Exercise 1: Collection

Question 1: List

Tạo 1 student có property id, name (trong đó có 3 student có name trùng nhau, id sẽ là auto increment)
Khai báo 1 ArrayList students, sau đó
➔Tạo Class Student

```
package com.vti.entity;

public class Student {
	public static int COUNT = 0;
	private int id;
	private String name;
	public Student( String name) {
		super();
		this.id = ++COUNT;
		this.name = name;
	}
	@Override
	public String toString() {
		return "Student [id=" + id + ", name=" + name + "]";
	}
	/**
	 * @return the id
	 */
	public int getId() {
		return id;
	}
	/**
	 * @param id the id to set
	 */
	public void setId(int id) {
		this.id = id;
	}
	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}
	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}


}
```

➔Tạo Class Exercise1 trong backend

```
package com.vti.backend;

import java.util.ArrayList;
import java.util.Collections;
```

```java
import java.util.List;

import org.w3c.dom.ls.LSInput;

import com.vti.entity.Student;
import com.vti.ultis.ScannerUltis;

public class Exercise1 {
	private java.util.List<Student> listStudent;

	public Exercise1() {
		listStudent = new ArrayList<Student>();
		System.out.println("Nhập số sinh viên muốn thêm: ");
		int num = ScannerUltis.inputIntPositive();
		for (int i = 0; i < num - 3; i++) {
			Student st = new Student("Student " + (i + 1));
			listStudent.add(st);
		}
		Student st1 = new Student("Student Name");
		Student st2 = new Student("Student Name");
		Student st3 = new Student("Student Name");
		listStudent.add(st1);
		listStudent.add(st2);
		listStudent.add(st3);
	}

	public void question1() {
		loadMenuQues1();
		while (true) {
			loadMenuQues1();
			int chose = ScannerUltis.inputInt();
			switch (chose) {

			case 1:
				System.out.println("Tổng số các sinh viên là: " +
listStudent.size());
				printStudent();
				break;
			case 2:
				System.out.println("Phần tử thứ 4 là : " +
listStudent.get(3));
				break;
			case 3:
				System.out.println("Phần tử đầu là : " +
listStudent.get(0));
				System.out.println("Phần tử cuối là : " +
listStudent.get(listStudent.size() - 1));
				break;
			case 4:
				int choseCase4 = choseSubMenu4();
				switch (choseCase4) {
				case 1:
					System.out.println("Nhập tên của sinh viên cần
thêm");
					String name1 = ScannerUltis.inputString();
					listStudent.add(0, new Student(name1));
					printStudent();
					break;
				case 2:
```

```
                                    System.out.println("Nhập tên của sinh viên cần
thêm");
                                    String name2 = ScannerUltis.inputString();
                                    listStudent.add(new Student(name2));
                                    printStudent();
                                    break;

                              }
                              break;
                        case 5:
                              System.out.println("Nhập tên của sinh viên cần
thêm");
                              String name5 = ScannerUltis.inputString();
                              listStudent.add(new Student(name5));
                              printStudent();
                              break;
                        case 6:
                              Collections.reverse(listStudent);
                              System.out.println("Đã đảo ngược vị trí ");
                              printStudent();
                              break;
                        case 7:
                              System.out.println("Nhập vào ID cần tìm kiếm:");
                              int id7 = ScannerUltis.inputIntPositive();
                              for (Student student : listStudent) {
                                    if (student.getId() == id7) {
                                          System.out.println(student);
                                    }
                              }
                              break;
                        case 8:
                              System.out.println("Nhập vào name cần tìm kiếm:");
                              String name8 = ScannerUltis.inputString();
                              for (Student student : listStudent) {
                                    if (student.getName().equals(name8)) {
                                          System.out.println(student);
                                    }
                              }
                              break;
                        case 9:
                              System.out.println("Các sinh viên trùng tên: ");
                              for (int i = 0; i < listStudent.size(); i++) {
                                    for (int j = i + 1; j < listStudent.size();
j++) {
                                          if
(listStudent.get(i).getName().equals(listStudent.get(j).getName())) {

      System.out.println(listStudent.get(i).toString());
                                          }
                                    }
                              }
                              break;
                        case 10:
                              System.out.println("Nhập vào ID cần xóa tên:");
                              int id10 = ScannerUltis.inputIntPositive();
                              for (Student student : listStudent) {
                                    if (student.getId() == id10) {
                                          student.setName(null);
                                    }
```

```
                        }
                        printStudent();
                        break;
                case 11:
                        System.out.println("Nhập vào ID của student cần
xóa:");
                        int id11 = ScannerUltis.inputIntPositive();
                        listStudent.removeIf(student -> student.getId() ==
id11);
                        printStudent();
                        break;

                case 12:
                        System.out.println("Tạo mới ArrayCopy:");
                        List<Student> arrayCopy = new ArrayList<Student>();
                        arrayCopy.addAll(listStudent);
                        System.out.println("In phần tử trong ArrayCopy: ");
                        for (Student student : arrayCopy) {
                                System.out.println(student);
                        }
                        break;
                case 13:
                        return;
                default:
                        System.out.println("Hãy chọn đúng menu");
                        break;
                }
        }
    }

    private void loadMenuQues1() {
        System.out.println(
                        "---     Lựa chọn chức năng muốn sử dụng
---");
        System.out.println(
                        "---     1.In ra tổng số phần tử của students
---");
        System.out.println(
                        "---     2.Lấy phần tử thứ 4 của students
---");
        System.out.println(
                        "---     3.In ra phần tử đầu và phần tử cuối của
students                                        ---");
        System.out.println(
                        "---     4.Thêm 1 phần tử vào vị trí đầu hoặc cuối
của students                                    ---");
        System.out.println(
                        "---     5.Thêm 1 phần tử vào vị trí cuối của
students                                        ---");
        System.out.println(
                        "---     6.Đảo ngược vị trí của students
---");
        System.out.println(
                        "---     7.Tạo 1 method tìm kiếm student theo id
---");
        System.out.println(
                        "---     8.Tạo 1 method tìm kiếm student theo name
---");
        System.out.println(
```

```
                        "---     9.Tạo 1 method để in ra các student có trùng
tên                                     ---");
            System.out.println(
                        "---     10.Xóa name của student có id = 2;
---");
            System.out.println(
                        "---     11.Delete student có id = 5;
---");
            System.out.println(
                        "---     12.Tạo 1 ArrayList tên là studentCopies và
add tất cả students vào studentCopies         ---");
            System.out.println(
                        "---     13. Exit
---");

      }

      private void printStudent() {
            for (Student student : listStudent) {
                  System.out.println(student);
            }

      }

      private int choseSubMenu4() {
            System.out.println("Chọn 1.Thêm vào đầu, 2.Thêm vào cuối");
            while (true) {
                  int chose1 = ScannerUltis.inputIntPositive();
                  switch (chose1) {
                  case 1:

                        return chose1;
                  case 2:

                        return chose1;

                  default:
                        System.out.println("Chọn lại: ");
                  }
            }
      }

}
```

