Generics

Generic

Generics → Tham số hóa kiểu dữ liệu

```
ArrayList<Integer> arr = new ArrayList<Integer>();
arr.add(5);
arr.add(35);
arr.add("Java"); //error
arr.add(true);   //error
```

Generic

Một số quy ước đặt tên kiểu tham số generic

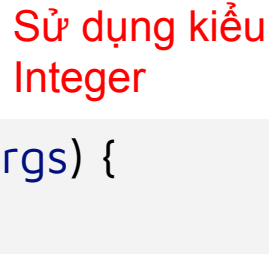
Ký tự	Ý nghĩa
E	Element (Phần tử)
K	Key (Khóa)
V	Value (Giá trị)
T	Type (Kiểu dữ liệu)
N	Number (Số)

Lớp generic

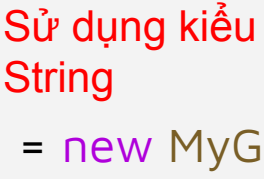
Một lớp có thể tham chiếu bất kỳ kiểu đối tượng nào được gọi là lớp generic


```
public class MyGeneric<T> {
   public T obj;
   public T getObj() {
      return obj;
   }
   public void add(T obj){
      this.obj = obj;
   }
}
```

Lớp generic




```
public static void main(String[] args) {
    //Use Integer
    MyGeneric<Integer> myGeneric1 = new MyGeneric<Integer>();
    myGeneric1.add(3);
    System.out.println(myGeneric1.getObj());

    //Use String
    MyGeneric<String> myGeneric2 = new MyGeneric<String>();
    myGeneric2.add("java");
    System.out.println(myGeneric2.getObj());
}
```

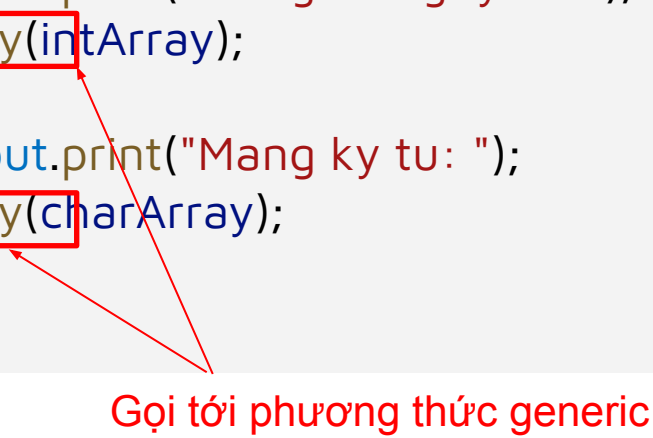


Phương thức generic

Một phương thức trong class hoặc interface đều có thể sử dụng generic

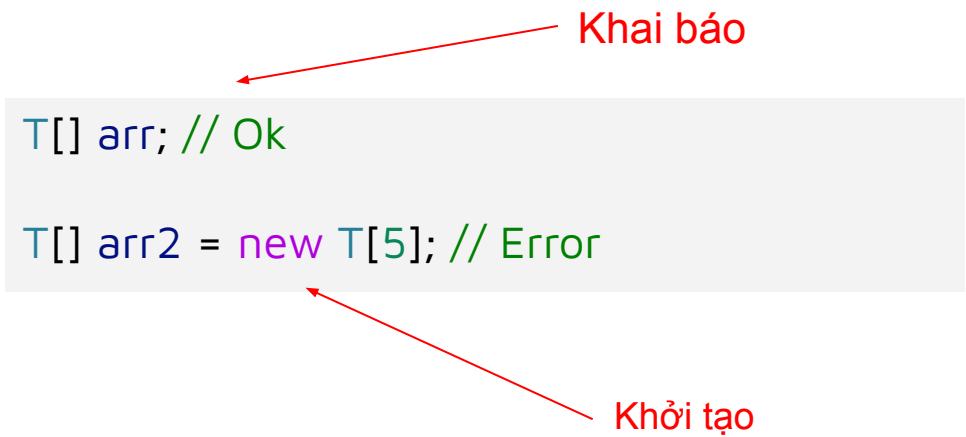
```
public static <E> void printArray(E[] elements) {
   for (E element : elements) {
     System.out.print(element + " ");
   }
   System.out.println();
 }
```

```
public static void main(String[] args) {
    Integer[] intArray = { 10, 20, 30, 40, 50 };
    Character[] charArray = { 'J', 'A', 'V', 'A' };

    System.out.print("Mang so nguyen: ");
    printArray(intArray);

    System.out.print("Mang ky tu: ");
    printArray(charArray);

  }
```



Mảng generic

Có thể khai báo một mảng generic nhưng không thể khởi tạo mảng generic vì kiểu generic không tồn tại tại thời điểm chạy. Generic chỉ có tác dụng với trình biên dịch để kiểm soát code.



