# 주의사항

주어진 예제 코드를 수정하지 않고, 새 코드를 추가하기만 해야 함.

# 프로젝트 업로드

## 프로젝트명 K학번

예:) K201032054

프로젝트명이 다른 경우에 감점(-5)

## 패키지명과 파일명

주어진 문제와 package명과 파일명이 동일해야 함.

다른 경우에 감점(-5)

## 프로젝트 제출

프로젝트 폴더만 압축해서 제출하라 (예: K201032054 폴더)

제출할 파일명: K학번**-일련번호.zip** 예: K201032054-1.zip

프로젝트 폴더가 아닌 다른 폴더를 압축해서 제출한 경우 감점(-5)

**중간 제출:**

30분 마다, 프로젝트폴더를 zip으로 압축하여 제출하라.

파일명에 한글이 포함되면 업로드할 때 에러 발생함.

제출된 파일은 다운로드 할 수 없고, 삭제할 수도 없다.

만약 이미 업로드한 파일에 문제가 있어서, 다시 업로드해야 한다면,

일련번호를 더한 파일명으로 새로 업로드하라. (예: K201032054-2.zip, K201032054-3.zip)

**최종 결과물 제출:**

제출할 파일명은 C학번-일련번호.zip (예: K201032054-2.zip) (파일명에 한글이 포함되면 절대 안 됨)

만약 이미 업로드한 파일에 문제가 있어서, 다시 업로드해야 한다면,

일련번호를 더한 파일명으로 새로 업로드하라. (예: K201032054-5.zip)

일련번호가 가장 큰 파일이 최종 결과물로 인정된다.

## 무선 네트웍

교내 무선 네트웍에 연결해야 ftp 업로드 가능함

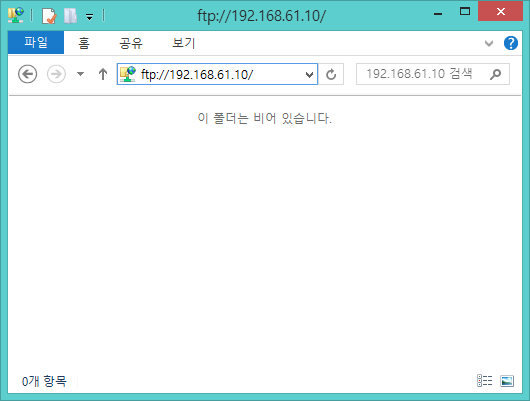
usb 메모리를 이용해서 강의실 PC에서 업로드 해도 됨.

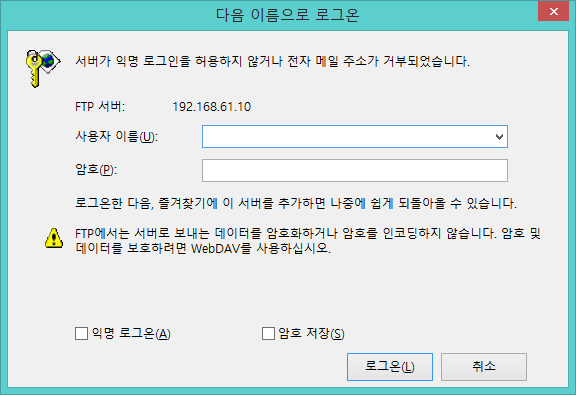
## ftp 업로드 방법

Windows 탐색기의 주소칸에

ftp://192.168.61.10

를 입력하고 엔터 키를 누른다.





사용자 이름: upload

암호: upload

### 업로드에 문제가 있으면

https://filezilla-project.org/

FileZilla FTP Client 프로그램을 다운로드하고 설치하여

이 프로그램을 이용해서 업로드하라.

## 업로드 잘되었는지 확인하기

Windows 탐색기를 종료한다.

Windows 탐색기를 다시 실행한다.

ftp 서버에 연결한다.

파일 목록에서 내가 제출한 파일명을 찾는다.

# 문제

## exam1

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  public class Exam01 {  public static List<String> solution(List<String> list) {  // 구현하라.  }  public static void main(String[] args) {  List<String> list1 = new ArrayList<>();  list1.addAll(Arrays.asList("d", "a", "b", "a", "c", "d", "e", "f", "e"));  List<String> list2 = solution(list1);  System.out.println(list1);  System.out.println(list2);  }  } |

solution 메소드를 구현하라.

파라미터 List 객체에는 변화가 없어야 한다.

새 List 객체를 생성하여 리턴하라.

새 List 객체에는 파라미터 List 객체의 값들이 들어있어야 한다.

새 List 객체에는 중복되는 값이 없어야 한다.

출력

|  |
| --- |
| [d, a, b, a, c, d, e, f, e]  [d, a, b, c, e, f] |

## exam2

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  public class Exam02 {  public static void solution(List<String> list) {  // 구현하라.  }  public static void main(String[] args) {  List<String> list = new ArrayList<>();  list.addAll(Arrays.asList("d", "a", "b", "a", "c", "d", "e", "f", "e"));  solution(list);  System.out.println(list);  }  } |

solution 메소드를 구현하라.