➔Tạo Class Demo_Exercise1 chứa main để demo

```
package com.vti.frontend;

import com.vti.backend.Exercise1;

public class Demo_Exercise1 {
      public static void main(String[] args) {
            Exercise1 ex1 = new Exercise1();
            ex1.question1();
      }
}
```

a) In ra tổng số phần tử của students
Viết hàm này trong Class Exercise1 để xử lý logic

```
	case 1:
				System.out.println("Tổng số các sinh viên là: " +
listStudent.size());
				printStudent();
				break;
```

b) Lấy phần tử thứ 4 của students, viết hàm trong Class ListQ1

```
	case 2:
				System.out.println("Phần tử thứ 4 là : " +
listStudent.get(3));
				break;
```

c) In ra phần tử đầu và phần tử cuối của students

```
	case 3:
				System.out.println("Phần tử đầu là : " +
listStudent.get(0));
				System.out.println("Phần tử cuối là : " +
listStudent.get(listStudent.size() - 1));
				break;
```

d) Thêm 1 phần tử vào vị trí đầu hoặc của students

```
case 4:
				int choseCase4 = choseSubMenu4();
				switch (choseCase4) {
				case 1:
					System.out.println("Nhập tên của sinh viên cần
thêm");
					String name1 = ScannerUltis.inputString();
					listStudent.add(0, new Student(name1));
					printStudent();
					break;
				case 2:
					System.out.println("Nhập tên của sinh viên cần
thêm");
					String name2 = ScannerUltis.inputString();
					listStudent.add(new Student(name2));
					printStudent();
					break;

				}
				break;
```

choseSubMenu4 ();

```
	private int choseSubMenu4() {
		System.out.println("Chọn 1.Thêm vào đầu, 2.Thêm vào cuối");
		while (true) {
			int chose1 = ScannerUltis.inputIntPositive();
			switch (chose1) {
			case 1:

				return chose1;
			case 2:
```

```
                        return chose1;

                    default:
                        System.out.println("Chọn lại: ");
                    }
                }
            }
```

e) Thêm 1 phần tử vào vị trí cuối của students

```
case 5:
                        System.out.println("Nhập tên của sinh viên cần
thêm");
                        String name5 = ScannerUltis.inputString();
                        listStudent.add(new Student(name5));
                        printStudent();
                        break;
```

f) Đảo ngược vị trí của students

```
case 6:
                        Collections.reverse(listStudent);
                        System.out.println("Đã đảo ngược vị trí ");
                        printStudent();
                        break; printStudent();
        }
```

g) Tạo 1 method tìm kiếm student theo id

```
        case 7:
                        System.out.println("Nhập vào ID cần tìm kiếm:");
                        int id7 = ScannerUltis.inputIntPositive();
                        for (Student student : listStudent) {
                            if (student.getId() == id7) {
                                System.out.println(student);
                            }
                        }
                        break;
```

h) Tạo 1 method tìm kiếm student theo name

```
            case 8:
                        System.out.println("Nhập vào name cần tìm kiếm:");
                        String name8 = ScannerUltis.inputString();
                        for (Student student : listStudent) {
                            if (student.getName().equals(name8)) {
                                System.out.println(student);
                            }
                        }
                        break;
```

i) Tạo 1 method để in ra các student có trùng tên

```
case 9:
                        System.out.println("Các sinh viên trùng tên: ");
                        for (int i = 0; i < listStudent.size(); i++) {
                            for (int j = i + 1; j < listStudent.size();
j++) {
```

```
                                        if
(listStudent.get(i).getName().equals(listStudent.get(j).getName())) {

        System.out.println(listStudent.get(i).toString());
                                        }
                                }
                        }
                        break;
```

j) Xóa name của student có id = 2;

```
case 10:
                        System.out.println("Nhập vào ID cần xóa tên:");
                        int id10 = ScannerUltis.inputIntPositive();
                        for (Student student : listStudent) {
                                if (student.getId() == id10) {
                                        student.setName(null);
                                }
                        }
                        printStudent();
                        break;
```

k) Delete student có id = 5;

```
case 11:
                        System.out.println("Nhập vào ID của student cần
xóa:");
                        int id11 = ScannerUltis.inputIntPositive();
                        listStudent.removeIf(student -> student.getId() ==
id11);
                        printStudent();
                        break;
```

l) Tạo 1 ArrayList tên là studentCopies và add tất cả students vào studentCopies

```
        case 12:
                        System.out.println("Tạo mới ArrayCopy:");
                        List<Student> arrayCopy = new ArrayList<Student>();
                        arrayCopy.addAll(listStudent);
                        System.out.println("In phần tử trong ArrayCopy: ");
                        for (Student student : arrayCopy) {
                                System.out.println(student);
                        }
                        break;
```

Question 2 (Optional): Stack & Queue

Khai báo 1 danh sách lưu các tên học sinh tới tham dự phỏng vấn, thứ tự tới của các học sinh như sau:
Nguyễn Văn Nam, Nguyễn Văn Huyên, Trần Văn Nam, Nguyễn Văn A

➔Sử dụng Class Student đã tạo ở bước trên để tạo các đối tượng Student cụ thể

➔Tạo danh sách sinh viên sử dụng Stack:

```
case 1:
                        System.out.println("Nhập số sinh viên muốn tạo: ");
                        int num2 = ScannerUltis.inputIntPositive();
                        for (int i = 0; i < num2; i++) {
```

```
                                        Student st = new Student("Student Ques2 " + (i
+ 1));
                                        stackStudent.push(st);
                                }
                                System.out.println("Tạo thành công " + num2 + " sinh
viên.");
                                break;
```

a) Tạo 1 method để in ra thứ tự tới của các học sinh theo thứ tự sớm nhất tới muộn nhất (gợi ý dùng Stack) ➔ Trong bài tập này sử dụng Iterator để duyệt qua các phần tử trong Stacs, chú ý, khi duyệt thì Iterator sẽ duyệt từ đáy Stack đến đỉnh.

```
        case 2:
                                System.out.println("Thứ tự sinh viên theo thứ tự từ
sớm nhất đến muộn nhất dùng Stack: ");
                                Iterator<Student> iteratorStudent =
stackStudent.iterator();
                                while (iteratorStudent.hasNext()) {
                                        Student st = iteratorStudent.next();
                                        System.out.println(st);
                                }
                                break;
```

Lấy ra sinh viên đến muộn nhất. ➔Sử dụng hàm peek() của stack

Lấy danh sách sinh viên theo thứ tự từ các học sinh theo thứ tự muộn nhất tới sớm nhất

```
case 3:
                                System.out.println("Sinh viên đến muộn nhất: " +
stackStudent.peek());
                                System.out.println("Thứ tự sinh viên theo thứ tự từ
muộn nhất đến sớm nhất dùng Stack: ");
                                for (int i = 0; i < stackStudent.size(); i++) {
                                        System.out.println(stackStudent.pop());
                                }
                                System.out.println("Số sinh viên còn lại trong Stack:
" + stackStudent.size());
                                break;
```

b) Tạo 1 method để in ra thứ tự tới của các học sinh theo thứ tự từ sớm nhất tới muộn nhất (gợi ý dùng Queue) ➔

boolean add(E e): Thêm phần tử vào Queue. Nếu thêm thành công trả về true, ngược lại false.

E element(): Trả về phần tử đầu tiên của Queue.

boolean offer(object): Tương tự add().

E remove(): Xoá phần tử đầu tiên của Queue.

E poll(): Tương tự remove(), điểm khác nhau giữa poll() và remove(), poll() trả về null khi Queue rỗng còn remove() quăng exception NoSuchElementException.

