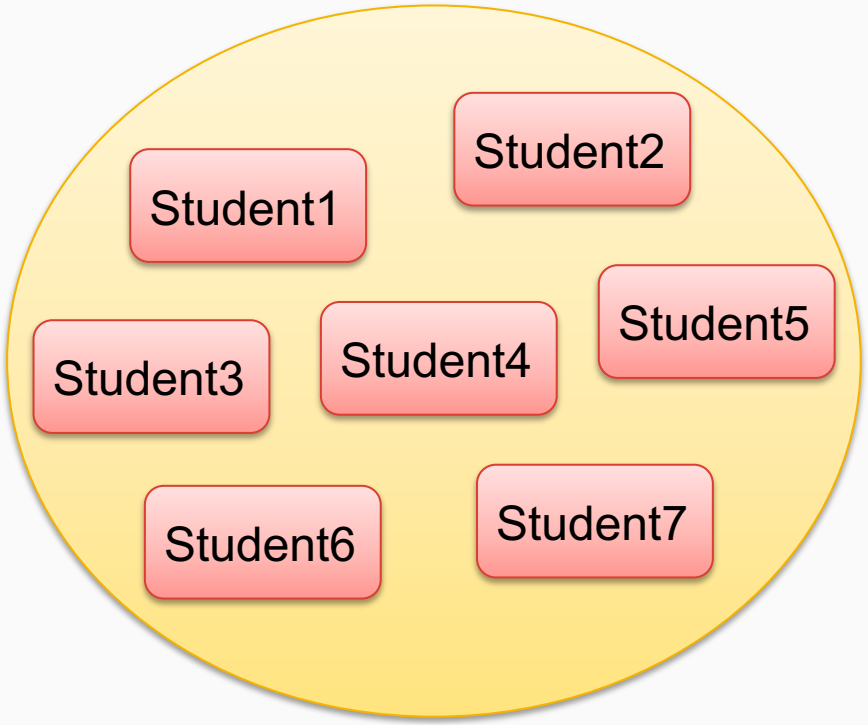
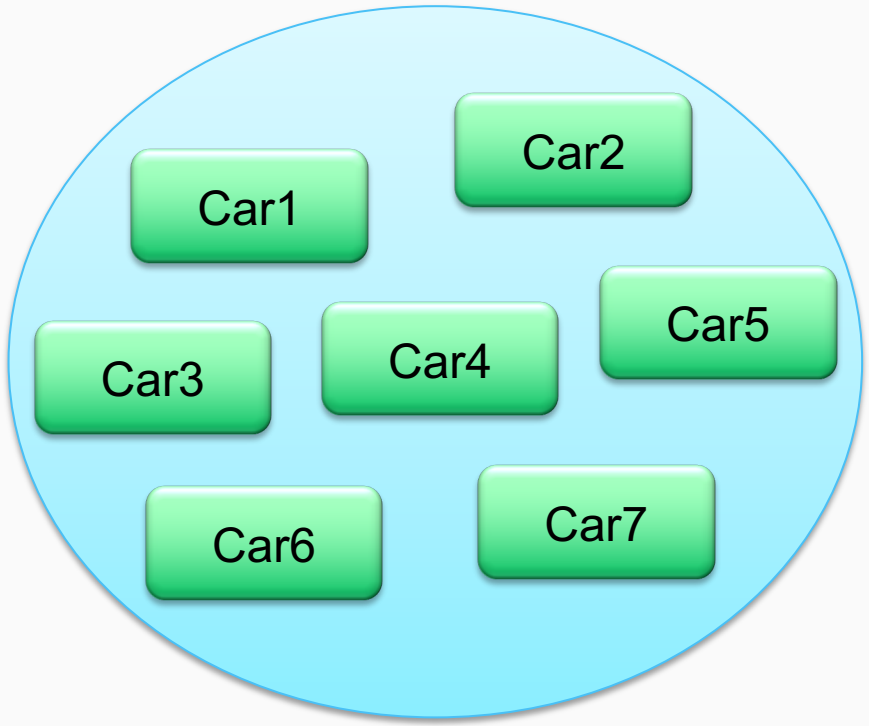
CYDEO

Collections

What Is Collection?

