BÀI THỰC HÀNH SỐ 3

Xử lý ngoại lệ - Generics - Collections

I. Mục tiêu.

Hiểu và nắm vững các cách thức để xử lý ngoại lệ (Exceptions) trong Java, các cách sử dụng Java Generics và Colllections.

II. Thời gian

- Hướng dẫn chung và thực hành tại lớp: 5 tiết
- Bài tập về nhà: 10 tiết

III. Hướng dẫn chung

Phần 1: Xử lý ngoại lệ trong Java.

Câu 1: Viết chương trình nhập vào một mảng cố định 5 phần tử các số nguyên dương từ bàn phím. Yêu cầu sử dụng **khối lệnh try...catch** để xử lý ngoại lệ có thể xảy ra với mảng này như là: nhập quá số lượng phần tử của mảng và nhập vào một số không phải số nguyên dương.

Hướng dẫn: sử dụng các lớp xử lý ngoại lệ ArrayIndexOutOfBoundsException và Exception để bắt các ngoại lệ xảy ra. Code ví dụ như sau:

```
// Tạo một đối tượng Scanner
Scanner scanIn=new Scanner(System.in);
// Tạo mảng số nguyên
int[] soNguyen = new int[5];
int so;
int i =0;
try{
    while(true){
        System.out.print("Nhập số nguyên: ");
        so = scanIn.nextInt();
        soNguyen[i] = so;
        i++;
    }
}catch(ArrayIndexOutOfBoundsException aie){
    System.out.println("Bạn đã nhập quá số lượng phần tử");
}
catch (Exception e){
    System.out.println("Vui lòng nhập số nguyên");
}
//In ra mảng vừa nhập
System.out.println("Mảng vừa nhập");
for (int j = 0; j < soNguyen.length; j++) {
    System.out.println(soNguyen[j]);
}
```

Câu 2: Viết chương trình tạo ra 10 số nguyên ngẫu nhiên và lưu vào một mảng 10 phần tử; người dùng nhập vào một chỉ số của mảng từ bàn phím và xuất ra giá trị của vị trí đó trong mảng. Yêu cầu sử dụng **throw và throws** để xử lý ngoại lệ có thể xảy ra với mảng này khi nhập chỉ số vượt chỉ số của mảng và khi nhập chỉ số mảng nhỏ hơn 0.

Hướng dẫn:

- Sử dụng các lớp xử lý ngoại lệ ArrayIndexOutOfBoundsException và ArithmeticException để bắt các ngoại lệ xảy ra. Code ví dụ như sau:

```
public static void exceptionThrows() throws ArrayIndexOutOfBoundsException, Exception{
    //Tạo mảng 10 phần tử các số nguyên ngẫu nhiên
    int randomNums[] = new int[10];
    Random rand = new Random();
    for (int i = 0; i < randomNums.length; i++) {
        randomNums[i] = rand.nextInt(100);
    }

    //Nhập vào chỉ số màng
    Scanner scanIn=new Scanner(System.in);
    System.out.print("Nhập vào vị trí phần tử muốn in giá trị: ");
    int index = scanIn.nextInt();
    if (index <0)
        throw new Exception();
    System.out.println("OK, phần tử mảng thứ " + index + " có giá trị " + randomNums[index]);
}
```

- Xử lý ngoại lệ ở hàm main

```
public static void main(String[] args) {
    // TODO code application logic here
    while (true){
        try{
            exceptionThrows();
        }
        catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Bạn đã nhập quá chỉ số của mảng.");
            break;
        }
        catch (Exception e){
            System.out.println("Chỉ số index phải lớn hơn 0");
        }
    }
}
```

Phần 2: Generic và collection:

Câu 1: Viết một hàm dùng để in ra các phần tử trong mảng của nhiều kiểu dữ liệu khác nhau như: Integer, String, Double…

Hướng dẫn: sử dụng Generic để viết hàm **xuatMang.**

```
public static <T> void xuatMang(T[] arr){
    for (int i = 0; i < arr.length; i++) {
        System.out.println(arr[i]);
    }
}

public static void main(String[] args) {
    // TODO code application logic here

    Integer[] arrInt = {1,2,3,4};
    String[] arrString = {"A","B","C"};
    Double[] arrDouble = {10.0, 9.0, 8.5, 5.5};
    xuatMang(arrInt);
    xuatMang(arrString);
    xuatMang(arrDouble);
}
```

Câu 2: Sử dụng bounded wildcard trong phương thức để viết phương thức Generic cho phép tính trung bình các giá trị trong mảng.

