**Computer Language 2022**

**Assignment #7**

**Due: 23/May 23:59:59 (Questions No.1~No.4)**

**Due: 30/May 23:59:59 (ITM Project Part IV)**

**Student ID: 2110252 Name: Lee Jeong-Yun**

**ITM Engineering Co., Ltd. – Part IV (FINAL)**

* ITM Engineering Co., Ltd. intends to develop a system for managing employee information. The employees can be categorized into 1) Full-time and 2) Part-time. Write the final complete system to meet the following requirements.
  + Re-define classes using inheritance, polymorphism, and abstract class
  + Add advanced search and edit features
  + You can use the template code for Part IV or your own codebase.
  + Refer to the execution result screen at the bottom.
* Menu (Refer to MainMenuEnum enum)

1. Register → Register a new employee (RegisterMenuEnum enumeration)

11. Registration of full-time employees

12. Registration of part-time employees

19. Back

2. Search → Search for employees (SearchMenuEnum enumeration)

21. Full search

22. Fulltime search

23. Parttime search

24. Name search

29. Back

3. Edit → Update employee information (EditMenuEnum enumeration)

31. Edit employee information (by ID)

39. Back

9. Exit → End Program (until user types 9, the program continues)

* Information on the employee category
  + Fullt-ime

1. id (int) // employee id
2. name (string) // employee name
3. department (DepartmentEnum) // employee department
4. position (PositionEnum)
5. grade (int) // salary-level
   * Part-time
6. id (int) // employee id
7. name (string) // employee name
8. department (DepartmentEnum) // employee department
9. hours (int) // working hours

* Requirements
  + Define a super class for Fulltime and Parttime Employee classes
    - The super class must be an abstract class
    - The name of the superclass MUST be Employee
    - Employee class MUST have print() method
      * print() method should be declared as abstract
    - **Employee class MUST have getters/setters for its fields**
    - All fields of Employee class MUST have “protected” access modifier.
    - Use inheritance for redefining Fulltime and Parttime employee classes
  + EmployeeSet class MUST be able to manage all employees using a single array
    - In case of full employee search, print all the employee information in the order in which they were registered (use polymorphism here).
    - In case of full-time/part-time employee search, use “instanceof” operator to check the category of an employee.
  + Implement “advanced search” features
    - Fulltime employee search (print out fulltime employees only)
    - Parttime employee search (print out parttime employees only)
    - Search by employee name
    - SearchMenuEnum needs to be modified
  + Implement “edit” feature
    - Editing information of the employee specified by “ID” MUST be possible
    - EditMenuEnum needs to be declared
  + **Leave some comments for important code sections!**

**Expected output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**<continued>**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**<continued>**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**<continued>**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**<continued>**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

**<continued>**

텍스트이(가) 표시된 사진

자동 생성된 설명

* Submission

**\* Zip your project into a single file.**

**\* Make sure your zip file include all \*.java files for your project.**

**\* Submit your zipped file on time.**

**Q1. Write a program that takes a string line from the user and counts the words separated by whitespaces. The program continues until the user types “exit”.   
Use split() method of String class.**

**Output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Your code:**

import java.util.Scanner;  
  
public class WordCount {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 while (true) {  
 String strings = scanner.nextLine();  
 if (strings.equals("exit")) {  
 System.*out*.println("Exit.");  
 break;  
 }  
 String[] words = strings.split(" ");  
 System.*out*.println("The number of words is "+words.length);  
 }  
 scanner.close();  
 }  
}

**Your result (screenshot):**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

**scanner would get strings by nextLine(). If the variable ‘strings’ get ‘exit’, then break the loop. ‘strings’ is split by whitespace. String array ‘words’ would have elements each including a single word. Array’s length is number of words.**

**Q2. Write a program that takes an “add equation” from the user and prints out the result.   
Use StringTokenizer.**

**Output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**Your code:**

import java.util.Scanner;  
import java.util.StringTokenizer;  
  
public class Operation {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 String input = scanner.nextLine();  
 input = input.replace(" ","");  
 StringTokenizer line = new StringTokenizer(input, "+");  
 int sum = 0;  
 while (line.hasMoreTokens()) {  
 String str = line.nextToken();  
 int num = Integer.*parseInt*(str);  
 sum = sum + num;  
 }  
 System.*out*.println("Result is "+sum);  
 }  
}

**Your result (screenshot):**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

**input variable would get strings by scanner.nextLine() method. Using replace() method to remove whitespace. And StringTokenizer would distinguish by “+” and then, String tokens are converted to Integer type. Calculate the summation of each element using variable converted type from String to int.**

**Q3. Write a program using HashMap to manage customer names and point scores. This program manages customer points in a cumulative way. Once one customer data is inserted, information of all the customers so far should be printed out. This program continues until the user types “exit”.**