파라미터 List 객체에서 중복되는 값을 제거하라.

출력

|  |
| --- |
| [d, a, b, c, e, f] |

## exam3

|  |
| --- |
| import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class Exam03 {  public static void solution(String s) {  // 구현하라  }  public static void main(String[] args) {  String s = "a1223 (235) 157, 7814; asd\_32as";  solution(s);  }  } |

파라미터 문자열에서 숫자들만 출력하라

숫자는 0~9 사이 문자들로 구성됨.

출력

|  |
| --- |
| 1223  235  157  7814  32 |

## exam4

|  |
| --- |
| import java.util.Arrays;  public class Exam04 {  public static String[] solution(String s) {  // 구현하라.  }  public static void main(String[] args) {  String s = "a1223 (235) 157, 7814; asd\_32as";  String[] a = solution(s);  System.out.println(Arrays.toString(a));  }  } |

solution 메소드를 구현하라.

파라미터 문자열에서 숫자들만 추출하여,

그 숫자들이 들어있는 문자열 배열을 리턴하라.

숫자는 0~9 사이 문자들로 구성됨.

출력

|  |
| --- |
| [1223, 235, 157, 7814, 32] |

## exam5

|  |
| --- |
| public class Exam05 {  public static List<String> union(String[] a1, String[] a2) {    }  public static void main(String[] args) {  String[] a1 = { "a", "d", "a", "b", "b", "c" };  String[] a2 = { "c", "a", "e", "f", "b", "g" };  List<String> list = union(a1, a2);  System.out.println(list.toString());  }  } |

union 메소드를 구현하라.

두 파라미터 배열의 합집합을 List<String> 객체에 담아서 리턴하라.

리턴값에는 중복되는 값이 없어야 한다.

출력

|  |
| --- |
| [a, d, b, c, e, f, g] |

## exam6

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  public class Exam06 {  public static List<String> intersection(String[] a1, String[] a2) {    }  public static void main(String[] args) {  String[] a1 = { "a", "d", "a", "b", "b", "c" };  String[] a2 = { "c", "a", "e", "f", "b", "g" };  List<String> list = intersection(a1, a2);  System.out.println(list.toString());  }  } |

intersection 메소드를 구현하라.

두 파라미터 배열의 교집합을 List<String> 객체에 담아서 리턴하라.

리턴값에는 중복되는 값이 없어야 한다.

출력

|  |
| --- |
| [a, b, c] |

## exam7

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  public class Exam07 {  public static List<Integer> toList(.......) {  }  public static void main(String[] args) {  System.out.println(toList(7));  System.out.println(toList(2, 3, 5));  System.out.println(toList(7, 3, 4, 5, 6));  }  } |

toList 메소드를 구현하라.

파라미터 숫자 목록을 List<Integer> 객체에 담아서 리턴하라.

출력

|  |
| --- |
| [7]  [2, 3, 5]  [7, 3, 4, 5, 6] |

## exam8

|  |
| --- |
| import java.util.Arrays;  public class Exam08 {  public static void insert(String[] a, int index, String s) {    }  public static void main(String[] args) {  String[] a = {"a", "c", "d", "f", "g", "h"};  insert(a, 1, "b");  insert(a, 4, "e");  System.out.println(Arrays.toString(a));  }  } |

insert 메소드를 구현하라.

파라미터 배열 a의 index 위치에 s 값을 끼워 넣어라.

index 위치 뒤의 값들은 한칸씩 뒤로 밀려야 한다.

출력

|  |
| --- |
| [a, b, c, d, e, f] |

## exam9

|  |
| --- |
| import java.util.Arrays;  public class Exam09 {  public static String[] insert(String[] a, int index, String s) {    }  public static void main(String[] args) {  String[] a = {"a", "c", "d", "f", "g", "h"};  a = insert(a, 1, "b");  a = insert(a, 4, "e");  System.out.println(Arrays.toString(a));  }  } |

insert 메소드를 구현하라.

파라미터 배열 a의 index 위치에 s 값을 끼워 넣은, 새 배열을 생성해서 리턴하라.

index 위치 뒤의 값들은 한칸씩 뒤로 밀려야 한다.

파라미터 배열은 변화가 없어야 하고, 새 배열을 생성해서 리턴하라.

새 배열의 크기는 a 배열 크기 + 1 이다.

출력

|  |
| --- |
| [a, b, c, d, e, f, g, h] |

## exam10

|  |
| --- |
| import java.util.Arrays;  public class Exam10 {  public static void remove(String[] a, int index, String s) {    }  public static void main(String[] args) {  String[] a = {"a", "b", "b", "c", "d", "d", "e"};  remove(a, 1, "f");  remove(a, 3, "g");  System.out.println(Arrays.toString(a));  }  } |

remove 메소드를 구현하라.

파라미터 a 배열에서 index 위치의 값을 제거하라.

index 위치 뒤의 값들은 한 칸씩 앞으로 이동해야 한다.

그리고 맨 뒤의 빈 칸에 파라미터 s 값이 채워져야 한다.

