



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №10
«Основи програмування»
Тема: Колекції. Списки

Гуменюк К.Е Тильна.М.С Любченко.І.М

Виконали:

студенти групи ІА-24
Любченко.І.М
Гуменюк.К.Е
Тильна.М.С

Хід роботи:

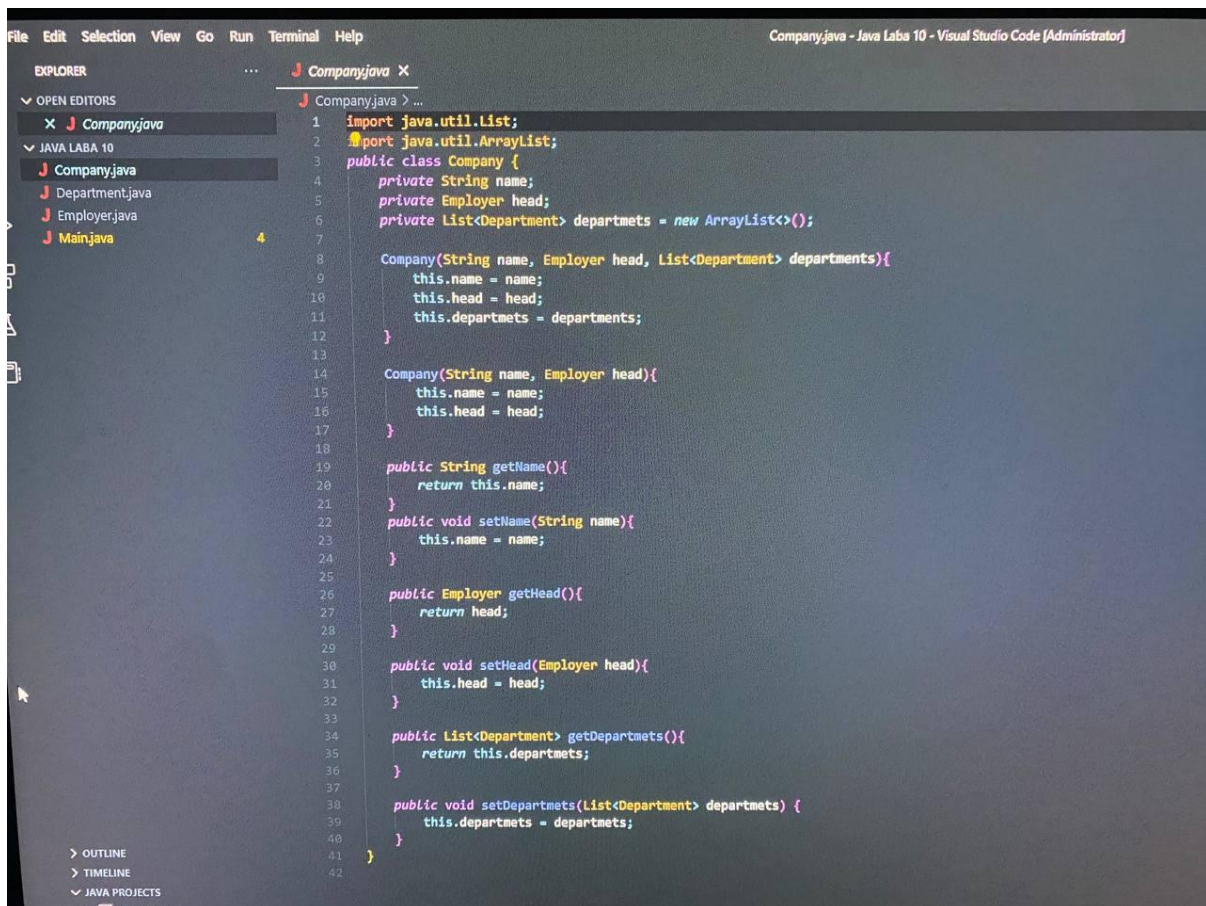
1. Ознайомитись з javadoc для наступних інтерфейсів та класів:

- Collection
- List
- ArrayList
- LinkedList
- Iterator
- RandomAccess

2. Виконати завдання з таблиці 2 відповідно до свого варіанту у таблиці 1. Для цього:

- проаналізувати завдання;
- створити зазначенні класи;
- для створення списків слід використовувати класи та інтерфейси з Collection Framework (заборонено використовувати масиви);
- усі списки мають бути типізованими (наприклад, `ArrayList<Student>`, а не просто `ArrayList`);
- при реалізації задач «1)», «2)», «3)» слід застосувати наступні методи перегляду колекцій у відповідності до свого варіанту (табл. 1):
 - a) нетипізований ітератор;
 - b) типізований ітератор;
 - c) типізований цикл «for-each».