E peek(): Tương tự element() điểm khác nhau giữa peek() và element() là peek() trả về null nếu Queue rỗng, element() quăng NoSuchElementException

```
case 4:
                                Queue<Student> studentQueue = new
LinkedList<Student>();
```

```
                        System.out.println("Nhập vào số sinh viên muốn quản
lý bằng Queue: ");
                        int num4 = ScannerUltis.inputIntPositive();
                        for (int i = 0; i < num4; i++) {
                              Student st4 = new Student("Student Queue " + (i
+ 1));
                              studentQueue.add(st4);
                        }
                        System.out.println("Đã thêm " + num4 + " sinh viên
vào queue thành công.");
                        System.out.println("Danh sách sinh viên vừa thêm: " +
studentQueue.toString());
                        System.out.println(
                                    "Danh sách sinh viên theo thứ tự từ sớm
nhất tới muộn nhất dùng Poll để lấy dữ liệu: ");
                        for (int i = 0; i < num4; i++) {
                              System.out.println(studentQueue.poll());
                        }
                        System.out.println("Danh sách sinh viên còn lại trong
Queue: " + studentQueue);
                        break;
```

Question 3 (Optional): Set

Tạo 1 student có property id, name

Khai báo 1 Set students, sau đó làm các chức năng tương tự List

Tạo 1 student có property id, name (trong đó có 3 student có name trùng nhau, id sẽ là auto increment)

Khai báo 1 Set students, sau đó ➔ Trong bài tập này sử dụng HashSet để demo, do trong HashSet không tính đến số thứ tự khi Add vào nên không thể thêm được phần tử vào đầu hay vào cuối của Set.

```
Set<Student> studentSet = new HashSet<Student>();
```

Tạo danh sách SET:

```
      case 1:
                        System.out.println("Nhập số sinh viên muốn tạo: ");
                        int num2 = ScannerUltis.inputIntPositive();
                        for (int i = 0; i < num2; i++) {
                              Student stSet = new Student("Student Ques3 " +
(i + 1));
                              studentSet.add(stSet);
                        }
                        System.out.println("Tạo thành công " +
studentSet.size() + " sinh viên.");
                        Iterator<Student> iteratorSet =
studentSet.iterator();
                        for (int i = 0; i < studentSet.size(); i++) {
                              System.out.println(iteratorSet.next());
                        }
                        break;
```

a) In ra tổng số phần tử của students

```
case 2:
                        System.out.println("Tổng số phần tử trong
Set: " + studentSet.size());
                        break;
```

b) Lấy phần tử thứ 4 của students

```
case 3:
                        System.out.println("Phần tử thứ 4 trong Set: ");
                        Iterator<Student> iteratorSet3 =
studentSet.iterator();
                        for (int i = 0; i <= 2; i++) {
                              iteratorSet3.next();
                        }
                        System.out.println(iteratorSet3.next());
                        break;
```

c) In ra phần tử đầu và phần tử cuối của students

```
case 4:
                        Iterator<Student> iteratorSet4 =
studentSet.iterator();
                        System.out.println("Phần tử đầu của Set: " +
iteratorSet4.next());
                        for (int i = 0; i < studentSet.size() - 2; i++) {
                              iteratorSet4.next();
                        }
                        System.out.println("Phần tử cuối của Set: " +
iteratorSet4.next());
                        break;
```

d) Thêm 1 phần tử vào vị trí đầu của students

Hiện tại chưa tìm được giải pháp để thêm

e) Thêm 1 phần tử vào vị trí cuối của students, Sau khi Add thêm thì phần tử không nằm ở cuối.

```
case 5:
                        Student stEnd = new Student("Student Ques3 End");
                        studentSet.add(stEnd);
                        System.out.println("Danh sách trong Set: ");
                        Iterator<Student> iteratorSet5 =
studentSet.iterator();
                        for (int i = 0; i < studentSet.size(); i++) {
                              System.out.println(iteratorSet5.next());
                        }
                        break;
```

```
5
Danh sách trong Set:
Student [id=5, name=Student Ques3 5]
Student [id=6, name=Student Ques3 6]
Student [id=8, name=Student Ques3 8]
Student [id=7, name=Student Ques3 7]
Student [id=3, name=Student Ques3 3]
Student [id=2, name=Student Ques3 2]
Student [id=11, name=Student Ques3 End]
Student [id=9, name=Student Ques3 9]
Student [id=4, name=Student Ques3 4]
Student [id=1, name=Student Ques3 1]
Student [id=10, name=Student Ques3 10]
```

f) Đảo ngược vị trí của students

g) Tạo 1 method tìm kiếm student theo id

```
case 6:
                        System.out.println("Nhập vào ID cần tìm kiếm: ");
                        int id6 = ScannerUltis.inputIntPositive();
                        Iterator<Student> i6 = studentSet.iterator();
                        for (int i = 0; i < studentSet.size(); i++) {
                                Student stFind = i6.next();
                                if (stFind.getId() == id6) {
                                        System.out.println(stFind);
                                }
                        }
                        break;
```

h) Tạo 1 method tìm kiếm student theo name

```
case 7:
                        Student stFind = new Student("daonq");
                        studentSet.add(stFind);
                        System.out.println("Nhập vào name cần tìm kiếm: ");
                        String name7 = ScannerUltis.inputString();
                        Iterator<Student> i7 = studentSet.iterator();
                        for (int i = 0; i < studentSet.size(); i++) {
                                Student stFind7 = i7.next();
                                if (stFind7.getName().equals(name7)) {
                                        System.out.println(stFind7);
                                }
                        }
                        break;
```

Tạo 1 method để in ra các student có trùng tên

j) Xóa name của student có id = 2;

```
        case 9:
                        System.out.println("Nhập vào id của sinh viên cần xóa
tên: ");
                        int id9 = ScannerUltis.inputIntPositive();

                        System.out.println("Set sau khi xóa: ");
                        Iterator<Student> i9 = studentSet.iterator();
                        for (int i = 0; i < studentSet.size(); i++) {
                                Student stFind9 = i9.next();
                                if (stFind9.getId() == id9) {
                                        stFind9.setName(null);
                                }
                        }
                        Iterator<Student> i10 = studentSet.iterator();

                        for (int i = 0; i < studentSet.size(); i++) {
                                System.out.println(i10.next());
                        }
```

```
                        break;
```

k) Delete student có id = 5;

```
case 8:
                        System.out.println("Nhập vào ID của sinh viên cần xóa
khỏi danh sách: ");
                        int id8 = ScannerUltis.inputIntPositive();
                        studentSet.removeIf(student -> student.getId() ==
id8);
                        System.out.println("Set sau khi xóa: ");
                        Iterator<Student> i8 = studentSet.iterator();
                        for (int i = 0; i < studentSet.size(); i++) {
                              System.out.println(i8.next());
                        }
                        break
```

l) Tạo 1 Set tên là studentCopies và add tất cả students vào studentCopies

```
case 10:
                        System.out.println("Tạo Set mới để copy sang:");
                        Set<Student> studentSetCopy = new HashSet<Student>();
                        studentSetCopy.addAll(studentSet);
                        System.out.println("Copy thành công");
                        System.out.println("In danh sách phần tử trong
studentSetCopy: ");
                        Iterator<Student> i10_1 = studentSetCopy.iterator();
                        for (int i = 0; i < studentSetCopy.size(); i++) {
                              System.out.println(i10_1.next());
                        }
                        break;
```

Question 4 (Optional): Set

Tạo 1 danh sách có chứa name của các student không trùng nhau

In ra set đó

```
case 1:
                        System.out.println("Nhập số lượng sinh viên: ");
                        int num45 = ScannerUltis.inputIntPositive();
                        for (int i = 0; i < num45; i++) {
                              System.out.println("Nhập vào ID: ");
                              int id45 = ScannerUltis.inputIntPositive();
                              System.out.println("Nhập vào tên sinh viên");
                              String name45 = ScannerUltis.inputString();
                              StudentQues45 st45 = new StudentQues45(id45,
"Student_" + name45);
                              studentQues45.add(st45);
                        }
                        System.out.println("Tổng số sinh viên là: " +
studentQues45.size());
                        Iterator<StudentQues45> i1 =
studentQues45.iterator();
                        for (int i = 0; i < studentQues45.size(); i++) {
                              System.out.println(i1.next());
                        }
                        break;
```

➔ Trường hợp này vẫn add được các dữ liệu trùng cả id và name do đang khác địa chỉ ô nhớ.

