1. **HashTable (bảng băm)**

Các method insert và search rất nhanh (kể cả khi data lớn)

VD:

Đặt f(x) = x mod 10.

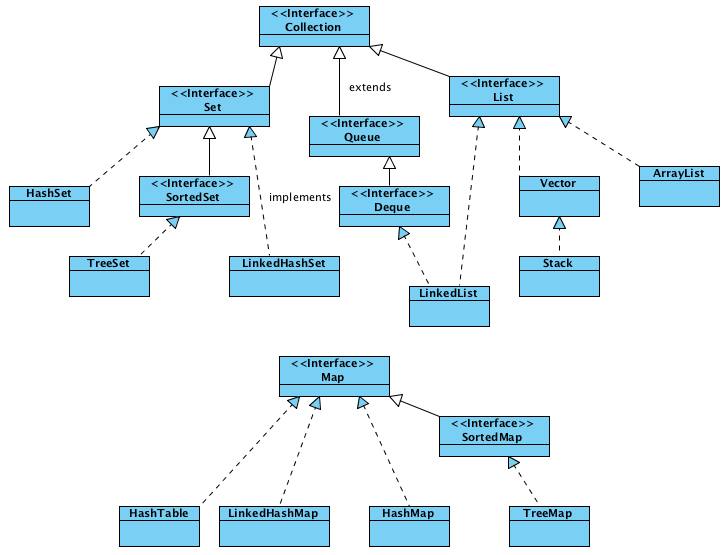
Khi đó

* x = 11 sẽ nằm ở vị trí số 1 (11 chia 10 dư 1)
* x = 20 sẽ nằm ở vị trí số 0 (20 chia 10 dư 0)
* x = 41 đáng ra phải nằm ở vị trí số 1 nhưng vị trí này đã có 11 chiếm mất rồi, nên nó sẽ tự động dịch sang 1 vị trí là số 2

… cứ tiếp tục như vậy

1. **Collection**

* Collection là một đối tượng java cung cấp sẵn để lưu trữ và thao tác trên dữ liệu có cấu trúc (chỉ quản lý các Object, ko quản lý dữ liệu primitive)
* Kiểu dữ liệu của Collection phải đồng nhất (kế thừa hoặc đa hình)



* 1. **List (interface)**

Là danh sách cấu trúc dữ liệu **tuyến tính** trong đó các phần tử được sắp sếp theo **1 thứ tự nhất định**

Các phần tử trong list có thể trùng nhau

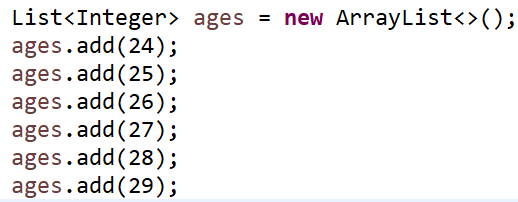
Có thể convert sang array

Các method hay dùng

* size()
* add()
* get(index)
* remove(index)
* clear()

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Mô tả** | **Tốc độ Insert, Remove** | **Synchronized** |
| ArrayList | * Giống mảng nhưng có thể thay đổi được kích thước mảng * Chỉ cần data và index * Phân mảnh nhiều (do các element phải sắp xếp liền nhau) | * Get phần tử rất nhanh (dựa vào index)   Do lưu trữ các phần tử trong list liền nhau trong bộ nhớ   * Insert và remove thì chậm do phải sắp sếp lại các phần tử | Không |
| LinkedList | * Hoạt động trên cơ sở của cấu trúc dữ liệu danh sách liên kết đôi (double-linked list) * Sử dụng Queue(FIFO) * Cần data và địa chỉ của con trỏ next và prev * Không phân mảnh nhiều do các element không nhất thiết phải sắp xếp liền nhau  |  |  |  |  | | --- | --- | --- | --- | | 1 | 2 | ////// | 3 | | ////// |  | ////// | 4 | | ///// | ////// | ////// |  | | //// | 5 |  |  |   *Cách sắp xếp data* | * Get phần tử chậm (không cho truy cập dựa vào index)   Do lưu trữ các phần tử trong list không liền nhau trong bộ nhớ   * Insert và remove thì nhanh | Không |
| Vector | Giống ArrayList |  | Có |
| Stack | * Mỗi stack có thể biểu diễn bằng array, cấu trúc (Struct), con trỏ (Point) và linked list * Ứng dụng trong StackTrace (Exception)   Biá»u diá»n ngÄn xáº¿p (Stack)  Các method hay dùng   * pop() * push() * peek(): lấy phần tử đầu của stack * isFull() , isEmpty() |  | Có |

VD:

