Core Java APIs

Master Dev SS4

String - Khởi tạo và thao tác với String

String name = "Core Java Apis";

String name = new String("Core Java Apis");

- String là một class đặc biệt, khởi tạo giá trị mà không cần sử dụng từ khóa new.

Một số quy tắc cần nắm để khi thực hiện nối String

- Nếu cả 2 toán hạng là số thì + có nghĩa là phép cộng 2 số.
- Nếu một trong 2 toán hạng là chữ thì + có nghĩa là phép nối chuỗi.
- Biểu thức được thực hiện từ trái qua phải.

String - Nối chuỗi

Kết quả in ra màn hình là gì?

```
System.out.println(1 + 2);
System.out.println("a" + "b");
System.out.println("a" + "b" + 3);
System.out.println(1 + 2 + "c");
```

String - Immutability

- String là một class immutability.
- Immutability là class mà sau khi được khởi tạo đối tượng sẽ không được phép làm thay đổi giá trị trong class đó.
- Một số biện pháp được sử dụng để tạo 1 class immutability:
  - Thuộc tính cần thiết lập là private, final.
  - Chỉ có phương thức get, không có phương thức set trong class.
  - Sử dụng final ở cấp độ class để ngăn chặn class khác extend class là immutability.
  - Phương thức get với đối tượng object thì phải trả ra đối tượng mới được clone từ object muốn lấy ra trong class.

String - String Pool

- ➢ String pool được sử dụng để giải quyết vấn đề về các String trong chương trình, nhiều String lặp đi lặp lại trong CT chiếm nhiều bộ nhớ.
- ➢ String pool là một vùng nhớ đặc biệt được cấp phát bên trong JVM chưa tất cả các String literal.

```
String name = "Hello World";                 // Ê! JVM tao muốn sử dụng String Pool
String name = new String("Hello World"); // JVM tạo cho tao một đối tượng mới nhé
```

Class StringBuilder

```
String alpha = "";
for(char current = 'a'; current <= 'z'; current++) {
        alpha += current;
}
System.out.println(alpha);
```

➢ Đoạn code trên tạo ra bao nhiêu Object mới?

=> Java tạo ra StringBuilder để giải quyết vấn đề này của String. Vậy StringBuilder khác với String như nào. Cùng tìm hiểu nhé.

StringBuilder - Mutability and Chaining

➢ StringBuilder là class mutability - Có khả năng thay đổi giá trị sau khi được khởi tạo.

Nếu StringBuilder thay cho String trong ví dụ trên thì sẽ ntn:

```
StringBuilder alpha = new StringBuilder();
for(char current = 'a'; current <= 'z'; current++) {
        alpha.append(current);
}
System.out.println(alpha);
```

=> Chỉ có duy nhất 1 đối tượng mới là alpha được sinh ra và chứa toàn bộ giá trị.

Kết quả của đoạn code này là gì?

```
StringBuilder str = new StringBuilder("a");
str.append("b").append("c");
System.out.println(str.toString());
```

Arrays

Các cách khởi tạo arrays

```
int[] array1 = new int[3];
int[] array2 = new int[] {1, 2, 3};
int[] array3 = {1, 2, 3};
```

Arrays - Sắp xếp

```
int[] numbers = { 6, 9, 1 };
Arrays.sort(numbers);
```

Kết quả numbers sau khi sắp xếp:

```
String[] strings = { "10", "9", "100" };
Arrays.sort(strings);
```

Kết quả strings sau khi sắp xếp:

➢ Thứ tự ưu tiên sẽ là số trước chữ cái và uppercase sẽ trước lowercase.

Arrays - Tùy chỉnh hàm sắp xếp Arrays

Sử dụng Comparable

Comparable là 1 interface chỉ có duy nhất 1 method.

```
public interface Comparable<T> {

    public int compareTo(T o);

}
```

3 quy tắc về compareTo:

- ➢ Hai đối tượng bằng nhau thì return 0.
- ➢ Đối tượng hiện tại bé hơn thì return số < 0.
- ➢ Đối tượng hiện tại lớn hơn thì return số > 0.

