Hướng dẫn lập trình với Generic

1).Sử dụng Generic với Collection

Bước 1: Tạo Java Appication

Mở một trình soạn thảo, tạo mới một Java Application. Đặt tên và đường dẫn chưa ứng dụng.

Bước 2: Tạo method non-generic

Tạo mới một class, tiếp theo hãy định nghĩa một method không sử dụng genetic (như method bình thường). Mã nguồn của method như hình bên dưới:

```
public static void demoNoneGenerics() {
    ArrayList listFruits = new ArrayList();
    listFruits.add("Orange");
    listFruits.add("Lemon");
    //phan tu thu 3 co kieu du lieu la Integer
    listFruits.add(5);

    //lay ra phan tu thu nhat
    String value1 = (String)listFruits.get(0);

    System.out.println("value 1 = " + value1);

    //lay ra phan tu thu 3, ep kieu sai
    String value3 = (String)listFruits.get(2);

    System.out.println("value 3 = " + value3);
}
```

Sau đó thực thi ứng dụng, ta sẽ thấy lỗi được ném ra lúc thực thi như hình bên dưới, do lúc lấy ra phần tử trong tuyển tập, ta đã ép kiểu sai. Ngoài ra, việc xử lý những lỗi runtime-exception sẽ mất thời gian và công sức hơn so với những lỗi xảy ra lúc biên dịch chương trình.

```
run:
value 1 = Orange
Exception in thread "main" java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.String
        at javaapplication13.DemoGenerics.demoNoneGenerics(DemoGenerics.java:25)
        at javaapplication13.DemoGenerics.main(DemoGenerics.java:47)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bước 3: Tạo generic method

Sau đó ta định nghĩa một method có sử dụng generic, rồi khai báo một tuyển tập và chèn một số phần tử vào tuyển tập như hình bên dưới.

```
public static void demoGenerics() {
    //type parameter
    ArrayList<String> listDishes = new ArrayList<String>();

    listDishes.add("Ga rang muoi");
    listDishes.add("Ga khong loi thoat");

    //se bi loi vi Generic kiem tra kieu du lieu ngay
    //luc bien dich
    //listDishes.add(8);

    //lay ra phan tu thu 2: khong can ep kieu
    String mon1 = listDishes.get(1);

    System.out.println("Mon an = " + mon1);
}
```

Tiếp theo thực thi method vừa tạo, ta có thể thấy kết quả như hình sau:

```
run:
Mon an = Ga khong loi thoat
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ta có thể thấy là khi sử dụng Generic, ta không cần phải ép kiểu khi lấy ra các phần tử trong tuyển tập, đồng thời Generic sẽ giúp kiểm tra chặt chẽ kiểu dữ liệu của phần tử khi đưa vào tuyển tập, điều này sẽ giúp giảm nguy cơ gặp ngoại lệ ClassCastException

khi ta phải ép kiểu các phân tử khi lấy ra từ tuyển tập.

2) Tạo và sử dụng Generic class và Generic method

Bước 1: Định nghĩa generic class/generic method

```
//dinh nghia mot generic class voi type parameter
public class DemoGenericClass <T>
{
        //dinh nghia mot generic method voi type parameter
        public <T> void display(T var)
        {
                System.out.println("Gia tri cua bien la: " + var);
        }
```

Đoạn code trên định nghĩa ra một generic class với type parameter là T, T sẽ được gán một kiểu dữ liệu cụ thể mỗi khi ta khởi tạo một đối tượng của lớp với một type parameter khác nhau.
Đồng thời trong đoạn code trên, ta còn định nghĩa một generic method cho phép in ra giá trị của một biến thuộc kiểu dữ liệu T, T sẽ được gán một kiểu dữ liệu cụ thể mỗi khi ta gọi method này với từng đối tượng khác nhau của lớp.

Bước 2: Sử dụng generic class

```
Integer x = new Integer(5);
DemoGenericClass<Integer> myObj1 = new DemoGenericClass<Integer>()
myObj1.display(x);

//Day la vi du demo viec khai bao 2 doi tuong thuoc cung
//mot lop nhung co su dung cac type parameter khac nhau
String str = new String("abc");
DemoGenericClass<String> myObj2 = new DemoGenericClass<String>();
myObj2.display(str);
```

Trong đoạn code trên, ta đã khởi tạo 2 đối tượng của Generic class với các giá trị cụ thể truyền cho type parameter lần lượt là: Integer, String