Capgemini

28th Nov 2023

EFFECTIVE USE OF COLLECTIONS FRAMEWORK

Expert Connect Session

Speakers

Sapna Nema

Architect
Europe C&CA

Shraddha Sawant

Architect
Europe C&CA

AGENDA

- Overview of Data Structures
- What is Collection and Collections Framework?
- Choosing right Collection Type
- Best practices of using Java Collections
- Key Takeaways
- Quiz

WHO AM I?

- 12+ years of IT Experience
- Role - Technical Architect
- Expertise in
  - Software Design
  - Java & J2EE Development

Sapna Nema

- 14+ years of IT Experience
- Role - Technical Architect
- Expertise in
  - Software Design
  - Java & J2EE Development
  - Cloud

Shraddha Sawant

OVERVIEW OF DATA STRUCTURE

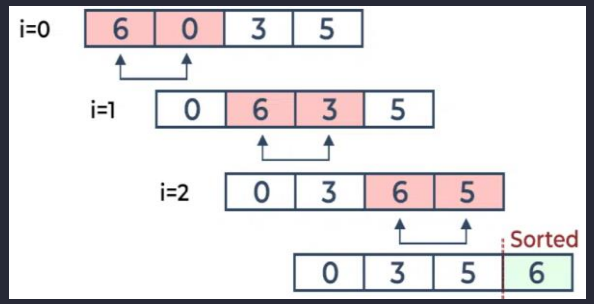
What is a data structure?

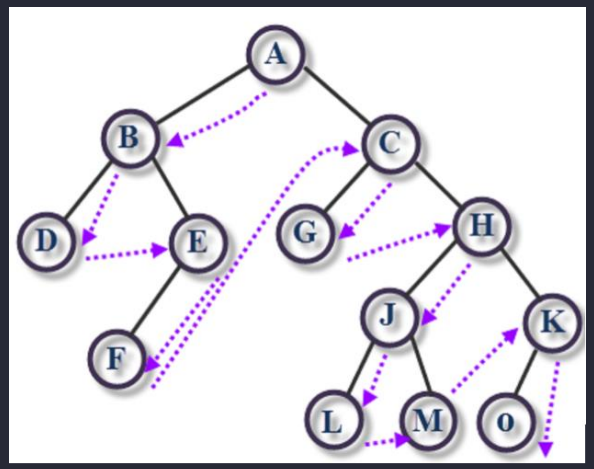
A data structure is a way for organizing, processing, retrieving and storing data.

Linear

- Array
- Linked List
- Stack
- Queue

Data structures types:





Non-Linear

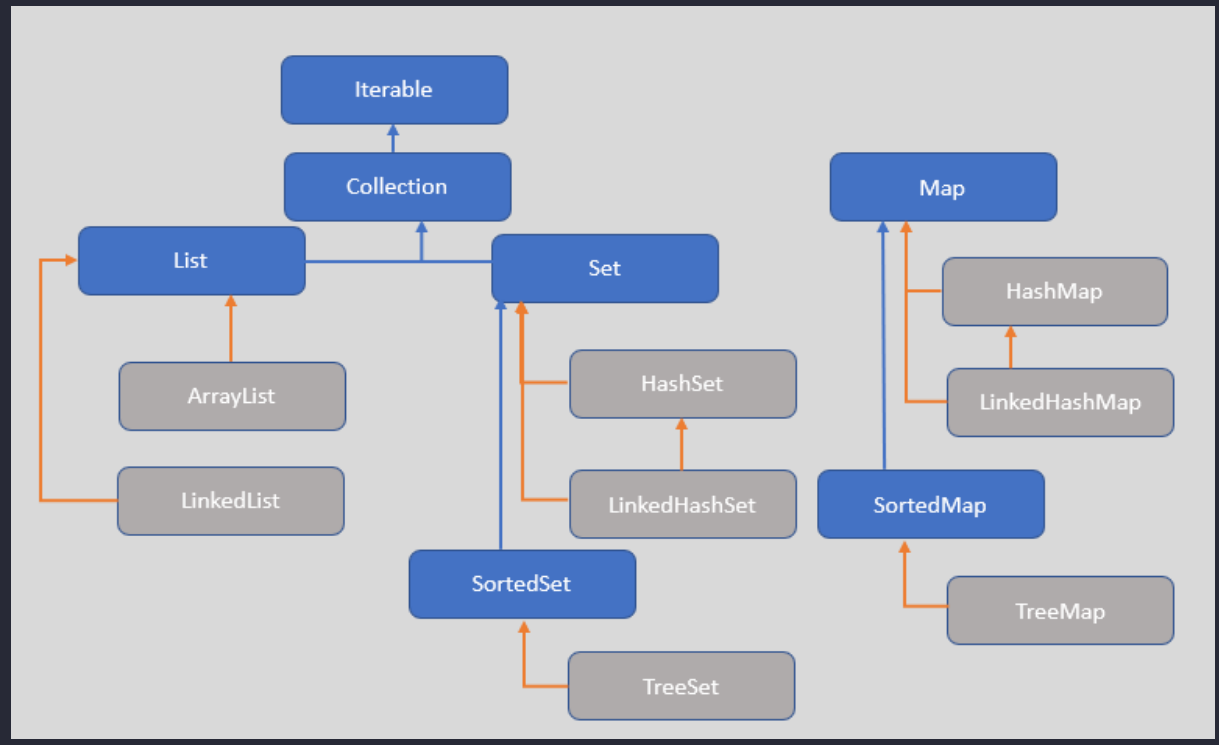
- Tree
- Graphs

WHAT IS COLLECTION AND COLLECTIONS FRAMEWORK

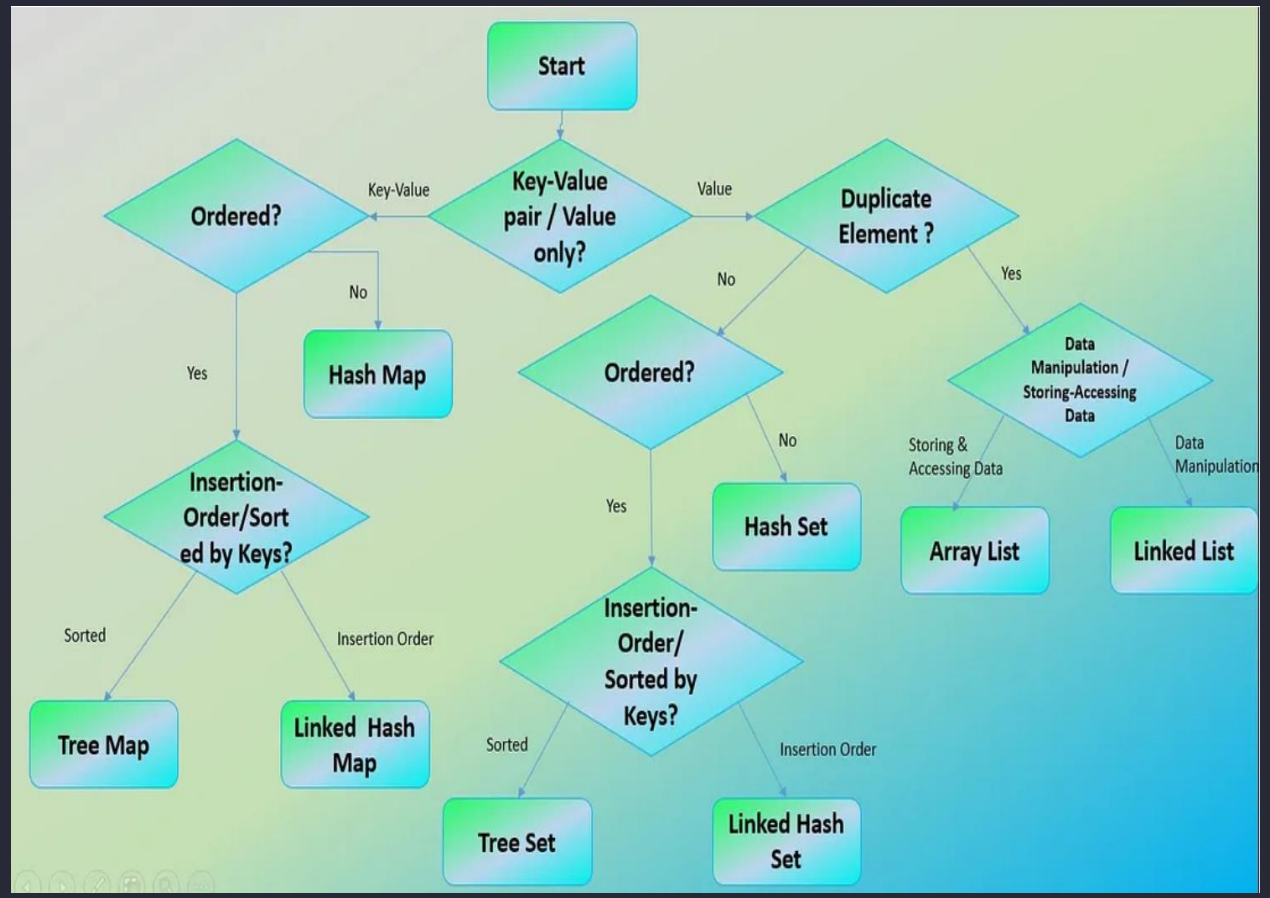
- A Collection(also known as container) is an object that contains a group of objects treated as a single unit.