3. Відповісти на контрольні питання



```
File Edit Selection View Go Run Terminal Help
Company.java - Java Labs 10 - Visual Studio Code [Administrator]

EXPLORER
OPEN EDITORS
Company.java X
JAVA LABS 10
Company.java
Department.java
Employer.java
Main.java

Company.java
1 import java.util.List;
2 import java.util.ArrayList;
3 public class Company {
4     private String name;
5     private Employer head;
6     private List<Department> departments = new ArrayList<>();
7
8     Company(String name, Employer head, List<Department> departments){
9         this.name = name;
10        this.head = head;
11        this.departments = departments;
12    }
13
14    Company(String name, Employer head){
15        this.name = name;
16        this.head = head;
17    }
18
19    public String getName(){
20        return this.name;
21    }
22    public void setName(String name){
23        this.name = name;
24    }
25
26    public Employer getHead(){
27        return head;
28    }
29
30    public void setHead(Employer head){
31        this.head = head;
32    }
33
34    public List<Department> getDepartments(){
35        return this.departments;
36    }
37
38    public void setDepartments(List<Department> departments) {
39        this.departments = departments;
40    }
41 }
42
```

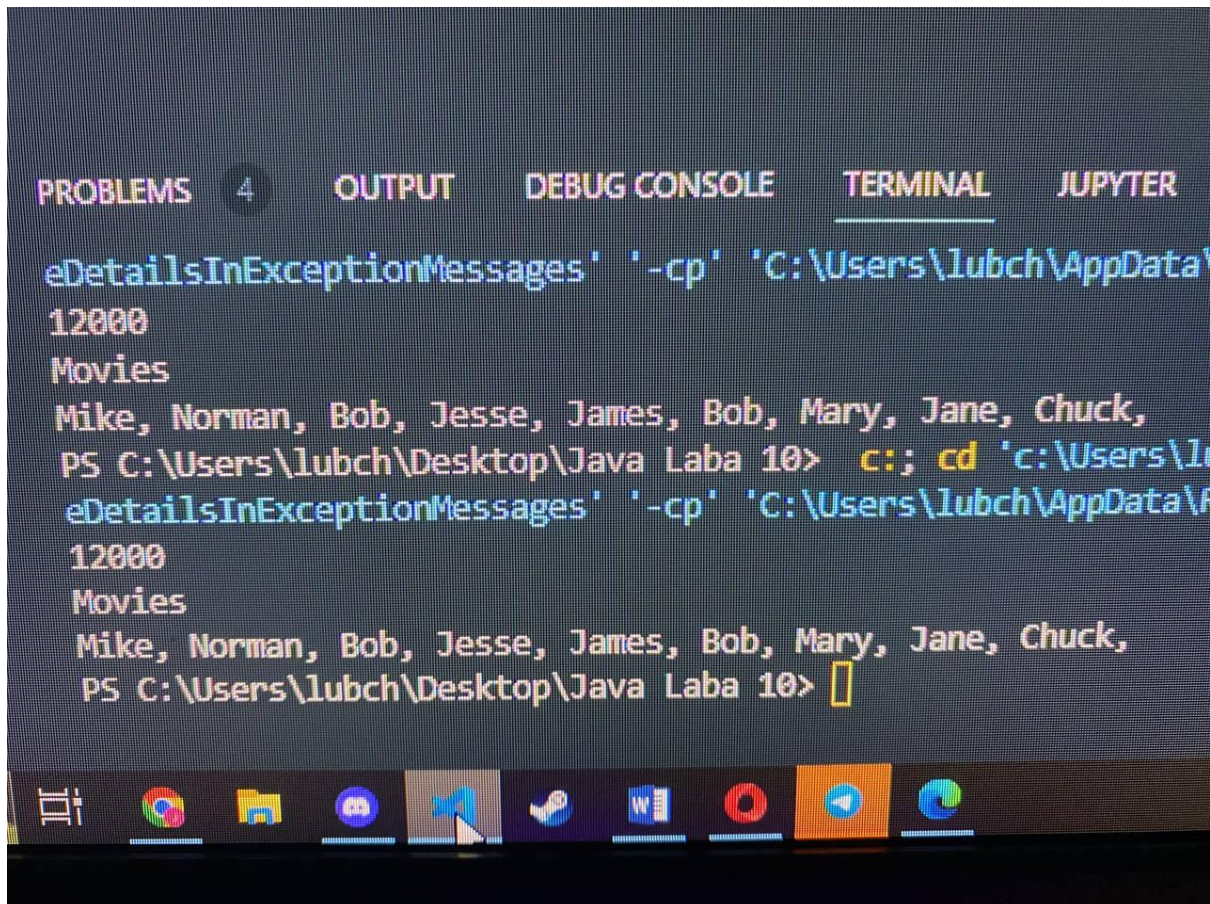
```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Department {
5     private String name;
6     private Employer manager;
7     private List<Employer> employers = new ArrayList<>();
8
9     Department(String name, Employer manager, List<Employer> employers){
10         this.name = name;
11         this.manager = manager;
12         this.employers = employers;
13     }
14
15     Department(String name, Employer manager){
16         this.name = name;
17         this.manager = manager;
18     }
19
20     public String getName(){
21         return this.name;
22     }
23
24     public void setName(String name) {
25         this.name = name;
26     }
27
28     public Employer getManager(){
29         return manager;
30     }
31
32     public void setManager(Employer manager) {
33         this.manager = manager;
34     }
35
36     public List<Employer> getEmployers(){
37         return this.employers;
38     }
39
40     public void setEmployers(List<Employer> employers) {
41         this.employers = employers;
42     }
43 }
44
```

```
1 public class Employer {
2     private String name;
3     private String surname;
4     private int salary;
5
6     Employer(String name, String surname, int salary){
7         this.name = name;
8         this.surname = surname;
9         this.salary = salary;
10    }
11    Employer(String name, String surname, int salary, Department department){
12        this.name = name;
13        this.surname = surname;
14        this.salary = salary;
15        department.getEmployers().add(Employer.this);
16    }
17
18    public String getName(){
19        return this.name;
20    }
21
22    public void setName(String name) {
23        this.name = name;
24    }
25
26    public String getSurname(){
27        return surname;
28    }
29
30    public void setSurname(String surname) {
31        this.surname = surname;
32    }
33
34    public int getSalary(){
35        return salary;
36    }
37
38    public void setSalary(int salary) {
39        this.salary = salary;
40    }
41 }
42
```



```
1 import java.util.ArrayList;
2 import java.util.Iterator;
3 import java.util.List;
4
5 public class Main {
6     private static int MaxSalary(Company company){
7         int maxSalary = 0;
8         for(Iterator iterator = EmployersList(company).iterator(); iterator.hasNext();){
9             Object employer = iterator.next();