- Collection is a group of individual objects as a single entity





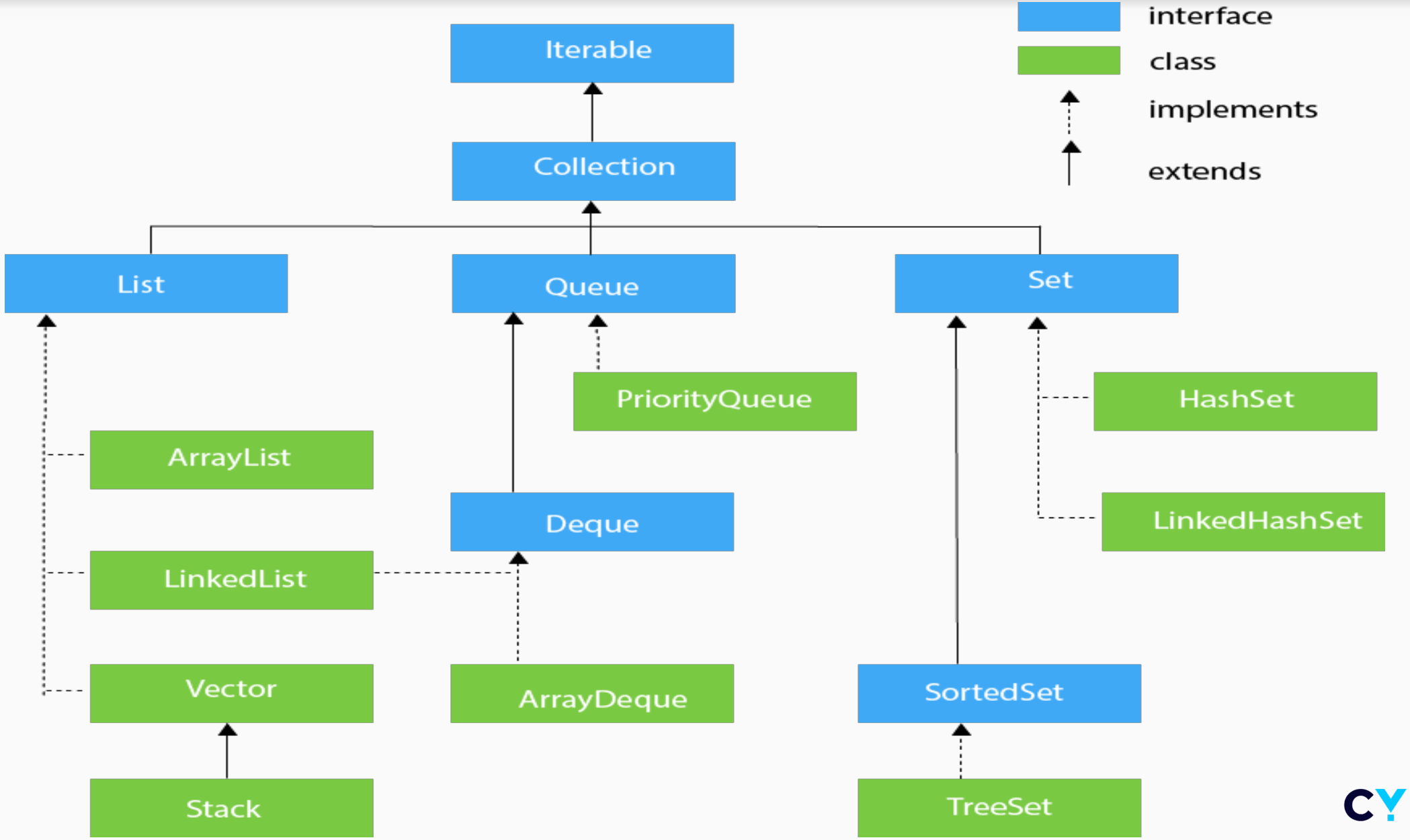
CYDEO

Collections

- Defines several classes and interfaces which can be used to represent a group of objects as single entity
- Growable in nature, can increase or decrease the size
- Can hold different non-primitive data types
- Standard data structure and there are ready methods to use