Question 5 (Optional): Set

Tạo 1 danh sách có chứa name của các student không trùng nhau.
Sắp xếp theo name và in ra set đó

Trong class StudentQues45 cần cài đặt Interface Coparable để so sánh. Và overide lại phương thức compareTo

```
package com.vti.entity;

public class StudentQues45 implements Comparable<StudentQues45> {
	private int id;
	private String name;

	public StudentQues45(int id, String name) {
		super();
		this.id = id;
		this.name = name;
	}

	@Override
	public String toString() {
		return "Student [id=" + id + ", name=" + name + "]";
	}

	/**
	 * @return the id
	 */
	public int getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(int id) {
		this.id = id;
	}

	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	@Override
	public int compareTo(StudentQues45 o) {

		return this.name.compareTo(o.getName());
	}
```

```
}
```

```
	case 2:
				List<StudentQues45> studentlist45 = new
ArrayList<StudentQues45>(studentQues45);
				Collections.sort(studentlist45);
				for (StudentQues45 studentQues452 : studentlist45) {
					System.out.println(studentQues452);
				}
				break;
```

Question 6: Map

Để thay thế 1 object ta có thể tạo 1 map tên là students có key = id của student , value là name của students.

```
Map<Integer, String> studentMap = new HashMap<Integer, String>();
		while (true) {
			loadMenuEx1Ques67();
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				System.out.println("Nhập số sinh viên muốn tạo: ");
				int num1 = ScannerUltis.inputIntPositive();

				for (int i = 0; i < num1; i++) {
					System.out.println("Nhập vào Student " + (i +
1) + ": ");

					System.out.println("Nhập vào tên: ");
					String name = ScannerUltis.inputString();
					studentMap.put(COUNT67++, name);
				}
				System.out.println("Danh sách sinh viên trong MAP:
");

				for (Map.Entry<Integer, String> mapStudent :
studentMap.entrySet()) {
					System.out.println("ID: " + mapStudent.getKey()
+ " Name: " + mapStudent.getValue());
				}
				break;
```

Question 7 (Optional): tiếp tục question 6

Thực hiện các chức năng sau:

a.In ra các key của students

```
	case 2:
				System.out.println("In ra các key trong Map:");
				for (Map.Entry<Integer, String> student :
studentMap.entrySet()) {
					System.out.println(student.getKey());
				}
				break;
```

b.In ra value của students

```
case 3:
				System.out.println("In ra các Value trong
Map:");
				for (Map.Entry<Integer, String> student :
studentMap.entrySet()) {
```

```
            System.out.println(student.getValue());
                        }
                        break;
```

c) In ra danh sách students được sắp xếp theo tên của student

Chưa có đáp án

d) Chuyển đổi map students sang set

Chưa có đáp án

Exercise 2 (Optional): Comparing

Tạo 1 students có property id, name, ngày sinh, điểm và tạo 5 student

➔Sử dụng lại class Student phía bên trên

➔Tạo Class để xử lý logic của chương trình:

➔Tạo class StudentEx2 , Class này phải cài đặt interface **implements** Comparable<StudentEx2> để thực hiện so sánh, trong này sẽ cài đặt các phương thức để comapre như: chỉ sắp xếp theo name, sắp xếp theo name+ score, sắp xếp theo name + Score + localdate:

+ Với name sẽ có hàm `this.name.compareTo(o.getName());` để so sánh 2 chuỗi ký tự theo unicode.

+ Với Localdate sẽ có hàm `this.birthDay.compareTo(o.getBirthDay());`để so sánh

+ Với số sẽ viết dạng if else để so sánh lớn hơn nhỏ hơn

Question 1: Comparable

In ra học sinh sắp xếp theo name

Question 2: Comparator

In ra học sinh sắp xếp theo name, nếu tên trùng nhau thì sẽ sắp xếp theo ngày sinh, nếu ngày sinh trùng nhau thì sẽ sắp xếp theo điểm

```
package com.vti.entity;

import java.time.LocalDate;

public class StudentEx2 implements Comparable<StudentEx2> {
	public static int COUNTEX2 = 0;
	private int id;
	private String name;
	private LocalDate birthDay;
	private int score;

	public StudentEx2(String name, LocalDate birthDay, int score) {
		super();
		this.id = ++COUNTEX2;
		this.name = name;
		this.birthDay = birthDay;
		this.score = score;
	}

	/**
	 * @return the name
	 */
```

```
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * @return the birthDay
	 */
	public LocalDate getBirthDay() {
		return birthDay;
	}

	/**
	 * @param birthDay the birthDay to set
	 */
	public void setBirthDay(LocalDate birthDay) {
		this.birthDay = birthDay;
	}

	/**
	 * @return the score
	 */
	public int getScore() {
		return score;
	}

	/**
	 * @param score the score to set
	 */
	public void setScore(int score) {
		this.score = score;
	}

	@Override
	public String toString() {
		return "StudentEx2 [id=" + id + ", name=" + name + ", birthDay=" +
birthDay + ", score=" + score + "]";
	}

//	@Override
//	public int compareTo(StudentEx2 o) {
//		return this.name.compareTo(o.getName());
//	}

//	@Override
//	public int compareTo(StudentEx2 o) {
//		int flagCompare = this.name.compareTo(o.getName());
//		if (flagCompare == 0) {
//			if (this.score > o.score) {
//				return 1;
//			}
//			else if (this.score < o.score) {
//				return -1;
```

```
//                  }else {
//                        return 0;
//                  }
//
//            } else {
//                  return flagCompare;
//            }
//
//
//      }

      @Override
      public int compareTo(StudentEx2 o) {
            int flagCompare = this.name.compareTo(o.getName());
            if (flagCompare == 0) {
                  if (this.score > o.score) {
                        return 1;
                  } else if (this.score < o.score) {
                        return -1;
                  } else if (this.score == o.score) {
                        return this.birthDay.compareTo(o.getBirthDay());
                  }
            } else {
                  return flagCompare;
            }
            return 0;

      }
//    @Override
//    public int compareTo(StudentEx2 o) {
//
//          return this.birthDay.compareTo(o.getBirthDay());
//
//    }
}
```

➔Hàm so sánh theo name

```
      @Override
      public int compareTo(StudentEx2 o) {
            return this.name.compareTo(o.getName());
      }
```

