BAHIR DAR UNIVERSITY

Bahir Dar Institute of Technology (BiT)

Faculty of Computing

Department of Software Engineering

Java Sports Management System Project

<- _____ ->

Group Members:

- **Ephrem Habtamu**
- **Abiriham Abayneh**
- **Melat Tesfaye**
- **Daniel Alemayehu**
- **John Assefa**
- **Kaleab Solomon**
- **Abiy Shiferaw**
- **Abriham Merkuz**

What are collection frameworks in java?

Before defining what collection frameworks in java are, we must look at collections and frameworks separately.

Collections in java refer to a collection of individual objects that are represented as a single unit. You can perform all operations such as searching, sorting, insertion, manipulation, deletion, etc., on Java collections just like you do it on data.

Frameworks are sets of classes and interfaces which provide a ready-made architecture. In order to implement a new feature or a class, there is no need to define a framework. However, an optimal object-oriented design always includes a framework with a collection of classes such that all the classes perform the same kind of task.

A framework:

- Provides readymade architecture.
- Represent a set of classes and interfaces.
- Is optional.

So what is a java collection framework?

A collection framework represents a unified architecture for storing and manipulating a group of objects. The Java Collections Framework is a collection of interfaces and classes which helps in storing and processing data efficiently. This framework has several useful classes which have numerous useful functions which makes a programmer task easy.

What is the difference between collections framework and collections?

- The Collections Framework is an interface, while Collections is a class.
- The Collection interface gives the standard functionality of data structures to List, Set, and Queue. The Collections class provides standard methods that you can use to search, sort, and coordinate collection elements.
- The Collection interface gives the standard functionality of data structures to List, Set, and Queue. The Collections class provides standard methods that you can use to search, sort, and coordinate collection elements.

Components of java collections framework

Interfaces: are defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a class. A Java interface contains static constants and abstract methods.

The basic interface of the collections framework is the collection interface which is implemented by all classes in the collection framework. It provides the basic operations for adding and removing elements in the collection.

Here are some methods of collections interface:

Method	Description
add(Object)	This method is used to add an object to the collection.
addAll(Collection c)	This method adds all the elements in the given collection to this collection.
clear()	This method removes all of the elements from this collection.
equals(Object o)	This method compares the specified object with this collection for equality.
max()	This method is used to return the maximum value present in the collection.

The above were some examples of methods of collections interface.

The collection hierarchy consists of **8** interfaces. These are:

Iterable Interface: This is the root interface for the entire collection framework. The collection interface extends the iterable interface. Therefore, inherently, all the interfaces and classes implement this interface. The main functionality of this interface is to provide an iterator for the collections. Therefore, this interface contains only one abstract method which is the iterator. It returns the

```
Iterator iterator();
```

Collection Interface: This interface extends the iterable interface and is implemented by all the classes in the collection framework. This interface contains all the basic methods which every collection has like adding the data into the collection, removing the data, clearing the data, etc. All these methods are implemented in this interface because these methods are implemented by all the classes irrespective of their style of implementation. And also, having these methods in this interface ensures that the names of the methods are universal for all the collections. Therefore, in short, we can say that this interface builds a foundation on which the collection classes are implemented.

As stated above, some methods of collection interface are add(), clear(), etc.

List Interface: This is a child interface of the collection interface. This interface is dedicated to the data of the list type in which we can store all the ordered collection of the objects. This also allows duplicate data to be present in it.

This list interface is implemented by various classes. these are

- ArrayList
- Vector
- Stack
- LinkedList

Queue Interface: maintains the first-in-first-out order. It can be defined as an ordered list that is used to hold the elements which are about to be processed.

The queue interface is implemented by the following classes.

- PriorityQueue
- ArrayDeque

Deque Interface: extends the Queue interface. In Deque, we can remove and add the elements from both the side. Deque stands for a double-ended queue which enables us to perform the operations at both the ends.