****

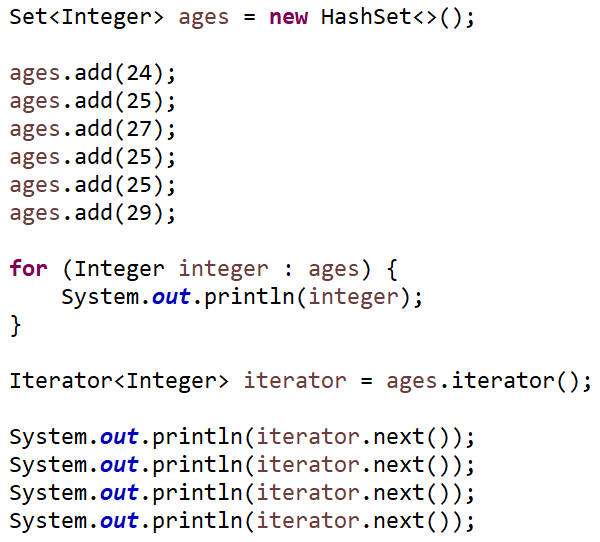
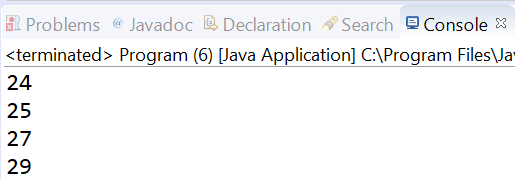
****

* 1. **Set (interface)**

Mỗi phần tử trong set chỉ xuất hiện duy nhất một lần

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Mô tả** | **Tốc độ Insert, Remove** | **Value null** | **Synchronized** |
| HashSet | Các phần tử được lưu trữ dưới dạng bảng băm (HashTable)  **Không biết thứ tự** của các phần tử 🡺 khi dùng vòng for in ra sẽ lộn xộn thứ tự |  |  | Không |
| LinkedHashSet | Giống **HashSet** nhưng được lưu theo **LinkedList** |  |  | Không |
| SortedSet (interface) | Tổ chức Set theo thứ tự tăng dần hoặc giảm dần |  |  | Không |
| TreeSet | Các phần tử đã được sắp sếp theo tree | Thêm, sửa, xóa và tìm kiếm 1 phần tử trong Set đều là O(log(n)) | Không | Không |
| EnumSet | Các phần tử trong set là các enum |  |  | Không |

VD:

* 1. **Queue (interface)**



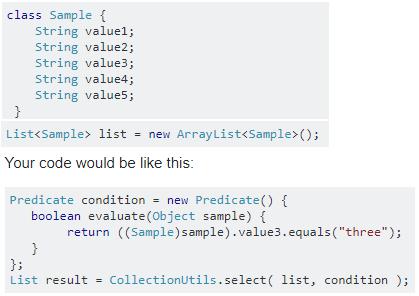
*Hoạt động theo FIFO*

Các method hay dùng

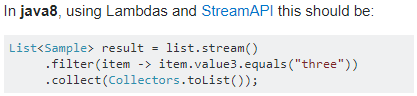
* enqueue(): thêm 1 phàn tử
* dequeue(): xóa 1 phần tử
* peek(): get phần tử ở đầu queue
* isFull()
* isEmpty()
  1. **Search Element in list**

VD: tìm kiếm những item có value3 = “three”

* **Java 7**

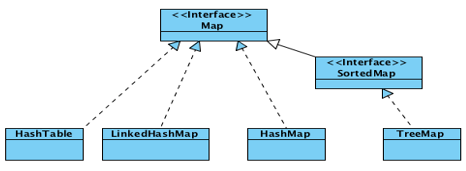
****

* **Java 8**

****

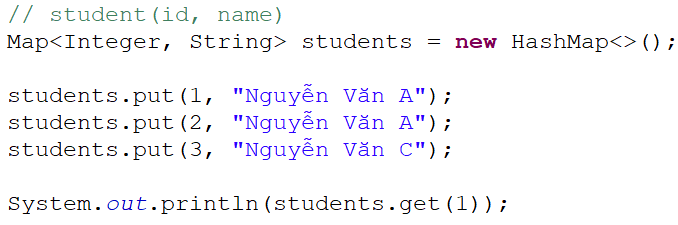
1. **Map**

* Quản lý dữ liệu theo dạng key - value.
* Truy cập data theo key
* Giống Set nhưng giá trị các phần tử trong map có thể trùng nhau nhưng key thì phải khác nhau
* Chuyển đổi qua Set thông qua method keySet();