➔Hàm so sánh theo name + score

```
      @Override
      public int compareTo(StudentEx2 o) {
            int flagCompare = this.name.compareTo(o.getName());
            if (flagCompare == 0) {
                  if (this.score > o.score) {
                        return 1;
                  }
                  else if (this.score < o.score) {
                        return -1;
                  }else {
                        return 0;
                  }

            } else {
                  return flagCompare;
```

```
            }


        }
```

➔Hàm so sánh theo name + score + localdate

```
@Override
        public int compareTo(StudentEx2 o) {
                int flagCompare = this.name.compareTo(o.getName());
                if (flagCompare == 0) {
                        if (this.score > o.score) {
                                return 1;
                        } else if (this.score < o.score) {
                                return -1;
                        } else if (this.score == o.score) {
                                return this.birthDay.compareTo(o.getBirthDay());
                        }
                } else {
                        return flagCompare;
                }
                return 0;

        }
```

➔Hàm so sánh theo LocalDate

```
        @Override
        public int compareTo(StudentEx2 o) {

                return this.birthDay.compareTo(o.getBirthDay());

        }
```

➔Tạo class Exercise2 để demo chương trình chạy

```
package com.vti.backend;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;

import com.vti.entity.StudentEx2;
import com.vti.ultis.ScannerUltis;

public class Exercise2 {

        public void question1() {
                List<StudentEx2> listStudentEx2 = new ArrayList<StudentEx2>();

                while (true) {
                        loadMenuEx2Que1();
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                System.out.println("Nhập vào số sinh viên: ");
                                int num2 = ScannerUltis.inputIntPositive();
                                for (int i = 0; i < num2; i++) {
                                        System.out.println("Nhập sinh viên " + (1) + ":
");
                                        System.out.println("Nhập vào tên: ");
```

```
                                    String nameStudent = ScannerUltis.inputString();
                                    System.out.println("Nhập vào năm sinh");
                                    LocalDate birthStudent =
ScannerUltis.inputLocalDate();
                                    System.out.println("Nhập vào điểm: ");
                                    int scoreStudent =
ScannerUltis.inputIntPositive();
                                    StudentEx2 stex2 = new StudentEx2(nameStudent,
birthStudent, scoreStudent);
                                    listStudentEx2.add(stex2);
                              }
                              System.out.println("Danh sách sinh viên vừa nhập: ");
                              for (StudentEx2 studentEx2 : listStudentEx2) {
                                    System.out.println(studentEx2);
                              }
                              break;
                        case 2:
                              Collections.sort(listStudentEx2);
                              System.out.println("Danh sách sinh viên sau khi sắp xếp
theo tên: ");
                              for (StudentEx2 studentEx2 : listStudentEx2) {
                                    System.out.println(studentEx2);
                              }
                              break;
                        case 3:
//                            Collections.sort(listStudentEx2);
//                            System.out.println("Danh sách sinh viên sau khi sắp xếp
theo tên, điểm, ngày sinh: ");
//                            for (StudentEx2 studentEx2 : listStudentEx2) {
//                                  System.out.println(studentEx2);
//                            }
//                            Giống với case 2: nhưng điều chỉnh lại hàm CompareTo
trong Class StudentEx2
                              break;
                        case 4:
                              return;
                        default:
                              System.out.println("Nhập lại: ");
                              break;
                        }
                  }
            }

            private void loadMenuEx2Que1() {
                  System.out.println("---              Question 6 7:  ");
                  System.out.println("---              1. Tạo danh sách sinh viên ");
                  System.out.println("---              2. Sắp xếp danh sách theo name
");
                  System.out.println("---              3. Sắp xếp theo name, ngày sinh,
điểm  ");
                  System.out.println("---              4.Exit  ");

            }

}
```

Exercise 3: Generic

Question 1: T generic (class)

Tạo class student có property id, name (trong đó id của student có thể là int, long, float)

a) Tạo đối tượng student có id là int

b) Tạo đối tượng student có id là float

c) Tạo đối tượng student có id là double

Tạo Class Student kiểu **T**

```
package com.vti.entity.exercise3;

public class Student<T> {
	private T id;
	private String name;

	public Student(T id, String name) {
		this.id = id;
		this.name = name;
	}

	public T getId() {
		return id;
	}

	public void setId(T id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	@Override
	public String toString() {
		return "Student{" + "id=" + id + ", name='" + name + '\'' + '}';
	}
}
```

➔Tạo class để khởi tạo các đối tượng theo yêu cầu bài toán

```
package com.vti.backend;
import com.vti.entity.exercise3.Employee;
import com.vti.entity.exercise3.Phone;
import com.vti.entity.exercise3.Staff;
import com.vti.entity.exercise3.Student;
public class Generic {

	public void question1_2_3() {
		// khởi tạo student có id là int
		Student<Integer> student1 = new Student<Integer>(1, "Dang Black");

		// khởi tạo student có id là float
		Student<Float> student2 = new Student<Float>(2.0f, "Hai Dang Black");

		// khởi tạo student có id là double
		Student<Double> student3 = new Student<Double>(3.0, "Duy Nguyen VTI");

		// Question 2: T generic method print object
		print(student1);
		print(student2);
		print(student3);

		// print number
		print(1);
		print(2f);
```

```
                print(3d);
        }

        public void question4() {
                // init array
                Integer[] arrInt = { 5, 10, 15 };
                Float[] arrFloat = { 6f, 5f, 15f };
                Double[] arrDouble = { 5.2, 2.6, 6.9 };

                // print array
                printArray(arrInt);
                printArray(arrFloat);
                printArray(arrDouble);
        }

        public void question5() {
                // khởi tạo employee có salaries datatype là int
                Integer[] salaryEmployee1 = { 1000, 1200, 1200 };
                Employee<Integer> employee1 = new Employee<Integer>(1, "Đăng", salaryEmployee1);

                System.out.println("Employee vừa khởi tạo: ");
                print(employee1);
                System.out.println("Tháng lương cuối cùng của employee 1: " +
salaryEmployee1[salaryEmployee1.length - 1]);

                // khởi tạo employee có salaries datatype là Float
                Float[] salaryEmployee2 = { 1000f, 1200f, 1200f, 3000f };
                Employee<Float> employee2 = new Employee<Float>(1, "Duy", salaryEmployee2);

                System.out.println("Employee vừa khởi tạo: ");
                print(employee2);
                System.out.println("Tháng lương cuối cùng của employee 2: " +
salaryEmployee2[salaryEmployee2.length - 1]);

                // khởi tạo employee có salaries datatype là double
                Double[] salaryEmployee3 = { 1000d, 1200d, 1200d, 6900d, 9600d, 12000d };
                Employee<Double> employee3 = new Employee<Double>(1, "Thắng", salaryEmployee3);

                System.out.println("Employee vừa khởi tạo: ");
                print(employee3);
                System.out.println("Tháng lương cuối cùng của employee1: " +
salaryEmployee3[salaryEmployee3.length - 1]);
        }

        public void question7() {
                // <email, phone number>
                Phone<String, String> email = new Phone<String, String>("duynn03@gmail.com",
"0332782799");
                System.out.println("Email: " + email.getKey() + " / " + "Phone Number: " +
email.getPhoneNumber());

                // <id, phone number>
                Phone<Integer, String> id = new Phone<Integer, String>(1, "0332782799");
                System.out.println("id: " + id.getKey() + " / " + "Phone Number: " +
id.getPhoneNumber());

                // <name, phone number>
                Phone<String, String> name = new Phone<String, String>("Duy", "0332782799");
                System.out.println("Name: " + name.getKey() + " / " + "Phone Number: " +
name.getPhoneNumber());
        }

        public void question8() {
                // Integer
                Staff<Integer> staff1 = new Staff<Integer>(1, "Nguyễn Văn A");
                System.out.println("ID: " + staff1.getId() + " / " + "Name: " + staff1.getName());

                // Float
                Staff<Float> staff2 = new Staff<Float>(1.5f, "Nguyễn Văn A");
                System.out.println("ID: " + staff2.getId() + " / " + "Name: " + staff2.getName());
        }

        private <T> void print(T a) {
                System.out.println(a);
```

```
	}

	private <T> void printArray(T[] arr) {
		for (T x : arr) {
			System.out.println(x);
		}
	}

}
```

