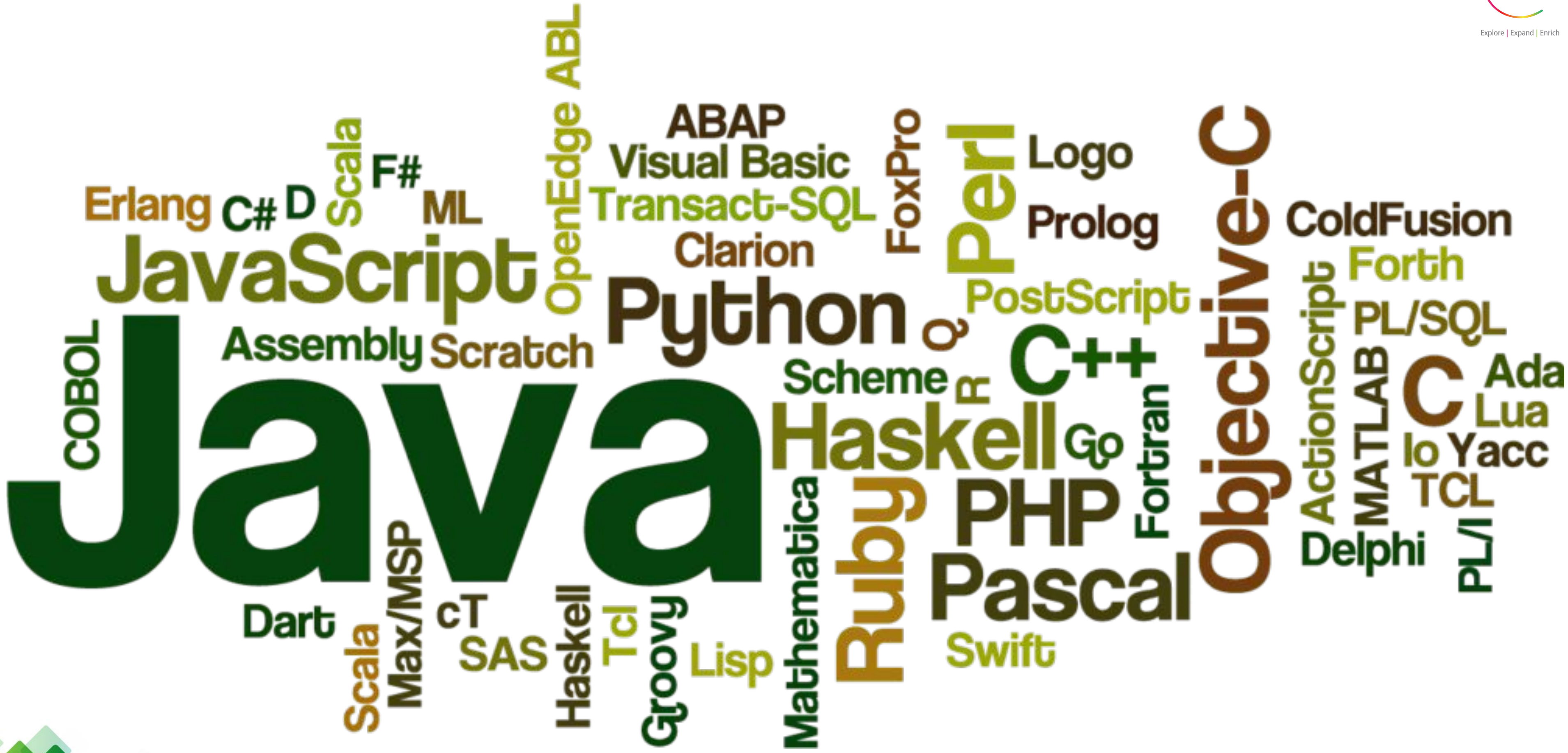
ETHNUS™

Explore | Expand | Enrich

COLLECTIONS WITH REAL TIME EXAMPLE


OpenEdge ABL
ABAP
Visual Basic
Transact-SQL
Clarion
FoxPro
Perl
Logo
Prolog
Objective-C
ColdFusion
Erlang
C#
D
Scala
F#
ML
JavaScript
Python
PostScript
Forth
PL/SQL
Q
C++
ActionScript
MATLAB
C
Ada
Lua
COBOL
Assembly
Scratch
Scheme
R
Haskell
Go
Fortran
Io
Yacc
TCL
Java
PHP
Delphi
PL/I
Ruby
Pascal
Mathematica
Dart
Scala
Max/MSP
cT
SAS
Haskell
Tcl
Groovy
Lisp
Swift