- Any type of object s can be stored , retrieved and manipulated as elements of collections.

- Main parts of collection framework
  1. Interfaces
     Core Interfaces defining common functionality exhibited by collections.
  2. Implementation
     Concreate classes of the core interfaces providing data structures
  3. Operations
     Methods that perform various operations on collections.



CHOOSING RIGHT COLLECTION TYPE



	Insertion Order	Duplicate Elements	Sorting	Null Elements	Synchronized/ Thread Safety
LIST					
Array List	✓	✓	✗	✓	✗
Linked List	✓	✓	✗	✓	✗
SET					
HashSet	✗	✗	✗	✓	✗
Linked Hash Set	✓	✗	✗	✓	✗
Tree Set	✗	✗	✓	✗	✗
MAP					
Hash Map	✗	Unique Key & Duplicate values	✗	1 Null Key & Multiple Null values	✗
Linked Hash Map	✓	Unique Key & Duplicate values	✗	1 Null key & Multiple Null values	✗
Tree Map	✗	Unique Key & Duplicate values	✓	No Null key & Multiple Null values	✗

SOME EXAMPLES..

- List Interface and implementation

ArrayList

```java
List<String> userList = new ArrayList<>();
userList.add("Ram");
userList.add("John");
userList.add("Sita");
userList.add("Geeta");

// empty check with List.
if (CollectionUtils.isNotEmpty(userList)) {
    // Iteration through Iterator interface.
    Iterator<String> itr = userList.iterator();
    while (itr.hasNext()) {
        System.out.println(itr.next());
    }
}

for (String user : userList) {
    // ConcurrentModificationException
    userList.remove(0);
}

// Performing filter operation using Stream API.
List<String> filteredList = userList.stream()
        .filter(name -> name.equals("John"))
        .collect(Collectors.toList());
```

Output

Ram
John
Sita
Geeta

Do's

- ✓ Initialize with capacity if known.
- ✓ Check size before access. Use isEmpty() instead of size==0 for better performance.
- ✓ Use Generics (diamond operator)for type safety.

Don't

- Avoid manual resizing.
- Avoid insertion and deletion while accessing it. (it will cause concurrent modification exception).

SOME EXAMPLES..

- Map Interface and implementation

  HashMap

  - Declaration and initialization
    HashMap<String, Integer> myMap = new HashMap<>();
  - Adding Key value pair
    myMap.put("John", 25);
    myMap.put("Alice", 30);
  - Accessing Values
    int age = MyMap.get("Alice");

Do's

- ✓ Handling null using isEmpty() method.
- ✓ Iterate with EntrySet.
- ✓ Use generics for type safety.
- ✓ Override the hashcode() and equals() method in case of custom object key.

Don't

- Avoid bad design for hashcode() and equals().

BEST PRACTICES

Key points	Purpose	Example/comment
Use interface instead of class	Provides more code flexibility. Used for type, method parameter,	List<String> names = new ArrayList<>();
Use generics	Ensure type safety	Map<String, Employee> map = new HashMap<>();
Handling Null and Empty Collections	To prevent unexpected errors/behavior do not return null	Collections.empty();