Question 1: Tạo class student có property id, name (trong đó id của student có thể là int, long, float)

a) Tạo đối tượng student có id là int

b) Tạo đối tượng student có id là float

c) Tạo đối tượng student có id là double

➔Tạo class StudentEx3<T>

```
package com.vti.entity;

public class StudentEx3<T> {
	private T id;
	private String name;

	public StudentEx3(T id, String name) {
		super();
		this.id = id;
		this.name = name;
	}

	@Override
	public String toString() {
		return "StudentEx3 [id=" + id + ", name=" + name + "]";
	}


}
```

```
case 1:
				System.out.println("Tạo các đối tượng Student: ");
				StudentEx3<Integer> studentInt = new
StudentEx3<Integer>(1, "studentInt");
				StudentEx3<Float> studentFloat = new
StudentEx3<Float>(2.0f, "studentFloat");
				StudentEx3<Double> studentDouble = new
StudentEx3<Double>(3.0, "StudentDouble");
				System.out.println("Danh sách sinh viên đã được
tạo");
				System.out.println(studentInt);
				System.out.println(studentFloat);
				System.out.println(studentDouble);
				break;
```

Question 2: T generic (method)

Tạo method để in ra thông tin nhập vào (parameter)
(parameter có thể là họ và tên, hoặc student, hoặc int)
Gợi ý: tạo method print(T a) và cài đặt system out (a) ra
Demo chương trình với print(student), print(4), print(4.0)

```
private <T> void printByT_Type(T var) {
		System.out.println("In bởi T_Generic: " + var);
	}
```

➔gọi phương thức:

```
	case 2:
				System.out.println("Nhập vào 1 số nguyên: ");
				printByT_Type(ScannerUltis.inputInt());
				System.out.println("Nhập vào 1 số Float: ");
				printByT_Type(ScannerUltis.inputFloat());
				System.out.println("Nhập vào 1 số Double: ");
				printByT_Type(ScannerUltis.inputDouble());
				System.out.println("In đối tượng: ");
				StudentEx3<Integer> studentT = new
StudentEx3<Integer>(10, "StudentTGenneric");
				printByT_Type(studentT);
				break;
```

Question 4: E generic

Tạo 1 array int, 1 array float, 1 array long, 1 array double
Tạo 1 method có parameter là array và in ra các số trong array đó
➔Viết phương thức Print Array:

```
	private <E> void printByE_Type(E[] arr) {
		System.out.println("Phần tử trong mảng là: ");
		for (int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + " ");
		}
		System.out.println(" ");
	}
```

➔ viết logic:

```
		case 3:
				Integer[] intArray = { 1, 5, 2, 7, 8 };
				Float[] floatArray = { 1.2f, 5.3f, 2f, 7.8f, 8.2f };
				Double[] doubleArray = { 1.2, 5.3, 2.2, 7.8, 8.2 };
				printByE_Type(intArray);
				printByE_Type(floatArray);
				printByE_Type(doubleArray);
				break;
```

Question 5: E generic

Tạo 1 class Employee có property id, name, salaries với salaries là lương các tháng của Employee đó và là 1 array có data type có thể là int, long, float.
Viết method trong Employee để in ra thông tin của Employee bao gồm id, name, salaris. ➔ Viết trong hàm ToString phía dưới

Viết method trong Employee để in ra thông tin tháng lương cuối cùng của Employee

Tạo Class `Employee` với salaries kiểu T

```
package com.vti.entity;

import java.util.Arrays;

public class Employee<T> {
	public static int COUNT = 0;
	private int id;
	private String name;
	private T[] salaries;

	public Employee(String name, T[] salaries) {
		super();
		this.id = ++COUNT;
		this.name = name;
		this.salaries = salaries;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", salaries=" +
Arrays.toString(salaries) + "]";
	}

	public String getLastSalary() {
		return "Employee [id=" + id + ", name=" + name + ", salaries=" +
salaries[salaries.length-1] + "]";
	}

}
```

a) Hãy tạo chương trình demo với Employee có salaries là datatype int

b) Hãy tạo chương trình demo với Employee có salaries là datatype float

c) Hãy tạo chương trình demo với Employee có salaries là datatype double

Hàm `getLastSalary` viết trong class Employee để lấy tháng lương cuối cùng.

```
public String getLastSalary() {
		return "Employee [id=" + id + ", name=" + name + ", salaries=" +
salaries[salaries.length-1] + "]";
	}
```

➔Tạo Class `Exercise3<E>`

```
		package com.vti.backend;

import java.util.ArrayList;
import java.util.List;
```

```
import com.vti.entity.Employee;
import com.vti.entity.StudentEx3;
import com.vti.ultis.ScannerUltis;

public class Exercise3<E> {

	public void question1() {
		while (true) {
			loadMenuEx3();
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				System.out.println("Tạo các đối tượng Student: ");
				StudentEx3<Integer> studentInt = new
StudentEx3<Integer>(1, "studentInt");
				StudentEx3<Float> studentFloat = new
StudentEx3<Float>(2.0f, "studentFloat");
				StudentEx3<Double> studentDouble = new
StudentEx3<Double>(3.0, "StudentDouble");
				System.out.println("Danh sách sinh viên đã được
tạo");
				System.out.println(studentInt);
				System.out.println(studentFloat);
				System.out.println(studentDouble);
				break;

			case 2:
				System.out.println("Nhập vào 1 số nguyên: ");
				printByT_Type(ScannerUltis.inputInt());
				System.out.println("Nhập vào 1 số Float: ");
				printByT_Type(ScannerUltis.inputFloat());
				System.out.println("Nhập vào 1 số Double: ");
				printByT_Type(ScannerUltis.inputDouble());
				System.out.println("In đối tượng: ");
				StudentEx3<Integer> studentT = new
StudentEx3<Integer>(10, "StudentTGenneric");
				printByT_Type(studentT);
				break;
			case 3:
				Integer[] intArray = { 1, 5, 2, 7, 8 };
				Float[] floatArray = { 1.2f, 5.3f, 2f, 7.8f, 8.2f };
				Double[] doubleArray = { 1.2, 5.3, 2.2, 7.8, 8.2 };
				printByE_Type(intArray);
				printByE_Type(floatArray);
				printByE_Type(doubleArray);
				break;
			case 4:
				System.out.println("Tạo Employee 1 -->OK");
				Integer[] salInt = { 1000, 2000, 3000 };
				Employee<Integer> emInt = new
Employee<Integer>("EmployeeInt", salInt);
				System.out.println("Tạo Employee 2 -->OK");
				Float[] salFloat = { 1000f, 2000f, 3000f };
				Employee<Float> emFloat = new
Employee<Float>("EmployeeFloat", salFloat);
				System.out.println("Tạo Employee 3 -->OK");
				Double[] salDouble = { 1000.1, 2000.2, 3000.3 };
				Employee<Double> emDouble = new
Employee<Double>("EmployeeInt", salDouble);
```

```
                        System.out.println("Thông tin các Employee vừa tạo
là:");
                        System.out.println(emInt);
                        System.out.println(emFloat);
                        System.out.println(emDouble);
                        System.out.println("Thông tin tháng lương cuối cùng:
");
                        System.out.println(emInt.getLastSalary());
                        System.out.println(emFloat.getLastSalary());
                        System.out.println(emDouble.getLastSalary());
                        break;
                  case 5:
                        return;
                  default:
                        break;
                  }
            }
      }

      private void loadMenuEx3() {
            System.out.println("---            Exercise 3.  ");
            System.out.println("---            1. Question1 ");
            System.out.println("---            2. Question2   ");
            System.out.println("---            3. Question4 ");
            System.out.println("---            4. Question5 ");
            System.out.println("---            5.Exit  ");

      }

      private <T> void printByT_Type(T var) {
            System.out.println("In bởi T_Generic: " + var);
      }

      private <E> void printByE_Type(E[] arr) {
            System.out.println("Phần tử trong mảng là: ");
            for (int i = 0; i < arr.length; i++) {
                  System.out.print(arr[i] + " ");
            }
            System.out.println(" ");
      }

}
```