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Mô tả** | **Tốc độ Inserrt, Remove** | **Value null** | **Synchronized** |
| HashMap | * Các key được lưu trữ dưới dạng HashTable * Không biết trước được thứ tự của các phần tử * Được Cài đặt theo LinkedHashMap và TreeMap |  | Có | Không |
| HashTable | * Giống HasMap nhưng không phải cài đặt của LinkedHashMap và TreeMap * không được sắp xếp... * Hastable cũ hơn HashMap |  | Không | Có |
| LinkedHashMap | Giống HashMap nhưng được lưu theo LinkedList |  |  |  |
| SortedMap (interface) | Tổ chức Map theo thứ tự tăng dần hoặc giảm dần | Thêm, sửa, xóa và tìm kiếm 1 phần tử trong Set đều là O(log(n)) | Không |  |
| TreeMap | Các key đã được sắp sếp theo tree | Thêm, sửa, xóa và tìm kiếm 1 phần tử trong Map đều là O(log(n)) | Không |  |
| EnumMap | Các key là enum |  |  |  |

VD:



**Override equals(), hashCode() trong Hash Collection**

Xem thêm trong equals() và hashCode() ở bài 3

1. **Comparing**

Xem lại ở Lesson 3

1. **Runtime & compileTime**

Compiletime là Build Project

Runtime là lúc Run Project

1. **Generic**

* Có từ Java 5
* Là cách lập trình tổng quát cho phép một object hoạt động với nhiều **datatype khác nhau**.
* Ưu điểm
  + Kiểm tra dữ liệu chặt chẽ ở Compile-time
  + Hạn chế việc ép kiểu dữ liệu (Datatype casting) thủ công, không an toàn
  + Viết các thuật toán được sử dụng nhiều, dễ dàng thay đổi
* Hạn chế
  + Không thể khởi tạo generic với dữ liệu kiểu Primitive (datatype tham số phải là Object)
  + Không thể tạo instance cho datatype

T obj = new T(); 🡺 error

* + Không thể là static trong class

static T obj; 🡺 Kiểu T không thể là static

* + Không thể tạo, catch, throw đối tượng của parameterized types
  + Không thể overload các method trong một class

public class Example {

public void print(Set<String> strSet) { }

public void print(Set<Integer> intSet) { }

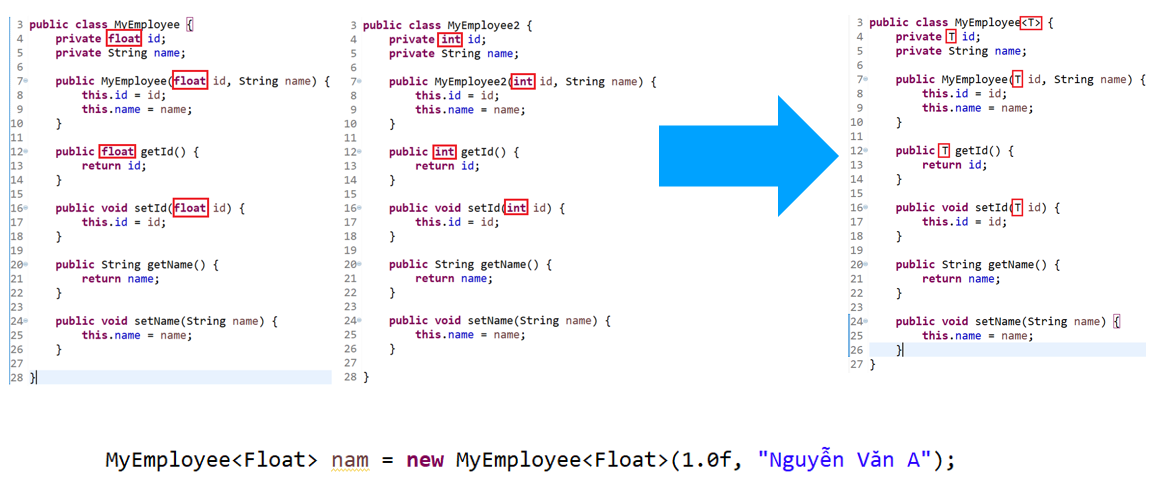
}

* Ký hiệu

|  |  |  |
| --- | --- | --- |
| **Ký** **tự** | **Ý** **nghĩa** | **Sử dụng cho** |
| E | Element | Collection |
| K, V | Key, Value | Object |
| T | Type | Object |
| N | Number | Object |

* Gồm 2 loại
  + **Class**

VD1:



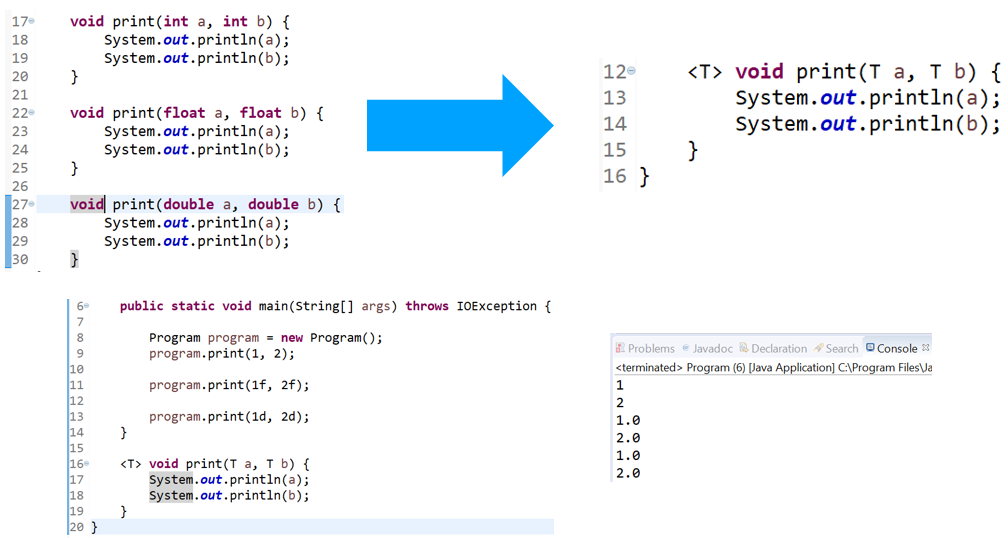
*Có thể chứa nhiều datatype tham số khác nhau và phân biệt nhau bởi dấu phảy*

VD2:



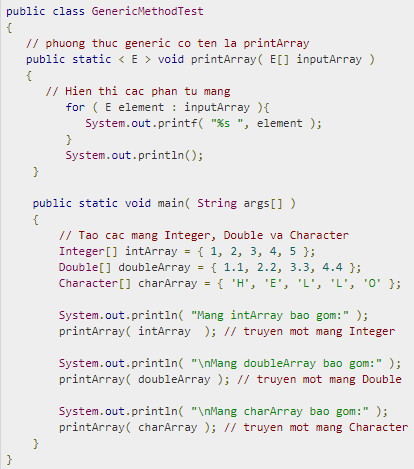
* + **Method**

VD1:

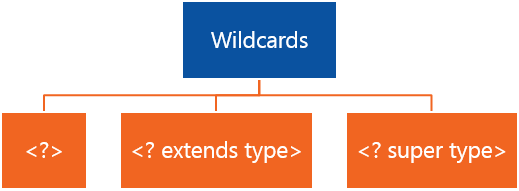


*Thay cho overloading*

VD2:



**Wildcard**



* Ký tự đại diện **<?>**

Chấp nhận bất kì DataType nào

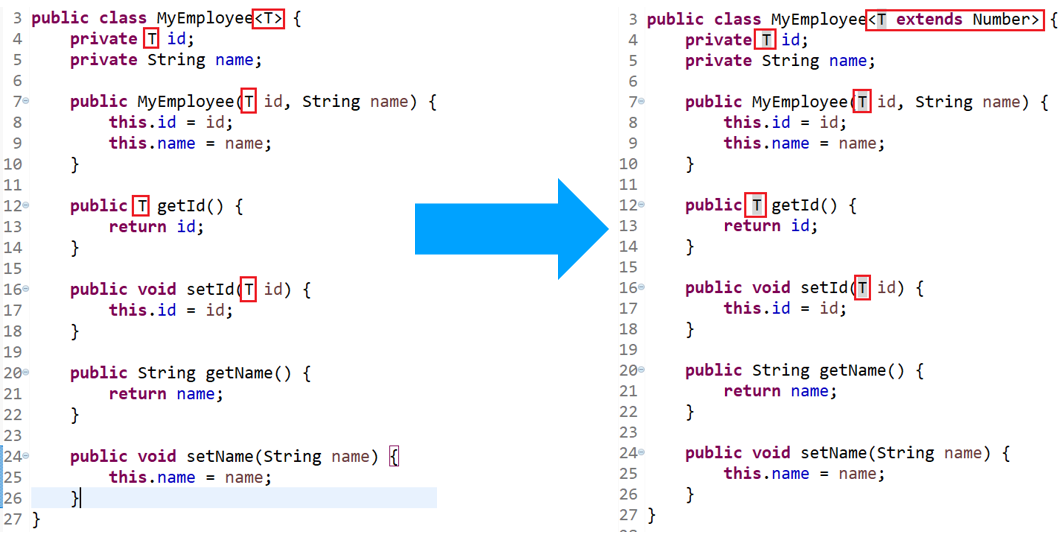
* Ký tự đại diện **<? extends type>**

Chấp nhận bất ký đối tượng nào miễn là đối tượng này kế thừa từ class type hoặc implement của type

* Ký tự đại diện **<? super type>**

Chấp nhận bất ký đối tượng nào miễn là đối tượng này là cha của type hoặc đối tượng của type

VD:



* **Reference**