Thừa kế lớp generic

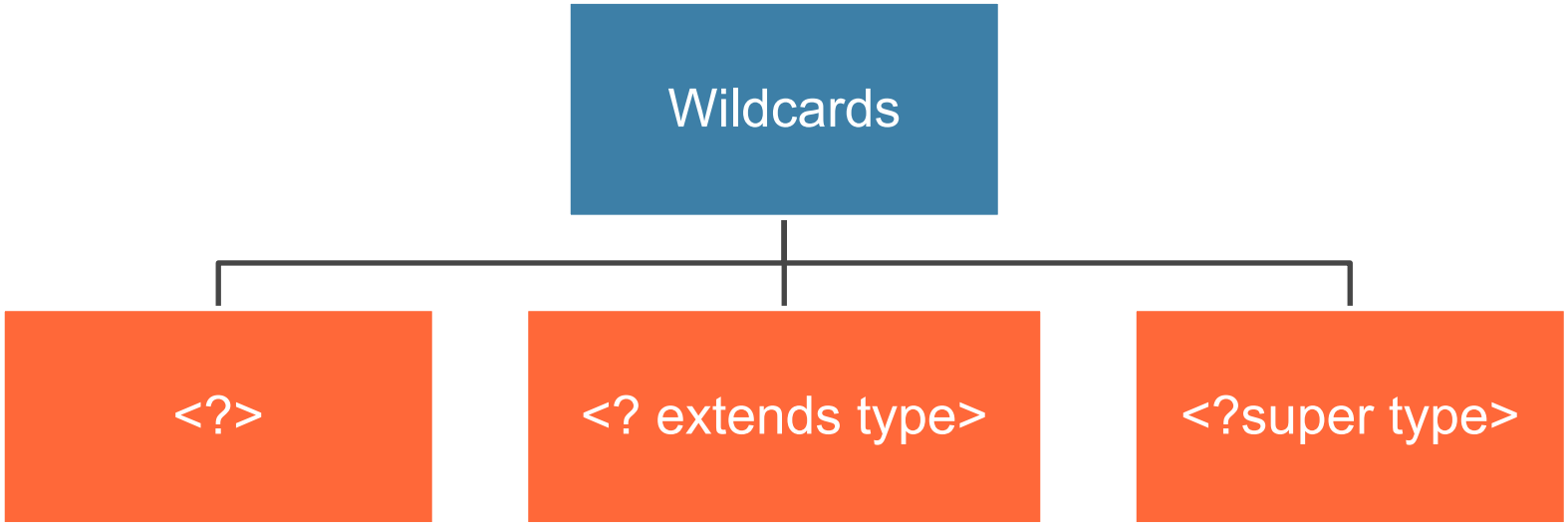
Một class mở rộng từ một class generics, nó có thể chỉ định rõ kiểu cho tham số Generics, giữ nguyên các tham số Generics hoặc thêm các tham số Generics

```
public abstract class AbstractParam<T> {
    protected T value;
    protected abstract void printValue();
}
```

```
public class Email extends AbstractParam<String>{
    @Override
    protected void printValue() {
        System.out.println("My email is:"+value);

    }
}
```

```
public static void main(String[] args) {
    Email email = new Email();
    email.value = "ngoc@techmaster.vn";
    email.printValue();// My email is:ngoc@techmaster.vn
    email.value = 10; // Lỗi
  }
```

Các ký tự đại diện generic



Ưu điểm của generics

01 Kiểu dữ liệu an toàn

02 Kiểm tra dữ liệu chặt chẽ ở Compile-time mà không phải là Runtime-error

03 Hạn chế việc ép kiểu (cast) thủ công mà không an toàn.

04 Giúp chúng ta viết các thuật toán được sử dụng nhiều, dễ dàng thay đổi, an toàn dữ liệu và dễ đọc hơn

Hạn chế của generics

- Không thể gọi Generics bằng kiểu dữ liệu nguyên thủy (Primitive type: int, long, double, …), thay vào đó sử dụng các kiểu dữ liệu Object (wrapper class thay thế: Integer, Long, Double, …).
- Không thể tạo instances của kiểu dữ liệu Generics, thay vào đó sử dụng reflection từ class.
- Không thể sử dụng static cho Generics.
- Không thể ép kiểu hoặc sử dụng instanceof.
- Không thể tạo mảng với parameterized types.
- Không thể tạo, catch, throw đối tượng của parameterized types (Generic Throwable)
- Không thể overload các hàm trong một lớp

Exercise

Tạo class có tên là MyArray và thực hiện các công việc sau:

- Thêm vào mảng một số nguyên
- Thêm vào mảng một số thực
- Thêm vào mảng một giá trị Boolean
- Thêm vào mảng một chuỗi
- In ra màn hình 4 giá trị trên

Exercise

1, Tạo class tên Student có các thuộc tính như id, name, age. Viết các phương thức constructor, setter, getter, toString.
2, Tạo class tên Employee có các thuộc tính như id, name, salary. Viết các phương thức constructor, setter, getter, toString.

Tạo class PersonModel và thực hiện các công việc sau:

```
public class PersonModel<T> {
    private ArrayList<T> al = new ArrayList<T>();

    public void add(T obj) {
        al.add(obj);
    }

    public void display() {
        for (T object : al) {
            System.out.println(object);
        }
    }
}
```

Tạo đối tượng **PersonModel<Student>**

- Gọi phương thức **add** để nhập vào 2 sinh viên
- Gọi phương thức **display** để hiển thị thông tin của 2 sinh viên vừa nhập

Tạo đối tượng **PersonModel<Employee>**

- Gọi phương thức **add** để nhập vào 2 nhân viên
- Gọi phương thức **display** để hiển thị thông tin của 2 nhân viên vừa nhập