출력

|  |
| --- |
| [a, b, c, d, e, f, g] |

## exam11

|  |
| --- |
| import java.util.Arrays;  public class Exam11 {  public static void solution(int[] a, int distance) {    }  public static void main(String[] args) {  int[] a1 = { 3, 5, 6, 7, 9, 10, 12 };  int[] a2 = { 9, 8, 10, 12, 14, 15 };  solution(a1, 2);  solution(a2, 3);  System.out.println(Arrays.toString(a1));  System.out.println(Arrays.toString(a2));  }  } |

solution 메소드를 구현하라.

파라미터 a 배열의 값들이 distance 값만큼 뒤로 이동해야 한다.

그리고 앞 부분의 빈칸에는 -1 값이 채워져야 한다.

출력

|  |
| --- |
| [-1, -1, 3, 5, 6, 7, 9]  [-1, -1, -1, 9, 8, 10] |

## exam12

|  |
| --- |
| import java.util.HashMap;  import java.util.List;  import java.util.Map;  public class Exam12 {  public static String solution(List<String> list) {    }  public static void main(String[] args) {  List<String> a1 = Arrays.asList("a", "a", "b", "a", "c", "d", "e", "b", "e");  List<String> a2 = Arrays.asList("b", "a", "d", "a", "a", "b", "i", "b", "b");  System.out.println(solution(a1));  System.out.println(solution(a2));  }  } |

solution 메소드를 구현하라.

파라미터 목록 객체에서 가장 여러 번 들어있는 값을 출력하라.

출력

|  |
| --- |
| a  b |

## exam13

|  |
| --- |
| import java.util.HashMap;  import java.util.Map;  import java.util.Objects;  public class Exam13 {  static class Data {  private int i;  private String s;  public Data(int i, String s) {  this.i = i;  this.s = s;  }  }  public static void solution(Data[] a) {    }  public static void main(String[] args) {  Data[] a = new Data[] {  new Data(1, "a"), new Data(2, "b"), new Data(1, "a"), new Data(1, "a"),  new Data(2, "b"), new Data(1, "c"), new Data(2, "b"), new Data(2, "a"),  new Data(1, "c"), new Data(2, "a"), new Data(1, "a"), new Data(1, "b")  };  solution(a);  }  } |

solution 메소드를 구현하라.

파라미터 목록 객체에 들어있는 Data 객체 각각에 대해서,

그 객체의 값과, 횟수를 출력하라.

Data 클래스에 필요한 메소드를 구현하라.

solution 메소드를 구현하라.

출력

|  |
| --- |
| Data(2, a) = 2회  Data(1, a) = 4회  Data(2, b) = 3회  Data(1, b) = 1회  Data(1, c) = 2회 |

## exam14

|  |
| --- |
| import java.util.Arrays;  public class Exam14 {  public static String[] solution(String[] a1, String[] a2) {    }  public static void main(String[] args) {  String[] a1 = { "d", "a", "b", "c" };  String[] a2 = { "B", "D", "C", "A" };  String[] a3 = solution(a1, a2);  System.out.println(Arrays.toString(a3));  }  } |

solution 메소드를 구현하라.

새 배열을 생성해서 리턴하라.

새 배열의 크기는 a1 크기 + a2 크기이다.

새 배열에는 a1의 값과 a2의 값이 들어있어야 한다.

출력

|  |
| --- |
| [d, a, b, c, B, D, C, A] |

## exam15

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  import java.util.ListIterator;  public class Exam15 {  public static void solution(List<Integer> list) {    }  public static void main(String[] args) {  List<Integer> list1 = new ArrayList<Integer>();  List<Integer> list2 = new ArrayList<Integer>();  list1.addAll(Arrays.asList(2, 3, 1, 22, 5, 8));  list2.addAll(Arrays.asList(7, 33, 9, 4, 2, 6));  solution(list1);  solution(list2);  System.out.println(list1.toString());  System.out.println(list2.toString());  }  } |

solution 메소드를 구현하라.

파라미터 목록에서 10 미만인 값들의 바로 뒤에, (그 값 + 10)을 삽입하라.

출력

|  |
| --- |
| [2, 12, 3, 13, 1, 11, 22, 5, 15, 8, 18]  [7, 17, 33, 9, 19, 4, 14, 2, 12, 6, 16] |

## exam16

|  |
| --- |
| interface OnClickListener {  void onClick();  }  class MyOnClickListener implements OnClickListener {  Window window;  public MyOnClickListener(Window window) {  this.window = window;  }  @Override  public void onClick() {  System.out.println(window.message);  }  }  class Window {  String message;  OnClickListener listener;  public Window(String message) {  this.message = message;  }  public void setOnClickListener(OnClickListener listener) {  this.listener = listener;  }  public void clickButton() {  listener.onClick();  }  }  public class Exam16 {  public static void main(String[] args) {  Window window = new Window("hello world");  OnClickListener listener = new MyOnClickListener(window);  window.setOnClickListener(listener);  window.clickButton();  }  } |

어노니머스 이너 클래스 문법을 활용하여, 위 소스코드를 최대한 간결하게 구현하라.

출력

|  |
| --- |
| hello world |