Arrays - Tùy chỉnh hàm sắp xếp Arrays

Sử dụng Comparator

Nếu muốn sắp xếp các đối tượng trong danh sách mà không muốn sử dụng đến Comparable hoặc muốn sắp xếp theo nhiều cách khác nhau thì phải làm như thế nào?

➢ Comparator là 1 functional interface.

Thế nào là 1 functional interface?

Arrays - Tùy chỉnh hàm sắp xếp Arrays

Sử dụng Comparator

Ví dụ: Với Student là 1 class có 2 thuộc tính là name(String) và age(Integer)

```
Comparator<Student> comparatorWithName = new Comparator<Student>() {
    @Override
    public int compare(Student student, Student t1) {
        return student.getName().compareTo(t1.getName());
    }
};
Arrays.sort(students, comparatorWithName);
```

Generics là gì?

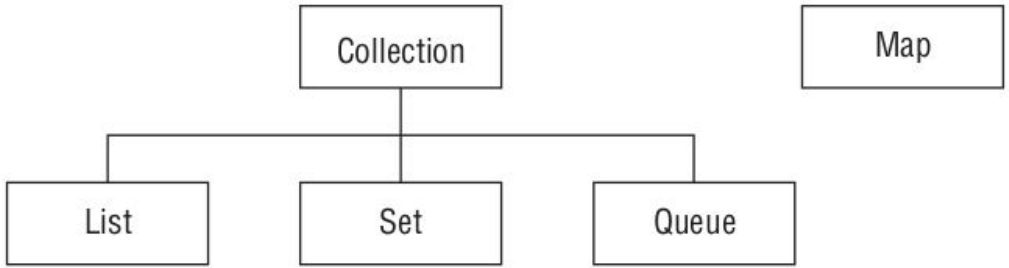
➢ Chúng ta cần tới generics để làm gì?

Ví dụ:

```
static void printNames(List list) {
    for (int i = 0; i < list.size(); i++) {
        Student student = (Student) list.get(i);
        System.out.println(student.getName());
    }
}
```

Giới thiệu về Collections

- Collection là 1 nhóm các đối tượng được chứa trong 1 object.
- Có 4 interface chính trong Java Collections Framework:
  - List
  - Set
  - Queue
  - Map



Giới thiệu về Collections

Một số methods phổ biến trong Collections:

- .add()
  boolean add(E element)
- .remove()
  boolean remove(Object object)
- .isEmpty() and size()
  boolean isEmpty()
  int size()
- .clear()
  void clear()
- .contains()
  boolean contains(Object object)

List Interface

- ➢ Sử dụng list khi quan tâm đến thứ tự của các phần tử và cho phép duplicate dữ liệu các phần từ trong đó.
- ➢ Implementations của List: ArrayList và LinkedList.

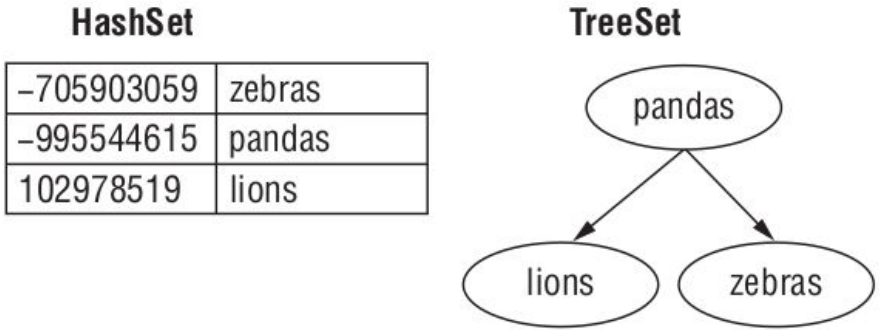
ArrayList	LinkedList
- Giống với array nhưng có thể thay đổi kích thước. - Ưu điểm - có thể truy cập đến phần tử bất kỳ với constant time. - Nhược điểm - thêm hoặc xóa phần tử thì sẽ chậm hơn so với truy cập đến phần tử.	- Implement cả List và Queue. - Có đầy đủ các methods trong List. Ngoài ra còn có các method để thêm và xóa các phần tử ở đầu và cuối list. - Ưu điểm - đó là có thể thêm hoặc xóa phần tử đầu hoặc cuối của list với constant time. - Nhược điểm - truy cập đến 1 phần tử bất kỳ thì phải duyệt qua các phần tử trước nó.