Hướng dẫn: cú pháp `GenericType<? extends upperBoundType>`

VD: `public static double getAverage(List<? extends Number> numList)`

```
public static double getAverage(List<? extends Number> numList){
    double total = 0.0;
    for (Number num : numList)
        total += num.doubleValue();
    return total/numList.size();
}

public static void main(String[] args) {
    // TODO code application logic here

    List<Integer> integerList = new ArrayList<Integer>();
    integerList.add(3);
    integerList.add(30);
    integerList.add(60);
    System.out.println(getAverage(integerList));

    List<Double> doubleList = new ArrayList<Double>();
    doubleList.add(3.0);
    doubleList.add(50.0);
    doubleList.add(33.0);
    System.out.println(getAverage(doubleList));
}
```

Câu 3: Dùng Generic để viết lớp PersonManager để quản lý các danh sách Student và Lecturer.

Hướng dẫn:

- Tạo class Student có các thuộc tính id, name, age và các phương thức constructor, setter, getter, toString.
- Tạo class Lecturer có các thuộc tính id, name, salary và các phương thức constructor, setter, getter, toString.
- Tạo class Generic PersonManager như sau:

```
public class PersonManager <T> {

    private ArrayList<T> arr = new ArrayList<T>();

    //Thêm vào ArrayList
    public void add(T obj){
        arr.add(obj);
    }
    //Hiển thị danh sách
    public void display(){
        for (T obj : arr){
            System.out.println(obj);
        }
    }
}
```

- Ở hàm main, tạo đối tượng PersonManager<Lecturer> và PersonManager<Student> để thêm lecturer và student vào; sau đó xuất thông tin các lecturer và student ra.

```
public static void main(String[] args) {
    // TODO code application logic here

    PersonManager<Lecturer> lec = new PersonManager<Lecturer>();
    lec.add(new Lecturer(1,"Lecturer 1", 5000.0));
    lec.add(new Lecturer(1,"Lecturer 2", 1000.0));
    lec.display();

    System.out.println("/---------------------------------------/");

    PersonManager<Student> stu = new PersonManager<Student>();
    stu.add(new Student(1,"Student 1",20));
    stu.add(new Student(1,"Student 2",22));
    stu.display();
}
```

Câu 4: Khai báo 1 List có Class triển khai là LinkedList, kiểu dữ liệu là String. Sau đó thêm vào phần tử là các hãng xe cho List. Viết code thực hiện các yêu cầu sau:

- Hiển thị các phần tử trong List bằng cách sử dụng Iterator.
- Thay đổi bất kỳ phần tử trong List theo vị trí trong List (nhập từ bàn phím).
- Chuyển các phần tử trong List thành chữ hoa.
- Sắp xếp List theo thứ tự tăng dần.
- Xóa các phần tử trong List từ vị trí … đến vị trí….
- Đảo List sử dụng ListIterator.

Hướng dẫn:

- Tạo danh sách là một List có Class triển khai là LinkedList:

```
List<String> listCars = new LinkedList<String>();
listCars.add("BMW");
listCars.add("Ford");
listCars.add("Chevrolet");
listCars.add("Toyota");
listCars.add("Nissan");
listCars.add("Honda");
listCars.add("Mazda");
listCars.add("Volkswagen");
listCars.add("Volvo");
listCars.add("Mercedes");
listCars.add("Hyundai");
listCars.add("Kia");
```

- Hiển thị các phần tử trong List bằng cách dùng Iterator:

```
// In list dùng Iterator
Iterator<String> it = listCars.iterator();
System.out.println("Danh sách xe: ");
while (it.hasNext()){
    System.out.println(it.next());
}
```

- Thay đổi bất kỳ phần tử trong List theo vị trí trong List (nhập từ bàn phím).

```
System.out.println("Nhập vào phần tử mới: ");
String temp = scanner.nextLine();
System.out.println("Nhập vào vị trí của phần tử cần thay đổi: ");
int index = scanner.nextInt();
listCars.set(index, temp);
```

- Chuyển các phần tử trong List thành chữ hoa.

```
ListIterator<String> ls = listCars.listIterator();
while (ls.hasNext()){
    String str = ls.next();
    ls.set(str.toUpperCase());
}
```

- Sắp xếp List theo thứ tự tăng dần.

```
Collections.sort(listCars);
```

- Xóa các phần tử trong List từ vị trí … đến vị trí…. (Nhập từ bàn phím)

```
System.out.print("Nhập vào vị trí đầu: ");
int firstInd = scanner.nextInt();
System.out.print("Nhập vào vị trí cuối: ");
int lastInd= scanner.nextInt();
// Sử dụng phương thức clear()
listCars.subList(firstInd, lastInd).clear();
```

- Đảo List sử dụng ListIterator.

```
ListIterator lsI = listCars.listIterator(listCars.size());
System.out.println("list sau khi đảo ngược: ");
while (lsI.hasPrevious()) {
    System.out.println(lsI.previous());
}
```

Câu 5: Viết chương trình tạo hai mảng chứa danh sách các môn học. Thực hiện các phép giao, hội và hiệu trên 2 mảng trên và xuất ra kết quả.

Hướng dẫn: Dùng **TreeSet** để lưu trữ hai mảng trên.