CYDEO

Collections Hierarchy



CYDEO

Collection

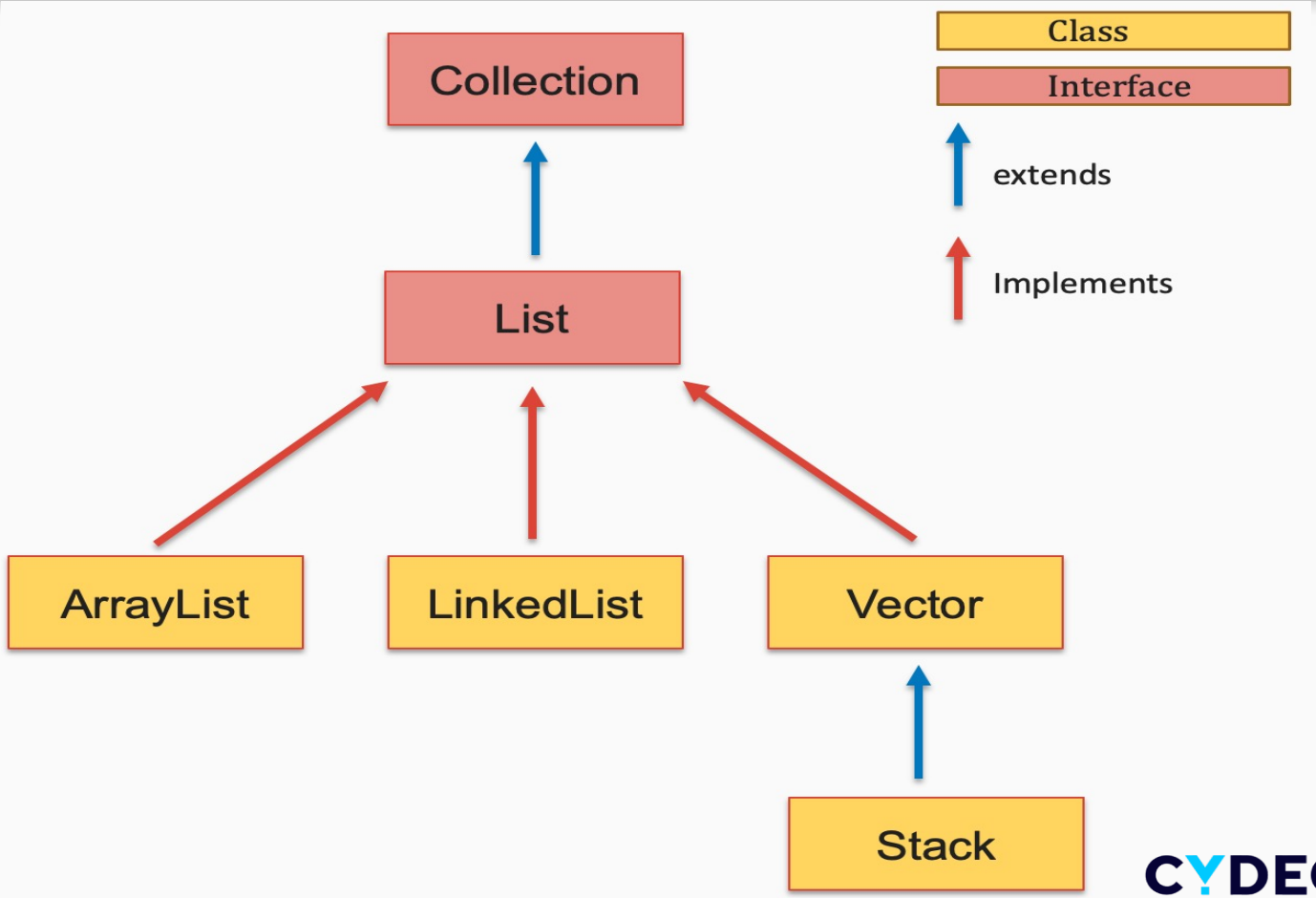
- Collection interface is considered as root interface of collection framework
- Collection interface defines the most common methods which are applicable for any collection object

Method Name	Method Name	Method Name
size()	isEmpty()	add()
addAll()	remove()	removeAll()
removeIf()	contains()	containsAll()
retainAll()	clear()	toArray()

CYDEO

List Interface

- Child interface of Collection
- Duplicates are allowed
- Insertion order preserved
- Has index numbers