Question 6: K & V generic

Tạo 1 class có tên là MyMap, lưu dữ liệu theo dạng key, value

Tạo các method

a) GetValue()

b) getKey ()

Viết chương trình demo: tạo 1 object Student có key là Mã sinh viên và value là tên của sinh viên đó

Tạo Class MyMap:

```
package com.vti.entity;

public class MyMap<K, V> {
      private K key;
```

```
	private V value;
	/**
	 * @return the key
	 */
	public K getKey() {
		return key;
	}
	/**
	 * @param key the key to set
	 */
	public void setKey(K key) {
		this.key = key;
	}
	/**
	 * @return the value
	 */
	public V getValue() {
		return value;
	}
	/**
	 * @param value the value to set
	 */
	public void setValue(V value) {
		this.value = value;
	}
	public MyMap(K key, V value) {
		super();
		this.key = key;
		this.value = value;
	}
	@Override
	public String toString() {
		return "MyMap [key=" + key + ", value=" + value + "]";
	}

}
```



```
public void question6() {
		MyMap<Integer, String> Student = new MyMap<Integer, String>(1,
"Student");
		System.out.println("Thông tin sinh viên khởi tạo: " + Student);
		while (true) {
			loadMenuquestion6();
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				System.out.println("Lấy Value trong MyMap: " +
Student.getValue());
				break;

			case 2:
				System.out.println("Lấy Key trong MyMap: " +
Student.getKey());
				break;

			case 3:
				return;
```

```
                    default:
                        System.out.println("Chọn lại:");
                        break;
                    }
                }
            }
```

Question 7: K & V generic

Tạo 1 class có tên là Phone, lưu dữ liệu theo dạng key, value (extends MyMap)
Với key là email hoặc là số thứ tự hoặc là tên người sử dụng
Với value là số điện thoại
➔Tạo Class Phone:

```
package com.vti.entity;

public class Phone<K, V> extends MyMap<K, V> {

	public Phone(K key, V value) {
		super(key, value);

	}

	public <T> T GetPhoneNumber() {
		return (T) super.getValue();
	}

	@Override
	public K getKey() {

		return super.getKey();
	}
}
```

➔Tạo Class Phone:

```
package com.vti.entity.exercise3;

public class Phone<K, V> extends MyMap<K, V> {

	public Phone(K key, V value) {
		super(key, value);
	}

	public V getPhoneNumber() {
		return super.getValue();
	}
}
```

Tạo các method
a) GetPhoneNumber()
b) getKey ()
Viết chương trình demo với
a) key là email

b) key là số thứ tự
c) key là tên của người sử dụng

```
public void question7() {
		while (true) {
			loadMenuquestion7();
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				System.out.println("Chương trình với key là Email:
");
				Phone<String, String> phone1 = new Phone<String,
String>("Email1", "090-001");
				Phone<String, String> phone2 = new Phone<String,
String>("Email2", "090-002");
				Phone<String, String> phone3 = new Phone<String,
String>("Email3", "090-003");
				Phone<String, String> phone4 = new Phone<String,
String>("Email4", "090-004");
				Phone<String, String> phone5 = new Phone<String,
String>("Email5", "090-005");
				System.out.println("Thông tin Phone trong hệ thống");
				System.out.println("Key: " + phone1.getKey() + "
Value: " + phone1.GetPhoneNumber());
				System.out.println("Key: " + phone2.getKey() + "
Value: " + phone2.GetPhoneNumber());
				System.out.println("Key: " + phone3.getKey() + "
Value: " + phone3.GetPhoneNumber());
				System.out.println("Key: " + phone4.getKey() + "
Value: " + phone4.GetPhoneNumber());
				System.out.println("Key: " + phone5.getKey() + "
Value: " + phone5.GetPhoneNumber());

				break;

			case 2:
				System.out.println("Chương trình với key là số thứ
tự: ");
				Phone<Integer, String> phone_num1 = new
Phone<Integer, String>(1, "090-001");
				Phone<Integer, String> phone_num2 = new
Phone<Integer, String>(2, "090-002");
				Phone<Integer, String> phone_num3 = new
Phone<Integer, String>(3, "090-003");
				Phone<Integer, String> phone_num4 = new
Phone<Integer, String>(4, "090-004");
				Phone<Integer, String> phone_num5 = new
Phone<Integer, String>(5, "090-005");
				System.out.println("Thông tin Phone trong hệ thống");
				System.out.println("Key: " + phone_num1.getKey() + "
Value: " + phone_num1.GetPhoneNumber());
				System.out.println("Key: " + phone_num2.getKey() + "
Value: " + phone_num2.GetPhoneNumber());
				System.out.println("Key: " + phone_num3.getKey() + "
Value: " + phone_num3.GetPhoneNumber());
				System.out.println("Key: " + phone_num4.getKey() + "
Value: " + phone_num4.GetPhoneNumber());
				System.out.println("Key: " + phone_num5.getKey() + "
Value: " + phone_num5.GetPhoneNumber());
				break;
```

```
                    case 3:
                          System.out.println("Chương trình với key là Email:
");
                          Phone<String, String> phone_name1 = new Phone<String,
String>("User_Name_1", "090-001");
                          Phone<String, String> phone_name2 = new Phone<String,
String>("User_Name_2", "090-002");
                          Phone<String, String> phone_name3 = new Phone<String,
String>("User_Name_3", "090-003");
                          Phone<String, String> phone_name4 = new Phone<String,
String>("User_Name_4", "090-004");
                          Phone<String, String> phone_name5 = new Phone<String,
String>("User_Name_5", "090-005");
                          System.out.println("Thông tin Phone trong hệ thống");
                          System.out.println("Key: " + phone_name1.getKey() + "
Value: " + phone_name1.GetPhoneNumber());
                          System.out.println("Key: " + phone_name2.getKey() + "
Value: " + phone_name2.GetPhoneNumber());
                          System.out.println("Key: " + phone_name3.getKey() + "
Value: " + phone_name3.GetPhoneNumber());
                          System.out.println("Key: " + phone_name4.getKey() + "
Value: " + phone_name4.GetPhoneNumber());
                          System.out.println("Key: " + phone_name5.getKey() + "
Value: " + phone_name5.GetPhoneNumber());

                          break;
                    case 4:
                          return;

                    default:
                          System.out.println("Chọn lại: ");
                          break;
                    }
              }
        }
```