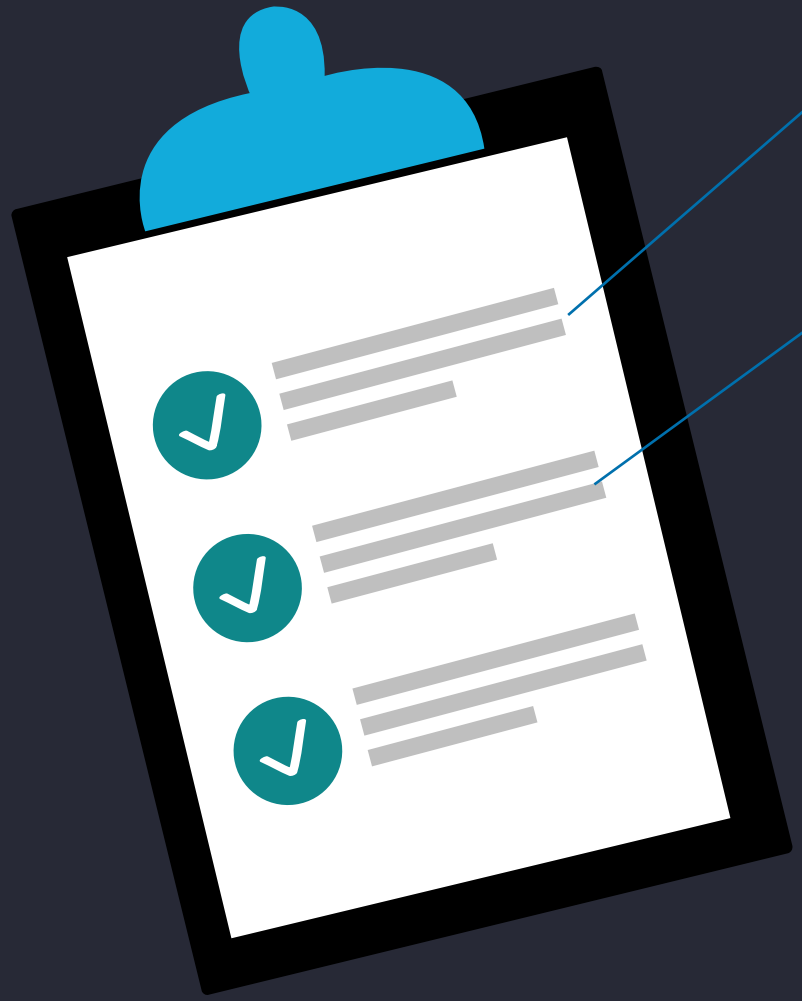
Efficient Iteration strategy	To eliminate potential bugs and for better readability	favor enhanced for loop, Iterator or foreach with lambda expression e.g. for (User user : userList) {
		// do something with user
		}

Reduce memory with right implementation	Memory usage can be decreased with right collection	Use Arraylist over LinkedList for frequent access
Utilizing Parallel Processing	Improves performance	Use Stream API: listOfNumbers.parallelStream().forEach(number -> //logic for number processing);
Use Collections Utility Methods	utility classes give us many useful functionalities	Collections.sort(), Collections.reverse(), etc
Use third-party collections libraries	provide additional features and better performance for specific use cases	Guava or Apache commons-collections

BEST PRACTICES

Key points	Purpose	Example/comment
Prefer best design for hashcode method	Improves hashmap performance	Overriding equals() and hashCode() properly for custom type
Remember to implement comparable interface for custom class	Provides natural ordering when sorted	Collections.sort(userList); //user class must implement comparable interface and override compareTo method
Prefer concurrent collections over synchronized wrappers	It's designed to provide maximum performance in concurrent applications	Instead of Collections.synchronizedXXX() use: - HashMap -> ConcurrentHashMap - ArrayList -> CopyOnWriteArrayList
Consider using wildcard types on input parameters	provides maximum flexibility	public double process(Collection<? extends Number> collection) //can accept a collection of any types which are subtypes of Number like Integer, Double etc

KEY TAKEAWAYS



- Collections minimizes programming effort by providing data structures and algorithms, so you don't have to write them.
- Increases performance by providing high-performance implementations of data structures and algorithms.
- List and Set are the main child interfaces of the collection interface.
- The Map interface is also part of the java collection framework, its preferred when values are stored in the form of keys and value pairs.
- Following best practices collection framework promotes code reusability and consistency.

Capgemini

THANK YOU

Quiz?

Feedback

HTTPS://FORMS.OFFICE.COM/E/MK7X9JI0VK

Feedback @ Expert Connect Session Sept 2023 | India Java Community

Sept 22, 2023

We would like to thank you for your participation. As a next step, we seek your feedback to understand how we can further improve such sessions in future.
For any clarification, please connect with Agarwal, Rajesh (rajesh.b.agarwal@capgemini.com)