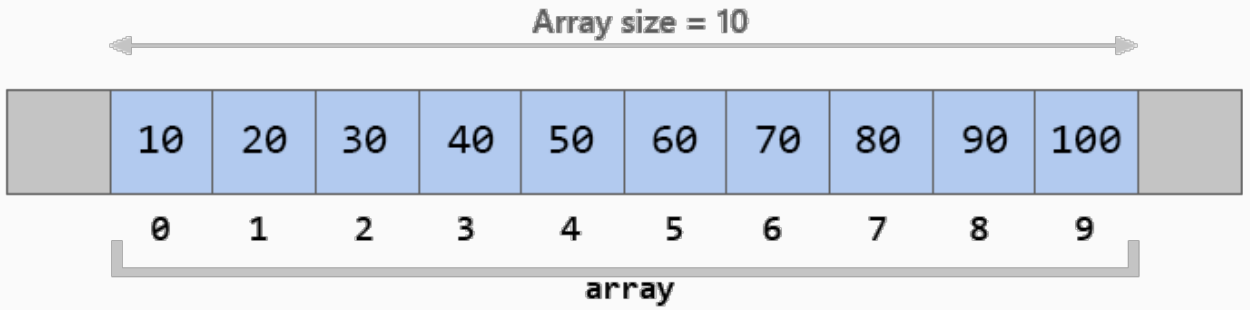


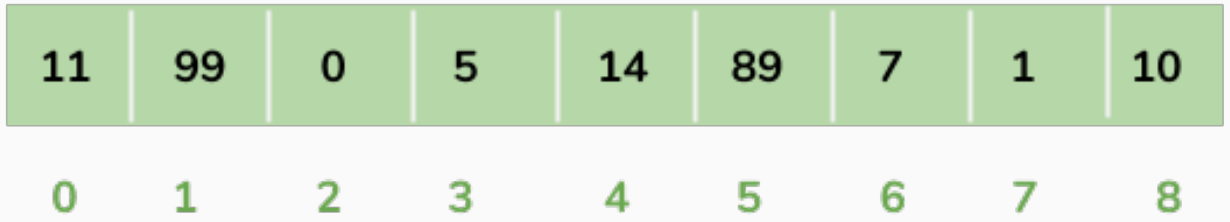

```
import java.util.List;
```

CYDEO

ArrayList

- Implements the List interface
- Internally uses array
- Retrieving the element is faster



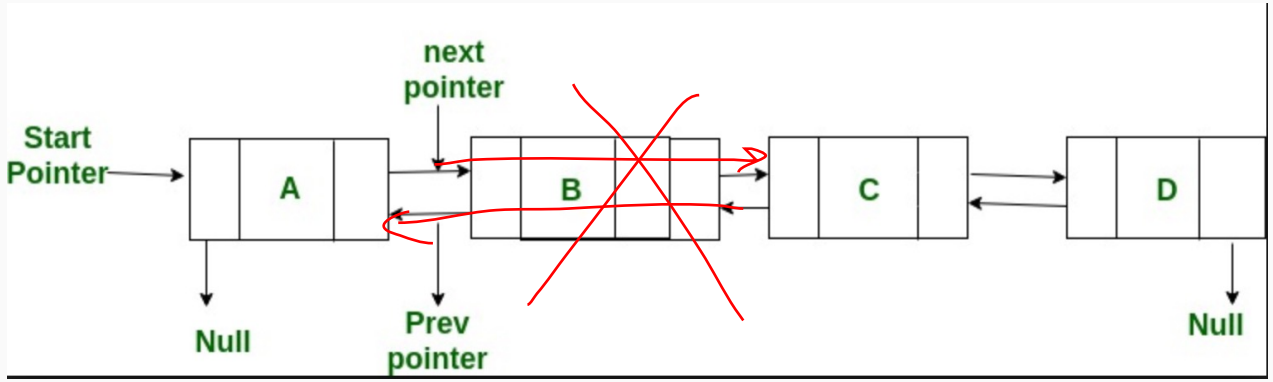


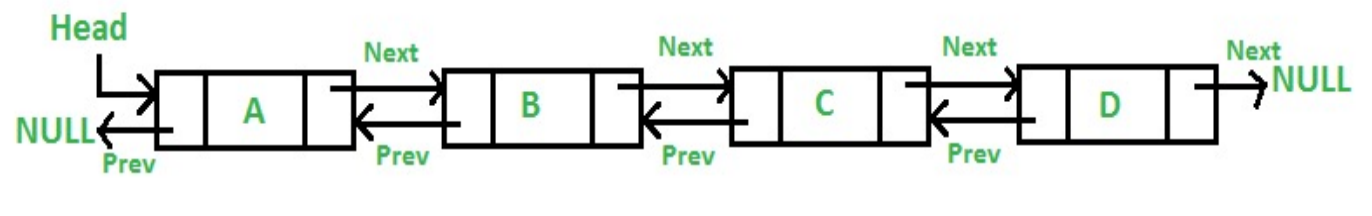

```
import java.util.ArrayList;
```

CYDEO

LinkedList Class

- Implements the List interface
- Internally uses doubly linked list
- Each value will have pointer to the next value
- Inserting and deleting the element is faster



