스가 3인 곳에 저장된 요소를 삭제

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

# 2021-01-15(HashSet) 복습

```java
public class HashSetTest { //순서x, 중복x
    public static void main(String[] args) {
        // - Set 인터페이스를 구현한 대표적인 컬렉션 클래스
        // - 순서를 유지하려면 Linked HashSet 클래스를 사용하면 된다.
        Set<String> s1 = new HashSet<String>();
        Set<String> s2 = new HashSet<String>();

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

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

```java
        // set은 집합 클래스이며 - 순서가 중요할 때 사용!
        // HashSet은 상대적으로 속도가 더 빠른 집합 클래스다.
        Set<String> union = new HashSet<String>(s1);
        // addAll의 경우 모든 내용을 합함(중복 허용x)
        union.addAll(s2);

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

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


        // set은 순서대로 정렬을 한다.
        // HashSet은 순서를 신경쓰지 않는다.
        // 순서가 중요하지 않을 때 사용!
        HashSet<String> hs = new HashSet<String>();
        String[] sample = {
                "안녕", "하하", "호호", "크크", "키키", "켈켈"
        };


        // sample에 있는 값을 순서대로 하나하나 가져와서 s에 배치하는 구문
        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);
        }
```

# 2021-01-15(Iterator) 복습

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

String s;
// Iterator = ArrayList 를 한번 순회(훑어보고) 할 수 있는 정보를 획득함.
Iterator e = list.iterator(); // Iterator 객체를 반환
// e를 통해 순회할 수 있는 정보가 있다면
while (e.hasNext()){ // 읽어올 요소가 있는지 확인
    s = (String) e.next(); // 해당 위치의 정보를 s에 저장(대입)하세요, 요소를 읽어오기
    System.out.println(s); // 출력하세요
    // Iterator는 1회용이라 다쓰고나면 다시 얻어와야한다.
```

# 2021-01-15(Thirteenth-Ticketing) 질문

*클래스명 : Ticketing

*Git 링크 :
https://github.com/djd4532/GroupStudy/tree/main/1/ParkJaemin/java_work/src/Thirteenth

```
public int allocRandomPersonNumber() { // 사람의 번호 임의 할당
    boolean isDup = false; // 중복이면 false
    int randNum; // 난수

    do { // do-while 문 활용
        randNum = (int) (Math.random() * 50);
        // 랜덤 값 할당 전체 50명 사람의 번호
        if (personNumberArr[randNum] != 0) {
            isDup = true; // 중복이 아니라면 if문을 벗어난다.
        } else {
            isDup = false;
            personNumberArr[randNum] = 1;
        }
    } while (isDup);

    return randNum; // 중복이면 다시 번호 할당?
}
```

# 2021-01-15(Thirteenth-Ticketing) 질문

*클래스명 : Ticketing

```
public int allocArrayListRandomPersonNumber() { // 사람들 번호 무작위 어레이 리스트 할당
    boolean isDup = false; // 중복이면 false
    int randNum;

    do {
        randNum = (int) (Math.random() * 50); // 50명 랜덤번호 randnum에 할당

        // 현재 ArrayList에 randNum이 있나요 ?
        if (personArrayList.contains(randNum)) {
            //.contains는 randNum에 중복이 포함되어 있는지 확인하는 함수인가?
            isDup = true;
        } else {
            isDup = false;
            personArrayList.add(randNum);
        }
    } while (isDup);

    return randNum; // 중복일 경우 다시 호출한 randNum으로 그냥 되돌아간다.
```

# 2021-01-15(Thirteenth-Ticketing) 코드 해석

*클래스명 : Ticketing

*올바르게 코드 해석을 했는지 리뷰 부탁드립니다.

```
public void printTicketArrayList() { // 티켓 어레이 리스트 20장 출력 메서드
    int cnt = 1; // 초기값 1
    Integer ticketNum; // 티켓 번호
    // Iterator = ticketArrayList를 한번 순회 할 수 있는 정보를 획득
    // 50명중 20명이 티켓팅한 정보 획득
    Iterator e = ticketArrayList.iterator(); // Iterator 객체를 반환

    while (e.hasNext()) {// hasNext()를 통해 읽어올 요소가 있는지 확인
        ticketNum = (Integer) e.next(); // 해당 e=tickeArrayList의 정보를 ticketNum에 저장(대입)
        System.out.printf("%3d", ticketNum); // 20명의 티켓팅한 사람들의 번호 출력
        if (cnt % 5 == 0) { // 5명씩 출력 후 줄 바꿈
            System.out.println("");
        }

        cnt++;
    }
}
```