The Deque interface is implemented by the ArrayDeque class.

Set Interface: is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to store the duplicate items. We can store at most one null value in Set.

The set interface is implemented by the following classes:

- HashSet
- LinkedHashSet
- TreeSet

Sorted Set Interface: This interface is very similar to the set interface. The only difference is that this interface has extra methods that maintain the ordering of the elements. The sorted set interface extends the set interface and is used to handle the data which needs to be sorted.

The class TreeSet implements this class.

Map Interface: A map is a data structure that supports the key-value pair mapping for the data. This interface doesn’t support duplicate keys because the same key cannot have multiple mappings. A map is useful if there is data and we wish to perform operations on the basis of the key.

This interface can be implemented by the following classes:

- HashMap
- TreeMap

Classes: A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

we have already listed some of the classes as examples while defining the interfaces above. The following is the complete list of the standard collection classes

- AbstractCollection
- AbstractList
- AbstractSequentialList
- LinkedList
- ArrayList
- AbstractSet

- HashSet
- LinkedHashSet
- TreeSet
- AbstractMap
- HashMap
- TreeMap
- WeakHashMap
- LinkedHashMap
- IdentityHashMap

The following legacy classes defined by java.util are also worth mentioning

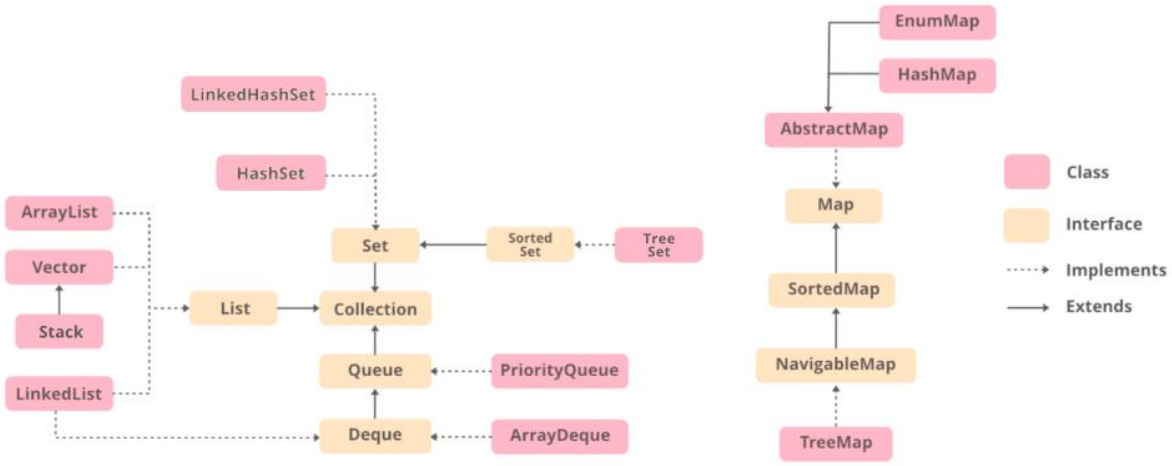
- Vector
- Stack
- Dictionary
- Hashtable
- Properties
- BitSet

So now that we have seen all the classes and interfaces in a collection framework, what is the use of these interfaces?

Interfaces allow collections to be manipulated independently of the details of their representation. In object-orientation languages, interfaces generally form a hierarchy.

We have seen that the interfaces and abstract classes that we have seen above have some methods. These methods can be overridden by concrete classes. These methods are called abstract methods.

The following figure shows the class hierarchy of collection framework.



References

- *Collections in Java - javatpoint*. (n.d.). Www.Javatpoint.Com. https://www.javatpoint.com/collections-in-java
- GeeksforGeeks. (2022, July 26). *Collections in Java*. https://www.geeksforgeeks.org/collections-in-java-2/
- *Java - Collections Framework*. (2022). Tutorialspoint. https://www.tutorialspoint.com/java/java_collections.htm#