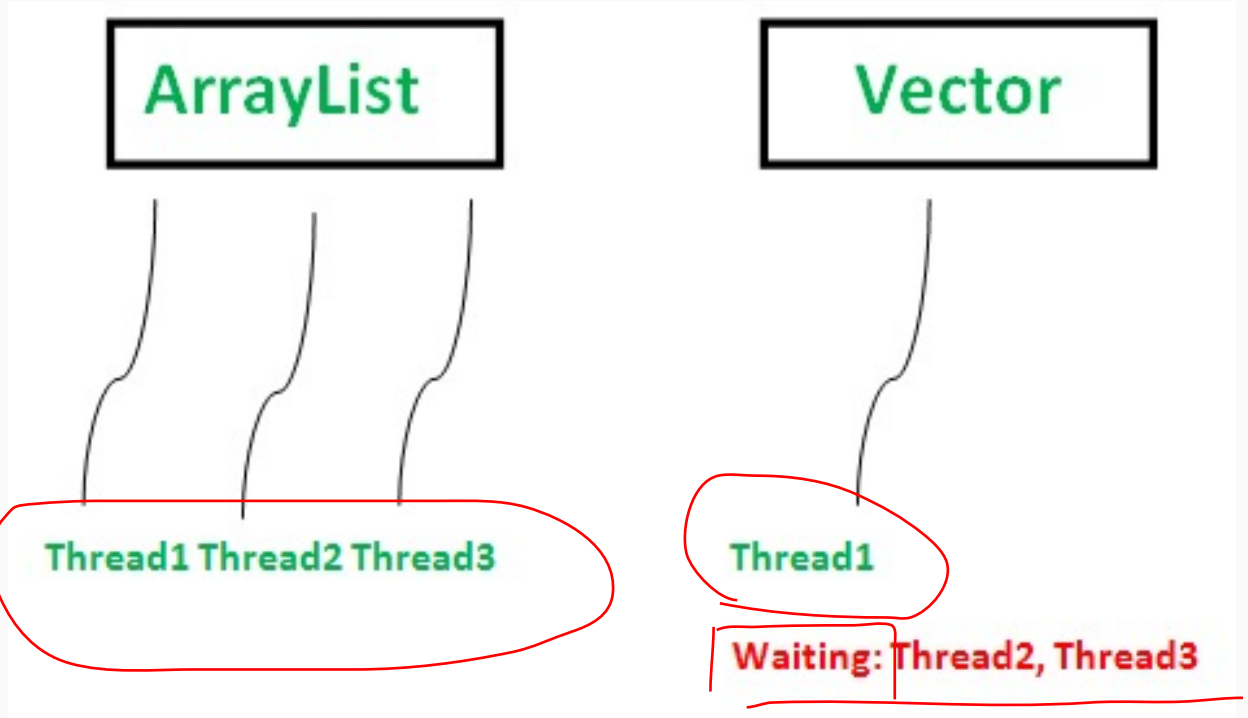



```
import java.util.LinkedList;
```

CYDEO

Vector Class

- Implements the List interface
- Internally uses array
- Is Synchronized (thread-safe)

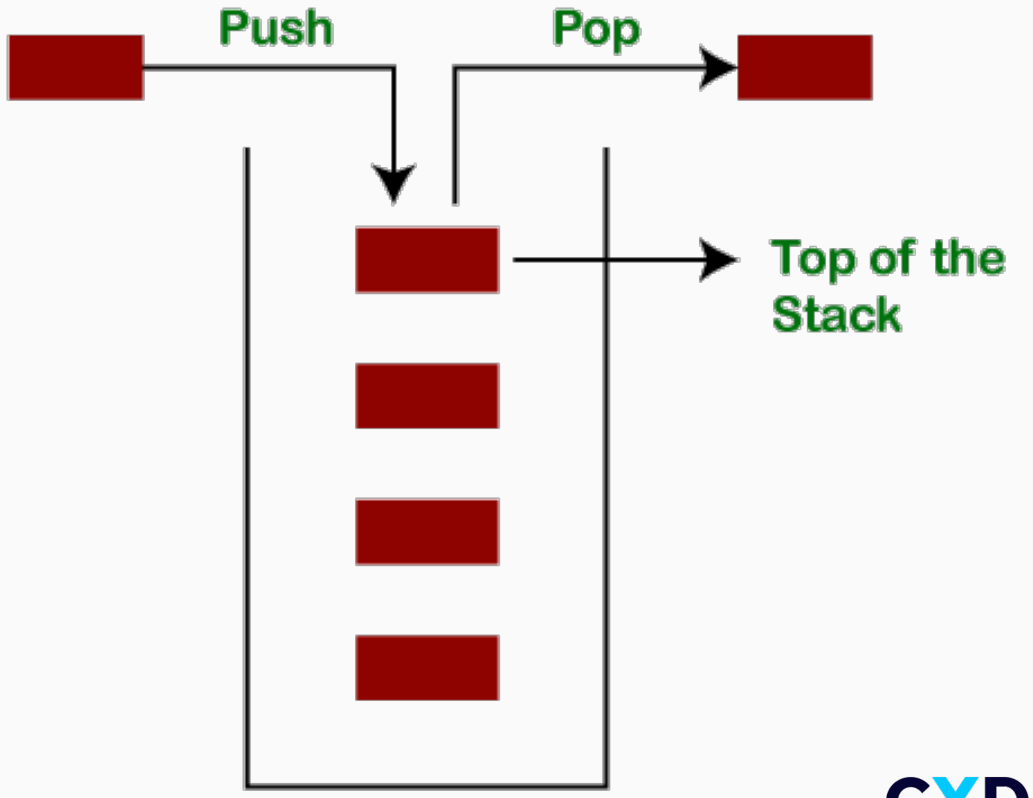



```
import java.util.Vector;
```

CYDEO

Stack Class

- Child class of Vector
- Is Synchronized (thread-safe)
- Last-in, first out order (LIFO)