10             if(employer instanceof Employer currentEmployer) {
11                 int current = currentEmployer.getSalary();
12                 if(current>maxSalary) {maxSalary=current;}
13             }
14         }
15         return maxSalary;
16     }
17
18     private static Department strangeDepartment(Company company){
19         for(Department department : company.getDepartments()){
20             for(Employer employer : department.getEmployers()){
21                 if(employer.getSalary()>department.getManager().getSalary()){
22                     return department;
23                 }
24             }
25         }
26         return null;
27     }
28
29     private static List<Employer> EmployersList(Company company){
30         List<Employer> EmployersList = new ArrayList<>();
31         EmployersList.add(company.getHead());
32         for(Iterator<Department> iterator = company.getDepartments().iterator(); iterator.hasNext();){
33             Department current = iterator.next();
34             EmployersList.add(current.getManager());
35             EmployersList.addAll(current.getEmployers());
36         }
37         return EmployersList;
38     }
39
40     private static void zavd1(Company company){
41         System.out.println(MaxSalary(company));
42     }
43     private static void zavd2(Company company){
44         System.out.println(strangeDepartment(company).getName());
45     }
46     private static void zavd3(Company company){
47         for (Employer i: EmployersList(company)) {
48             System.out.print(i.getName()+" ");
49         }
```

```
52 Run | Debug
53 public static void main(String[] args) {
54     Company Sony = new Company(name: "Sony", new Employer(name: "Mike", surname: "Ersantarov", salary: 10000));
55     Department Games = new Department(name: "Games", new Employer(name: "Norman", surname: "Osborn", salary: 5000));
56     Employer Bob = new Employer(name: "Bob", surname: "Oderkick", salary: 1000, Games);
57     Employer Jesse = new Employer(name: "Jesse", surname: "Pinkman", salary: 2000, Games);
58     Employer James = new Employer(name: "James", surname: "McGill", salary: 500, Games);
59
60     Department Movies = new Department(name: "Movies", new Employer(name: "Bob", surname: "Sponge", salary: 5000));
61     Movies.getEmployers().add(new Employer(name: "Mary", surname: "Oderkick", salary: 1000));
62     Movies.getEmployers().add(new Employer(name: "Jane", surname: "Overflow", salary: 12000));
63     Movies.getEmployers().add(new Employer(name: "Chuck", surname: "McGill", salary: 500));
64     Sony.getDepartments().add(Games);
65     Sony.getDepartments().add(Movies);
66
67     zavd1(Sony);
68     zavd2(Sony);
69     zavd3(Sony);
70
71 }
72 }
```

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
eDetailsInExceptionMessages' '-cp' 'C:\Users\lubch\AppData\
12000
Movies
Mike, Norman, Bob, Jesse, James, Bob, Mary, Jane, Chuck,
PS C:\Users\lubch\Desktop\Java Laba 10> c:; cd 'c:\Users\l
eDetailsInExceptionMessages' '-cp' 'C:\Users\lubch\AppData\
12000
Movies
Mike, Norman, Bob, Jesse, James, Bob, Mary, Jane, Chuck,
PS C:\Users\lubch\Desktop\Java Laba 10> 
```

Висновок: у цій роботі ми Ознайомитись з javadoc для наступних інтерфейсів та класів. Виконали завдання з таблиці 2 відповідно до свого варіанту у таблиці 1. Для цього: проаналізувати завдання; створити зазначенні класи; для створення списків використовували класи та інтерфейси з Collection Framework. А при реалізації задач «1)», «2)», «3)» застосували наступні методи перегляду колекцій у відповідності до свого варіанту (табл. 1): нетипізований ітератор; типізований ітератор; типізований цикл «for-each».