- Tạo hai mảng:

```
TreeSet<String> tree1 = new TreeSet<>();
TreeSet<String> tree2 = new TreeSet<>();
//Tạo mảng 1
tree1.add("Java");
tree1.add("PHP");
tree1.add("C#");
tree1.add("Python");
tree1.add("HTML");
tree1.add("Spring");
//Tạo mảng 2
tree2.add("Java");
tree2.add("ASP.Net");
tree2.add("C#");
tree2.add("R");
tree2.add("CSS");
tree2.add("Lavarel");
```

- Phép giao dùng hàm **retainAll()**

```
//Phép giao
treel.retainAll(tree2);
System.out.println("Phép giao "+ treel);
//Kết quả: [C#, Java]
```

- Phép hội dùng hàm **addAll()**

```
//Phép hội
treel.addAll(tree2);
System.out.println("Phép hội "+ treel);
//Kết quả: [ASP.Net, C#, CSS, HTML, Java, Lavarel, PHP, Python, R, Spring]
```

- Phép trừ dùng hàm **removeAll()**

```
//Phép trừ
treel.removeAll(tree2);
System.out.println("Phép trừ "+ treel);
//Kết quả: [HTML, PHP, Python, Spring]
```

IV. Thực hành

Câu 1: Tạo lớp **Sinhvien** để mô tả sinh viên gồm các thuộc tính: masv (kiểu **long**), tensv (kiểu **String**), và diem (kiểu **double)**; cùng với các phương thức getter, setter, constructor,toString. Nhập dữ liệu cho một sinh viên từ bàn phím và in thông tin sinh viên vừa nhập ra màn hình. Yêu cầu dùng xử lý ngoại lệ để xây dựng các hàm kiểm tra việc nhập dữ liệu vào các thuộc tính của sinh viên, nếu sai xuất hiện thông báo. Ví dụ: nhập điểm: 9a -> Xuất hiệnthông báo "Bạn phải nhập điểm là số".

Hướng dẫn: Các ngoại lệ có thể xảy ra: nhập mã sinh viên có ký tự; nhập điểm số nhỏ hơn 0 và lớn hơn 10. Sử dụng throw hoặc throws để xử lý.

Câu 2: Viết chương trình tra cứu danh bạ điện thoại lưu trữ các thông tin sau: Số điện thoại, địa chỉ đăng ký. Sử dụng cấu trúc Collection cho phù hợp để lưu trữ thông tin danh bạ và thực hiện các công việc sau:

- Tra cứu thông tin theo số điện thoại.

- Tra cứu theo số điện thoại theo địa chỉ đăng ký. Lưu ý: một địa chỉ đăng ký có thể có nhiều số điện thoại.

Câu 3: Sử dụng collection thực hiện các yêu cầu sau (Xác định đúng collection để thực hiện)

- Công ty ABC cần lưu tên nhân viên của mình và mỗi tháng một nhân viên sẽ được chọn ngẫu nhiên để nhận một phần quà tặng; viết hàm in ra tên nhân viên được chọn.
- Công ty dự định đặt tên các sản phẩm mới của mình bằng tên nhân viên (tên sản phẩm không được trùng nhau); viết hàm cung cấp danh sách tên sản phẩm cho công ty.
- Công ty muốn dùng tên nhân viên phổ biến nhất để đặt tên cho sản phẩm mới của mình (tên phổ biến là tên giống nhau nhiều nhất); viết hàm để cung cấp tên sản phẩm này.
- Trong đợt nghỉ hè, công ty muốn tổ chức cho nhân viên đi du lịch, yêu cầu đặt ra là ưu tiên cho 20 người đăng ký trước; viết hàm để đăng ký du lịch này.
- Mỗi khách hàng mua hàng của công ty sẽ có doanh số mua hàng. Công ty muốn quản lý danh sách này theo thứ tự giảm dần doanh số; viết hàm quản lý danh sách này.

Câu 4: Một bộ bài 52 lá với các quân bài số từ 2 tới 10, và các quân bài chữ: J, Q, K, A; mỗi quân bài có các thuộc tính: cơ, rô, chuồn, bích. Viết chương trình để chia bài thành 4 nhóm khác nhau, mỗi nhóm 13 lá.

Hướng dẫn: viết các lớp sau và dùng hàm **shuffle**() của collection để chia bài.

- Lớp mô tả quân bài.
- Lớp tạo bộ bài 52 quân bài không trùng nhau, và phương thức xào bài.

Câu 5: Viết chương trình tra từ điển tiếng Anh. Sử dụng Collection TreeMap để lưu trữ các từ. Xây dựng các lớp sau:

- **Lớp Từ** để mô tả một từ tiếng Anh với các thuộc tính: từ tiếng Anh, nghĩa, loại từ và phần ghi chú và các phương thức để in thông tin của từ ra.
- **Lớp Từ điển** gồm các phương thức: Thêm một từ vào danh sách và Tra từ.