➔Đoạn này lời giải chưa rõ ràng, cần xem lại

Question 8: K & V generic

Tạo 1 class có tên là Staff, lưu dữ liệu theo dạng key, value (extends MyMap)
Với key là id của Staff (ID có thể là int, long)
Với value là tên của Staff
Tạo các method

➔Tạo Class `Staff` extends MyMap

```
package com.vti.entity;

public class Staff<K> extends MyMap<K, String> {

      public Staff(K key, String value) {
            super(key, value);
      }

      public K getID() {
            return super.getKey();
```

```
	}

	public String getName() {
		return super.getValue();

	}
}
```

a) GetId ()
b) getName ()
Viết chương trình demo:

```
public void question8() {
		Staff<Integer> staffInt = new Staff<Integer>(1, "StaffInt");
		Staff<Long> staffLong = new Staff<Long>(2L, "staffLong");
		while (true) {
			loadMenuquestion8();
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				System.out.println("Thông tin ID: ");
				System.out.println("Nhân viên staffInt: " +
staffInt.getID());
				System.out.println("Nhân viên staffLong: " +
staffLong.getID());
				break;
			case 2:
				System.out.println("Thông tin Value: ");
				System.out.println("Nhân viên staffInt: " +
staffInt.getValue());
				System.out.println("Nhân viên staffLong: " +
staffLong.getValue());
				break;
			case 3:

				return;
			default:
				break;
			}
		}

	}
```


Exercise 4 (Optional): Wildcard (Generic) ➔ Nội dung này chưa viết lại lời giải

Question 1:

Tạo 1 class Salary để đại diện cho datatype là các số
a) Hãy config class Salary như sau: Salary <N> với N phải được
extends từ Number.class
b) Tạo method để print ra salary hiện tại
➔Tạo Class `Salary`

```
package com.vti.entity.exercise4;
```

```
public class Salary<N extends Number> {
	private N salary;

	public Salary(N salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		return "Salary{" + "salary=" + salary + '}';
	}
}
```

➔Tạo phương thức Question 1:

```
public void question1() {
		// Integer
		Salary<Integer> salary = new Salary<Integer>(69000);
		System.out.println(salary);
	}
```

Question 2:

Tạo 1 class MyNumber để đại diện cho datatype là các số
Hãy config class MyNumber như sau: MyNumber<N> với N phải được extends từ Number.class

➔Tạo Class `MyNumber`

```
package com.vti.entity.exercise4;

import java.util.Comparator;

public class MyNumber<N extends Number> implements Comparator<N> {

	private N number;

	public MyNumber(N number) {
		this.number = number;
	}

	@Override
	public String toString() {
		return "MyNumber{" + "number=" + number + '}';
	}

	@SuppressWarnings("unchecked")
	@Override
	public int compare(N a, N b) {
		if (!(a instanceof Comparable)) {
			throw new UnsupportedOperationException();
		}

		return ((Comparable<N>) a).compareTo(b);
	}

}
```

➔Tạo phương thức Ques2

```
public void question2() {
		// Double
		MyNumber<Double> number = new MyNumber<Double>(6900.900);
		System.out.println(number);
	}
```

Question 3: T generic (method)

Tạo method tính max của 2 số (số nhập vào có thể là float, double, int, long).

Demo chương trình

➔Tạo Class MyMath

```
package com.vti.entity.exercise4;

public class MyMath<N extends Number> {

	@SuppressWarnings({ "unchecked" })
	public N add(N... numbers) {

		Double sum = 0d;

		for (N number : numbers) {
			sum += number.doubleValue();
		}

		return (N) sum;
	}

	public <T extends Comparable<T>> T maximum(T x, T y, T z) {
		// assume x is initially the largest
		T max = x;

		// y is the largest so far
		if (y.compareTo(max) > 0) {
			max = y;
		}

		if (z.compareTo(max) > 0) {
			max = z; // z is the largest
		}

		return max;
	}

	@SuppressWarnings("unchecked")
	public N subtract(N x, N y) {
		Double subtract = x.doubleValue() - y.doubleValue();

		return (N) subtract;
	}

}
```

➔Tạo phương thức Ques 4

```
public void question3() {

		MyMath<Integer> math = new MyMath<>();

		// Integer
		int maxInt = math.maximum(1, 10, 8);
		System.out.println(maxInt);

		// float
		float maxFloat = math.maximum(1.5f, 10.3f, 10.2f);
		System.out.println(maxFloat);
	}
```

Question 4: T generic (method & class)

Tạo class MyMath<T> có thể làm việc được với các số int, long, double và có chứa các method sau:

a) tính tổng của 2 số
b) tính tổng của 3 số
c) tính tổng của 4 số
d) tính hiệu của 2 số
e) hàm tính số mũ (a,b)

Gợi ý: sử dụng optional parameter để làm câu a,b,c.

Demo chương trình

➔Tạo Class MyMath

```
package com.vti.entity.exercise4;

public class MyMath<N extends Number> {

	@SuppressWarnings({ "unchecked" })
	public N add(N... numbers) {

		Double sum = 0d;

		for (N number : numbers) {
			sum += number.doubleValue();
		}

		return (N) sum;
	}

	public <T extends Comparable<T>> T maximum(T x, T y, T z) {
		// assume x is initially the largest
		T max = x;

		// y is the largest so far
		if (y.compareTo(max) > 0) {
			max = y;
		}

		if (z.compareTo(max) > 0) {
			max = z; // z is the largest
		}

		return max;
	}

	@SuppressWarnings("unchecked")
	public N subtract(N x, N y) {
		Double subtract = x.doubleValue() - y.doubleValue();

		return (N) subtract;
	}

}
```

➔Tạo Ques 5 xử lý logic

```
	public void question4() {

		MyMath<Integer> mathInt = new MyMath<>();
		MyMath<Float> mathFloat = new MyMath<>();

		// add
		Integer sumInt1 = mathInt.add(1, 10);
		System.out.println(sumInt1);

		int sumInt2 = mathInt.add(1, 10, 8);
```

```
                System.out.println(sumInt2);

                Float sumFloat = mathFloat.add(1.5f, 10.3f, 10.2f);
                System.out.println(sumFloat);

                // subtract
                int subInt1 = mathInt.subtract(10, 1);
                System.out.println(subInt1);

                float subFloat2 = mathFloat.subtract(10f, 5f);
                System.out.println(subFloat2);
        }
```

Question 5:

Tạo 1 cursor như sau: ArrayList <Object> listWildcards.
Hãy khởi tạo đối tượng có chứa thông tin của khách hàng (mỗi thông tin là 1 element) và thêm các giá trị vào trong ArrayList như:
a) Nguyễn Văn Nam
b) 30 tuổi
c) Hà đông, Hà nội

```
public void question5() {
                List<Object> inforCustomers = new ArrayList<>();
                inforCustomers.add("Nguyễn Văn Nam");
                inforCustomers.add(30);
                inforCustomers.add("Hà đông, Hà nội");

                for (Object object : inforCustomers) {
                        System.out.println(object);
                }
        }
```