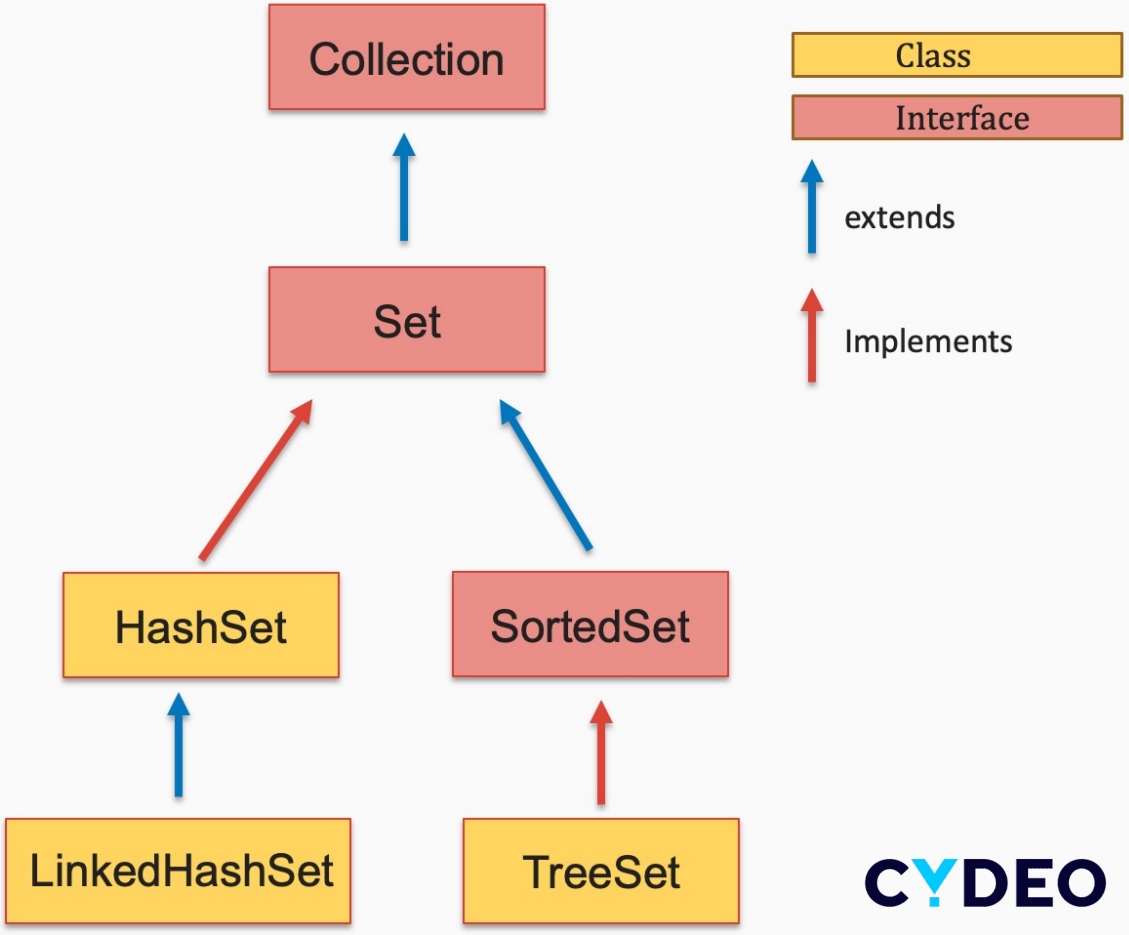

```
import java.util.Stack;
```

CYDEO

Set Interface

- Child interface of Collection
- Duplicates are not allowed
- Insertion order not preserved
- Does not have index

