# SWE 206: Introduction to

**Software Engineering 231**

## LAB01 : Software Development Process

Participants :

## Abedalaziz Hamad - 202183050

Mahmoud Alhussen – 202182290 Section : 51

|  |  |
| --- | --- |
| Distribution  of Tasks: | |
| Requirements &  Planning | Abedalaziz |
| Distribution  Methods | Abedalaziz |
| Classes | Mahmoud |
| Main methods | Mahmoud |
| Report | Shared |

### Requirements: -

* Generate groups based on students in a section.
* Be able to set a max number of members per group.
* Output the groups in a readable format.

### Design (pseudo code): -

main class {

main () {

1- read file

2- Ask for method of generating groups (by group number, or by studnets per group)

3- Ask for parameters according to choice of method

4- Generate according to parameters

5- Print groups in a loop, and Generate a file with output

}

method to read students file, with try catch

takes:

-path to file (hardcoded),

returns:

-a list of students

method to create output file, with try catch

takes:

-path to file (same input path),

method to pick a student from given list, and remove him from the list:

takes:

-list

returns:

-picked student

method to distribute students accross groups (according to nubmer of groups),

takes:

-list of students,

-number of groups,

-max number of studnets per group,

returns:

-list of group objects

method to distribute students accross groups (according to nubmer of studetns per group),

takes:

-list of students,

-nubmer of studetns per group,

-max number of studnets per group,

returns:

-list of group objects

}

group class {

list members (using max number per group)

toString {

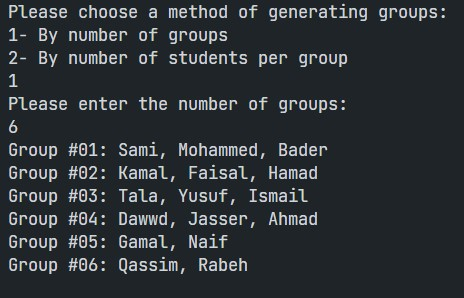
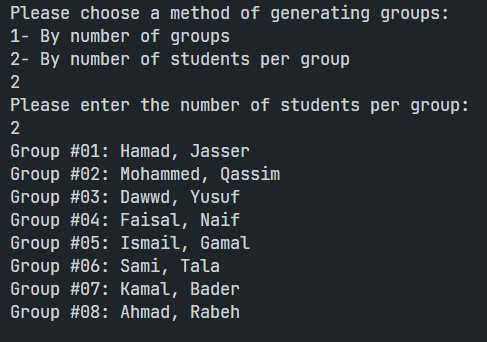
with printf formatting for spacing (using String.format method as return)

"group #: member1 , member2, ...

}

}

## **Testing** :



}

## **Code** :

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class App {

public static void main(String[] args) throws Exception {

Scanner scanner = new Scanner(System.in);

// 1- read file of path lib/Studnet List.txt and store in a list

List<String> students = readStudentsFile("Student List.txt");

// 2- Ask user for method of generating groups (by group number, or by studnets

// per group)

int methodOfChoice = askForGroupGenerationMethod(scanner);

// 3- Ask for parameters according to choice of method

int parameter = askForGroupGenerationParameters(scanner, methodOfChoice);

// 4- Check type of choice and Generate according to parameters

List<Group> groups = new ArrayList<Group>();

if (methodOfChoice == 1) {

groups = distributeViaGroups(students, parameter);

} else if (methodOfChoice == 2) {

groups = distributeViaMembers(students, parameter);

}

// 5- Print groups in a loop, and Generate a file with output

createOutputFile(groups, "Output.txt");

for (Group group : groups) {

System.out.println(group);

}

}

public static int askForGroupGenerationMethod(Scanner scanner) {

System.out.println("Please choose a method of generating groups:");

System.out.println("1- By number of groups");

System.out.println("2- By number of students per group");

int method = scanner.nextInt();

return method;

}

public static int askForGroupGenerationParameters(Scanner scanner, int method) {

if (method == 1) {

System.out.println("Please enter the number of groups:");

// int output = scanner.nextInt();

} else if (method == 2) {

System.out.println("Please enter the number of students per group:");

// int output = scanner.nextInt();

}

int output = scanner.nextInt();

return output;

}

public static List<String> readStudentsFile(String path) {

List<String> students = new ArrayList<String>();

try {

File file = new File(path);

Scanner scanner = new Scanner(file);

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

students.add(line);

}

scanner.close();

} catch (FileNotFoundException e) {

System.out.println("An error occurred, Couldn't create file.");

e.printStackTrace();

}

return students;

}

public static void createOutputFile(List<Group> groups, String path) {

try {

File file = new File(path);

PrintWriter writer = new PrintWriter(file);

for (Group group : groups) {

writer.println(group.toString());

}

writer.close();

} catch (FileNotFoundException e) {

System.out.println("An error occurred, Couldn't create file.");

e.printStackTrace();

}

}

public static String pickRandomStudent(List<String> students) {

int randomIndex = (int) (Math.random() \* students.size());

// System.out.println(randomIndex); /////

String pickedStudent = students.get(randomIndex);

students.remove(randomIndex);

return pickedStudent;

}

// \* This method distributes students

public static List<Group> distributeViaGroups(List<String> students, int numOfGroups) {

List<Group> groups = new ArrayList<Group>();

for (int i = 0; i < numOfGroups; i++) {

groups.add(i, new Group(i + 1));

}

for (int i = 0; students.size() > 0; i++) {

String student = pickRandomStudent(students);

groups.get(i % numOfGroups).addMember(student);

}

return groups;

}

// \* This method fills each group until it reaches the max students

public static List<Group> distributeViaMembers(List<String> students, int stuPerGroup) {

List<Group> groups = new ArrayList<Group>();

for (int iGroup = 0; students.size() > 0; iGroup++) {

groups.add(iGroup, new Group(iGroup + 1));

for (int iStudent = 0; (iStudent < stuPerGroup) && (students.size() > 0); iStudent++) { // Fills the

// required amount

// of students in

// Groups

// respectively

String student = pickRandomStudent(students);

groups.get(iGroup).addMember(student);

}

}

return groups;

}

}

class Group {

private List<String> members;

private int groupNumber;

public Group(int groupNumber) {

this.groupNumber = groupNumber;

this.members = new ArrayList<String>();

}

public void addMember(String member) {

this.members.add(member);

}

public String toString() {

String outputMembers = "";

for (String member : this.members) {

outputMembers += member + ", ";

}

outputMembers = outputMembers.substring(0, outputMembers.length() - 2);

return String.format("Group #%02d: %s", this.groupNumber, outputMembers);

}

}