List Interface - List Methods

Method	Mô tả
void add(E e)	Thêm phần tử vào cuối danh sách.
void add(int index, E e)	Thêm phần tử vào vị trí index và di chuyển phần sau của List về cuối.
E get(int index)	Trả ra phần tử tại vị trí index.
int indexOf(Object o)	Trả ra index của phần tử đầu tiên matching hoặc là -1 nếu không tìm thấy.
int lastIndexOf(Object o)	Trả ra index của phần tử cuối cùng matching hoặc là -1 nếu không tìm thấy.
void remove(int index)	Xóa phần tử tại vị trí index và di chuyển phần còn lại của list trên trước.
E set(int index, E e)	Thay thế phần tử tại vị trí index và trả ra phần tử vừa bị thay thế.

Set Interface

- Sử dụng set khi <u>không cho phép duplicate dữ liệu</u> các phần từ trong đó.
- Implementations của Set: HashSet, LinkedHashSet và TreeSet.



Queue Interface

- Sử dụng queue khi các phần tử trong đó cần phải được xử lý theo một thứ tự ưu tiên nhất định.
- Implementations của Queue: ArrayDeque, LinkedList và PriorityQueue.

Queue Interface - ArrayDeque

Method	Mô tả
boolean add(E e)	Thêm phần tử vào cuối queue và trả ra true hoặc 1 exception.
E element()	Trả ra phần tử tiếp theo trong queue hoặc throws 1 exception nếu queue rỗng.
boolean offer(E e)	Thêm phần tử vào queue. Nếu thành công thì trả ra true, không thì false.
E remove()	Xóa và trả ra phần tử tiếp theo hoặc throws 1 exception nếu queue rỗng.
void push(E e)	Thêm 1 phần tử vào đầu queue.
E poll()	Xóa và trả ra phần tử vừa xóa hoặc trả ra null nếu queue rỗng.
E peek()	Trả ra phần tử tiếp theo hoặc trả ra null nếu queue rỗng.
E pop()	Xóa và trả ra phần tử tiếp theo hoặc throws 1 exception nếu queue rỗng.

Map Interface

- Sử dụng map khi bạn muốn đinh danh giá trị bằng khóa.
- Implementations của Map: HashMap, TreeMap, Hashtable.

Map Interface

Method	Mô tả
void clear()	Xóa tất cả key và value trong map.
boolean isEmpty()	Kiểm tra xem map có rỗng hay không.
int size()	Trả ra số lượng cặp key value trong map.
V get(Object key)	Trả ra value được gắn với key hoặc null nếu không tìm thấy.
V put(K key, V value)	Thêm hoặc thay thế cặp key value. Trả ra value trước đó hoặc null.
V remove(Object key)	Xóa và trả ra value được gắn với key hoặc null nếu không tìm thấy.
boolean containsKey(Object key)	Kiểm tra key có tồn tại trong map không.
boolean containsValue(Object)	Kiểm tra value có tồn tại trong map không.
Set<K> keySet()	Trả ra Set keys trong map.
Collection<V> values()	Trả ra Collection values trong map.

Bài tập về nhà

1. Cho 2 danh sách chứa các số có thể duplicate. Viết chương trình lấy ra 1 danh sách duy nhất chứa tất cả các số đấy, không chứa phần tử duplicate và danh sách kq được sắp xếp theo thứ tự từ bé đến lớn.

VD:

Danh sách 1: 1,4,6,9,11,2,7,3,8

Danh sách 2: 1,3,2,4,5,10,11,6

Danh sách kq: 1,2,3,4,5,6,7,8,9,10,11

Project cuối khóa

- Ứng dụng đấu giá trực tuyến.
  - Xử lý nhiều người cùng đấu giá trong cùng 1 thời điểm.
  - Hệ thống phân quyền giữa các loại người dùng.
- Ứng dụng đặt đồ ăn, cho phép chia sẻ bill cho nhóm người.
  - Xử lý nhiều người cùng chọn món trong cùng 1 thời điểm.
- Ứng dụng quản lý link profile.
  - Tạo template profile.
  - Thống kê số lượt click của mỗi link profile.

Vd: https://bio.link/austinarcher