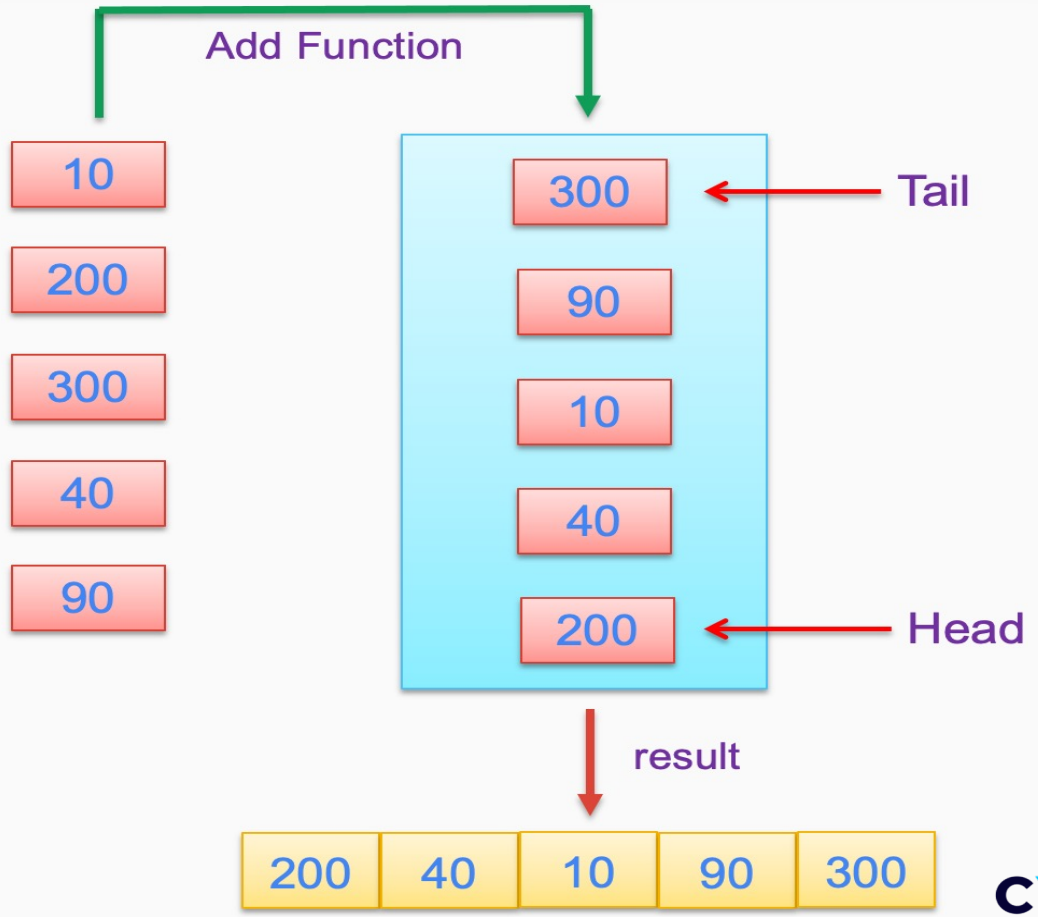
```
import java.util.Set;
```



CYDEO

HashSet Class

- Implements the Set interface
- Maintains the random order
- Accepts null values

