```java
package Thirteenth;

import java.util.ArrayList;

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

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

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

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

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

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

```java
package Thirteenth;

import java.util.HashSet;
import java.util.Set;

public class HashSetTest {
    public static void main(String[] args) {
        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");

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

        Set<String> intersection = new HashSet<String>(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>();
        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);
        }
    }
}
```

```java
import java.util.ArrayList;
import java.util.Iterator;

public class IteratorTest {
    public static void main(String[] args) {
        // ArrayList<>: 선추선형 순서.
        // 이 녀석은 다양한 데이터 타입을 수용할 수 있게해준다.
        // <String> 문자열, <Integer> 정수, <Float> float 타입 등등
        ArrayList<String> list = new ArrayList<String>();
        ArrayList<Integer> list2 = new ArrayList<Integer>();

        // ----------------------------------------
        // | 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
        // -----    -----    -----

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

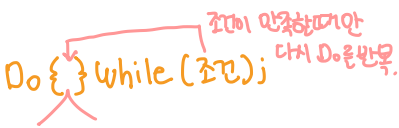
list2는 어디에 쓰이는건가요?

list의 값들은 iterator를통해 list2에 옮겨진 것?

```
    String s;
    // ArrayList 를 순회할 수 있는 정보를 획득함
    Iterator e = list.iterator();
                    list에 있는 정보확인.
    // e 를 통해 순회할 수 있는 정보가 있다면
    while(e.hasNext()) {
        // 해당 위치의 정보를 s에 저장하세요.
        s = (String) e.next();
        System.out.println(s);
    }
}
```

```java
package Thirteenth;

import java.util.ArrayList;
import java.util.Iterator;

public class Ticketing {
    private int numOfHuman;
    private int numOfTicket;

    private int[] personNumberArr;
    private int[] ticketNumberArr;

    private ArrayList<Integer> personArrayList;
    private ArrayList<Integer> ticketArrayList;

    public Ticketing(int numOfHuman, int numOfTicket) {
        this.numOfHuman = numOfHuman;
        this.numOfTicket = numOfTicket;

        // 사실 현재 케이스에서는 배열만 쓰거나
        // ArrayList만 쓰는것이 좋지만
        // 약간 손 볼 곳이 많으므로
        // 그냥 둘 다 할당하도록 한다.
        personNumberArr = new int[50];
        ticketNumberArr = new int[20];

        personArrayList = new ArrayList<Integer>();
        ticketArrayList = new ArrayList<Integer>();
    }
```

```java
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;
}

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

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

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



↳ Do 부터시작하기에 무조건 1번은 실행.

500명의 랜덤번호 중복없이 부여하는 식.

false ⇒ false 인동안만 Do반복해라.

allocRandomPersonNumber.

왜 이부분은 1이 아니라 personNum이 들어갔나요?

20개의 티켓중 이미 들어간 personNum의 값이 있는지 확인하는 식?

```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;

        for(int i = 0; i < numOfTicket; i++) {
            personNum = allocRandomPersonNumber();
            //System.out.printf("%3d", personNum);

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

            //cnt++;

            allocTicket(personNum);
        }
    }

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

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

            // 현재 ArrayList에 randNum이 있나요 ?
            if (personArrayList.contains(randNum)) {
                isDup = true;
            } else {
                isDup = false;
                personArrayList.add(randNum);
            }
        } while(isDup);

        return randNum;
    }
```

ArrayList를 사용해 중복확인하는 식.

무한반복문 실행.

매커니즘 ?

① if 문이 참일경우:
↓
isDup=false → while 를통해 Do반복
→ 50개 랜덤값 다시 뽑기.

② else 일 경우.
↓
PersonArrayList에 randNum add 해라!

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

    for(int i = 0; i < numOfTicket; i++) {
        personNum = allocArrayListRandomPersonNumber();

        ticketArrayList.add(personNum);
    }
}

/*
public void printPersonNumber() {
    int cnt = 1;

    for(int i = 0; i < personNumberArr.length; i++) {
        System.out.printf("%3d", personNumberArr[i]);

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

        cnt++;
    }
}
```

20개의 랜덤 숫자(30이하)를 얻고 ticketArrayList에 add해주기.

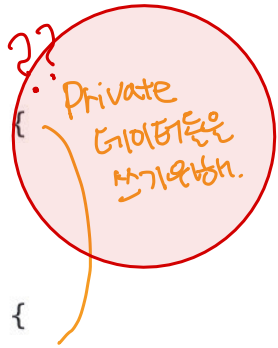
20개.

```java
    public void printTicketArrayList() {
        int cnt = 1;
        Integer ticketNum;
        // ticketArrayList를 순회할 수 있는 정보를 얻음
        Iterator e = ticketArrayList.iterator();

        // 순회할 수 있는가 ?
        // 데이터가 없으면 루프 진행 x
        // 데이터가 하나라도 있으면 루프 진행 o
        while(e.hasNext()) {
            // 존재하는 값을 가져와서 Integer 형식으로 저장합니다.
            ticketNum = (Integer) e.next();
            System.out.printf("%3d", ticketNum);

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

            cnt++;
        }
    }

    public void printArr(int[] arr) {
        int cnt = 1;

        for(int i = 0; i < arr.length; i++) {
            System.out.printf("%3d", arr[i] + 1);

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

            cnt++;
        }
    }
```

스캔느낌.

ticketArrayList에 다음의 인덱스 값이 있는지 확인.

```
153     public void printArr(int[] arr) {
154         int cnt = 1;
155 
156         for(int i = 0; i < arr.length; i++) {
157             System.out.printf("%3d", arr[i] + 1);
158 
159             if(cnt % 5 == 0) {
160                 System.out.println("");
161             }
162 
163             cnt++;
164         }
165     }
166 
167     public int[] getPersonNumberArr() {
168         return personNumberArr;
169     }
170 
171     public int[] getTicketNumberArr() {
172         return ticketNumberArr;
173     }
174 }
```



```java
package Thirteenth;

public class TicketingTest {
    public static void main(String[] args) {
        // Queue(ArrayList) 연습 문제
        // 20명이 승차할 수 있는 열차가 있다.
        // 이 열차에 50명이 예매를 하려고 한다.
        // 50명중 누가 예매에 성공했는지 출력하도록 프로그래밍 해보자!
        // 랜덤을 사용해야 한다.
        // 1 ~ 20 까지 중복되지 않는 숫자가 할당된다.
        // 사용자는 0 ~ 49 까지 순차적으로 예매를 하는 형식이 아니며
        // 랜덤한 숫자로 0 ~ 49까지중 선별되어 예매를 할 수 있게 되어있는 시스템이다.
        Ticketing t = new Ticketing(50, 20);

        //t.printPersonNumber();
        t.ticketingTicket();
        System.out.println("**********************");
        t.printArr(t.getTicketNumberArr());

        System.out.println("++++++++++++++++++++++++");
        Ticketing t2 = new Ticketing(50, 20);
        t2.ticketingArrayListTicket();
        System.out.println("++++++++++++++++++++++++");
        t2.printTicketArrayList();
    }
}
```

전체 50명의 사람 예매 번호는 중복없이 산출
20개의 자리 한당 후 50명의 사람들에게 제공.