JAVA COLLECTIONS

- The data stored in the collection is encapsulated and the access to the data is only possible via predefined methods. For example the developer can add elements to an collection via a method

- Collections use internally arrays for there storage but hide the complexity of managing the dynamic size from the developer

- For example if your application saves data in an object of type People, you can store several People objects in a collection

IMPORTANT DEFAULT IMPLEMENTATIONS

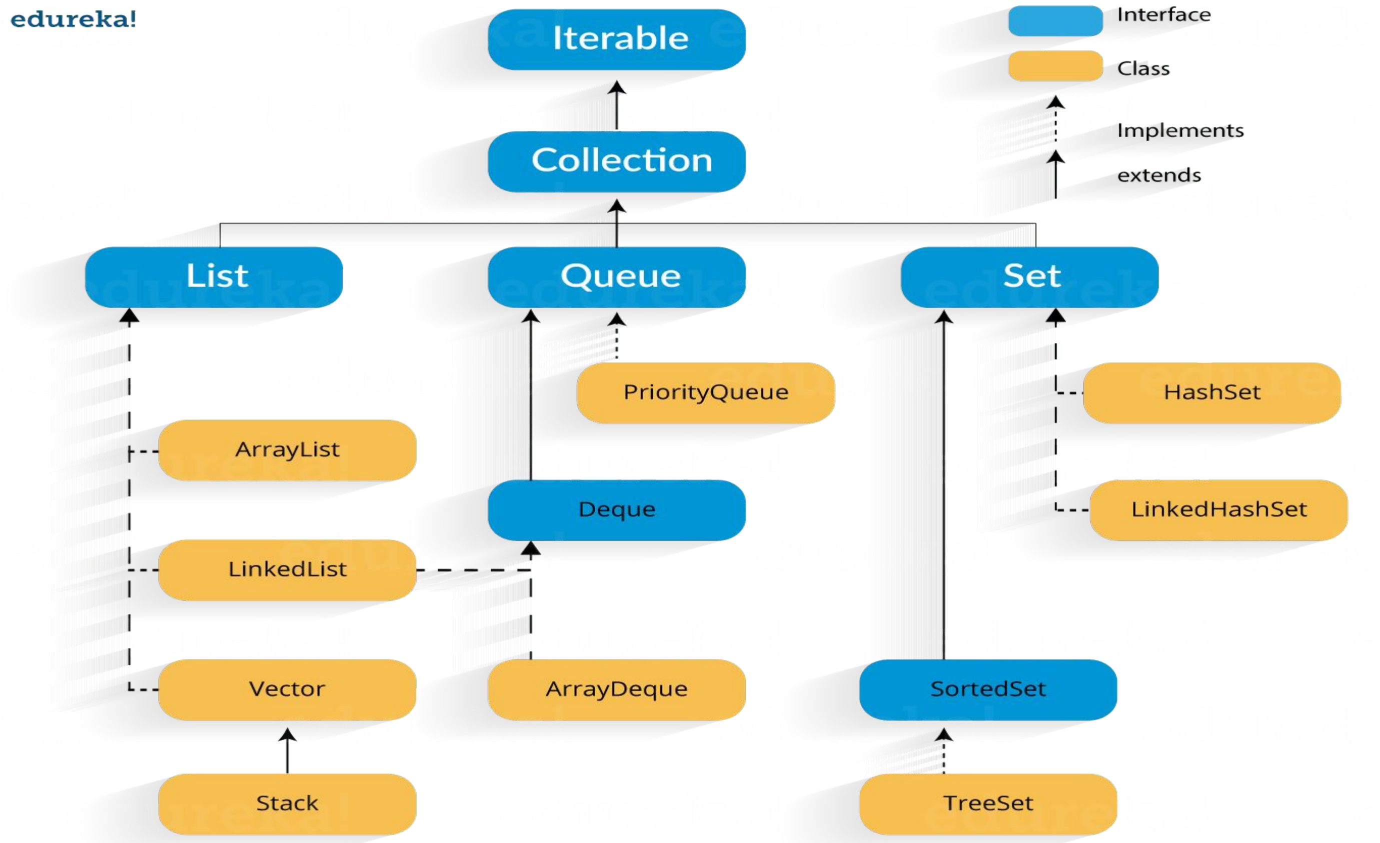
.
- Typical collections are

- Stacks
- Queues
- Lists
- Trees

- Java typically provides an interface, like List and one or several implementations for this interface, e.g., the ArrayList class and the LinkedList are implementations of the List interface

IMPORTANT DEFAULT IMPLEMENTATIONS

•



IMPORTANT DEFAULT IMPLEMENTATIONS

.

Type information with generics

- A class or interface whose declaration has one or more type parameters is a generic class or interface. For example List defines one type parameter List<E>

- Java collections should get parameterized with an type declaration. This enables the Java compiler to check if you try to use your collection with the correct type of objects. Generics allow a type or method to operate on objects of various types while providing compile-time type safety

JAVA COLLECTIONS : INTERFACE

Iterator interface : Iterator is an interface that iterates the elements. It is used to traverse the list and modify the elements. Iterator interface has three methods which are mentioned below

public boolean hasNext() – This method returns true if the iterator has more elements
public object next() – It returns the element and moves the cursor pointer to the next element.
public void remove() – This method removes the last elements returned by the iterator

JAVA COLLECTIONS: LIST

.
There are three components that extend the collection interface i.e List, Queue and Sets. Let’s learn about them in detail

A List is an ordered Collection of elements which may contain duplicates. It is an interface that extends the Collection interface. Lists are further classified into the following

- ArrayList
- LinkedList
- Vectors

ARRAY LIST

ArrayList is the implementation of List Interface where the elements can be dynamically added or removed from the list. Also, the size of the list is increased dynamically if the elements are added more than the initial size