**Output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Your code:**

import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Scanner;  
import java.util.Set;  
  
public class PointManagementSystem {  
 public static void main(String[] args) {  
 System.*out*.println("\*\* Point Management System \*\*");  
 Scanner scanner = new Scanner(System.*in*);  
 HashMap<String, Integer> data = new HashMap<>();  
 while (true) {  
 System.*out*.print("Input your name and point>> ");  
  
 String name = scanner.next();  
 if (name.equals("exit")) {  
 System.*out*.println("Exit the program.");  
 break;  
 }  
  
 Integer point = scanner.nextInt();  
  
 if (data.containsKey(name)) {  
 point = data.get(name) + point;  
 data.put(name, point);  
 }  
 else {  
 data.put(name, point);  
 }  
  
 Set<String> keys = data.keySet();  
 Iterator<String> iterator = keys.iterator();  
 while (iterator.hasNext()) {  
 String itName = iterator.next();  
 int itPoint = data.get(itName);  
 System.*out*.print("("+itName+","+itPoint+")");  
 }  
 System.*out*.println("");  
 }  
 scanner.close();  
 }  
}

**Your result (screenshot):**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

**Declaring HashMap instance called data, key is String type and value is Integer type. Using Scanner, the key and value are inserted into the HashMap. If the key input is “exit”, then break the infinite loop. If the input key is already exist in the data, then add the value to existing value. To get the all stored information in the data, the data is converted to Set<String> and its iterator. Using Iterator, all the key and value is printed out.**

**Q4. Write a program that can take student grades and then compute the minimum one. Our main code uses Collections.min() method to get the minimum one in the Student collection, however, no ideas on how to compare Student instances are given. Therefore, you MUST implement StudentComparator class which provides a comparison operation regarding our Student class based on the grade!**

**- As described in the main code, Collections.min() can take a custom comparator instance to compare the items in the given collection. In our case, we need a comparator instance to compare student items in the vector.**

**- StudentComparator must implement Comparator<T> interface to meet the goal.**

**- A class implementing Comparator<T> interface should have a method that takes two arguments and compares them for order.**

**- Code in the main method and Student class cannot be modified.  
- Refer to the Java API reference to study more about Comparator<T> interface (**[**https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Comparator.html**](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Comparator.html)**).**

class Student{  
 int grade=0;  
 Student(int grade){  
 this.grade = grade;  
 }  
}  
  
**class StuduentComparator {  
  
 *// This class must be implemented  
 // Search for how Comparator<T> works!*}**public class Ex04 {  
 public static void main(String[] args) {  
  
 Scanner scn = new Scanner(System.*in*);  
 Vector<Student> vs = new Vector<>(4);  
  
 for(int i=0;i<4;i++){  
 System.*out*.print("Input S"+i+"'s grade: ");  
 vs.add(new Student(scn.nextInt()));  
 }  
  
 StuduentComparator stucomp = new StuduentComparator();  
 System.*out*.println("Minimum grade: "+Collections.*min*(vs,stucomp).grade);  
  
 }  
  
}

**Output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**Your code:**

import java.util.Collections;  
import java.util.Comparator;  
import java.util.Scanner;  
import java.util.Vector;  
  
class Student {  
 int grade = 0;  
 Student (int grade) {  
 this.grade = grade;  
 }  
}  
  
class StudentComparator implements Comparator<Student> {  
 public int compare(Student o1, Student o2) {  
 if (o1.grade > o2.grade)  
 return 1;  
 else if (o1.grade == o2.grade)  
 return 0;  
 else  
 return -1;  
 }  
}  
  
public class Compare {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Vector <Student> vs = new Vector<>(4);  
  
 for (int i=0; i<4; i++) {  
 System.*out*.print("Input S"+i+"'s grade: ");  
 vs.add(new Student(scanner.nextInt()));  
 }  
  
 StudentComparator stucomp = new StudentComparator();  
 System.*out*.println("Minimum grade: "+  
 Collections.*min*(vs, stucomp).grade);  
 }  
}

**Your result (screenshot):**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

**Student class is defined. The class has int grade variables and constructor which get int parameter and the variable is set by this.grade keyword.**

**Declaring StudentComparator class implements Comparator<String> interface. The class overrides compare() method which get 2 parameters type Student. If the o1.grade is larger than o2.grade, then return 1. If both is the same, then return 0. If o1.geade is smaller than o2.grade, then return -1. In main method, Student type Vector is instantiated with capacity 4. Scanner add points by using loop statement into declared Vector, vs. To find out the minimal value of the 4 input, Collections.min(collection, comparator) is introduced. For the collection, vs is used. And for the comparator, stucomp is used. stucomp is instantiated for StudentComparator type.**