(ArrayList object = new ArrayList ();)

Method Description

- boolean add(Collection c) - Appends the specified element to the end of a list
- void add(int index, Object element) - Inserts the specified element at the specified position
- void clear() - Removes all the elements from this list
- int lastIndexOf(Object o) - Return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element

ARRAY LIST

```
import java.util.*;
 class ArrayListExample{
 public static void main(String args[]){

 ArrayList al=new ArrayList();  // creating array list
 al.add("Jack");                // adding elements
 al.add("Tyler");
 Iterator itr=al.iterator();
 while(itr.hasNext()){
 System.out.println(itr.next());
 }
 }
 }
```

LINKED LIST

Linked List is a sequence of links which contains items. Each link contains a connection to another link

Syntax:

```
Linkedlist object = new Linkedlist();
```

Java Linked List class uses two types of Linked list to store the elements

- Singly Linked List
- Doubly Linked List

- **Singly Linked List**: In a singly Linked list each node in this list stores the data of the node and a pointer or reference to the next node in the list

LINKED LIST

Doubly Linked List: In a doubly Linked list, it has two references, one to the next node and another to previous node

- **Vectors** : Vectors are similar to arrays, where the elements of the vector object can be accessed via an index into the vector. Vector implements a dynamic array. Also, the vector is not limited to a specific size, it can shrink or grow automatically whenever required. It is similar to ArrayList, but with two differences

- Vector is synchronized

- Vector contains many legacy methods that are not part of the collections framework

REAL TIME USAGE OF JAVA COLLECTIONS

- suppose in your project you got a requirement like you have to sort the employee based on seniority and show the result ,now you have already list of employee with you,sort that list based on age property of employee ,in this case use collections.sort().in this case you are avoiding making any DB call.so this will give you faster result

- map collection being used to create in memory cache.suppose in your project you got requirement like first 100 records need show based on the default selection whenever user login in the system.In this case use map to store the 100 records in cache when your application gets up.with this you will give user good experience about usage of your application

THANK YOU