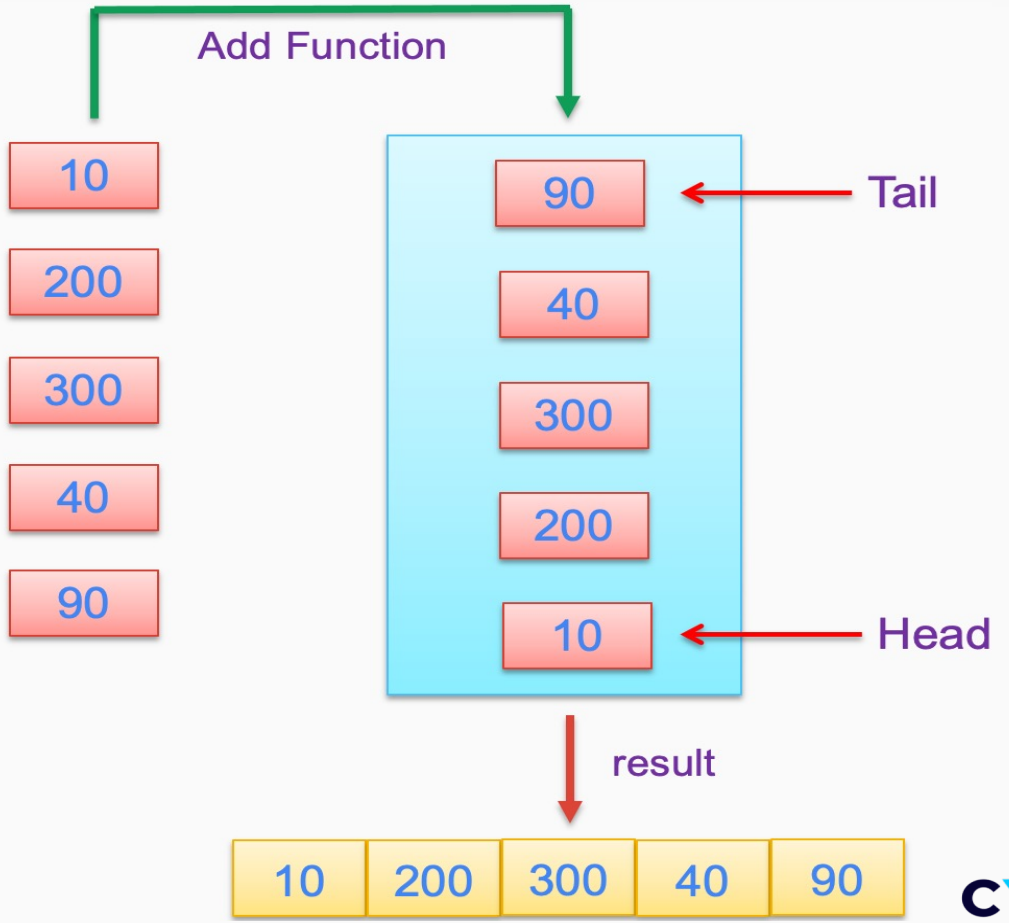



```
import java.util.HashSet;
```

CYDEO

LinkedHashSet Class

- Child class of HashSet
- Maintains the insertion order
- Accepts null values

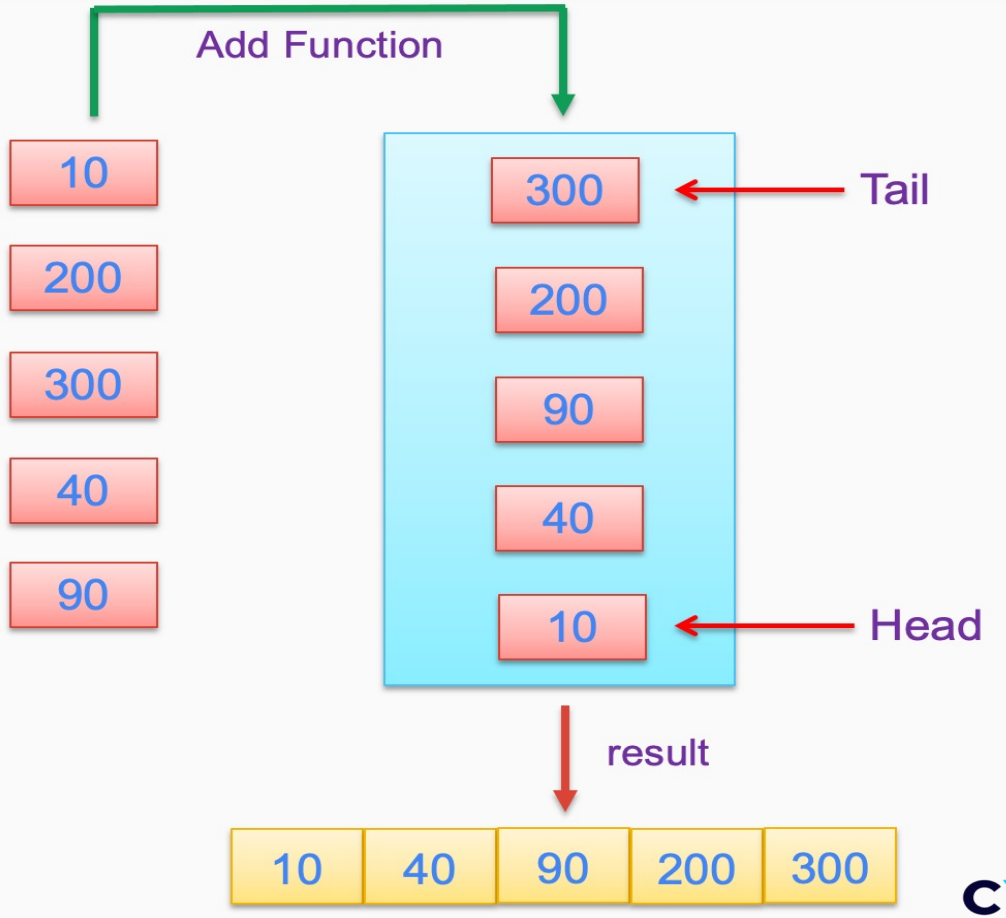



```
import java.util.LinkedList;
```

CYDEO

TreeSet Class

- Implements the SortedSet interface
- Maintains the sorted (Ascending) order
- Does not accept null values



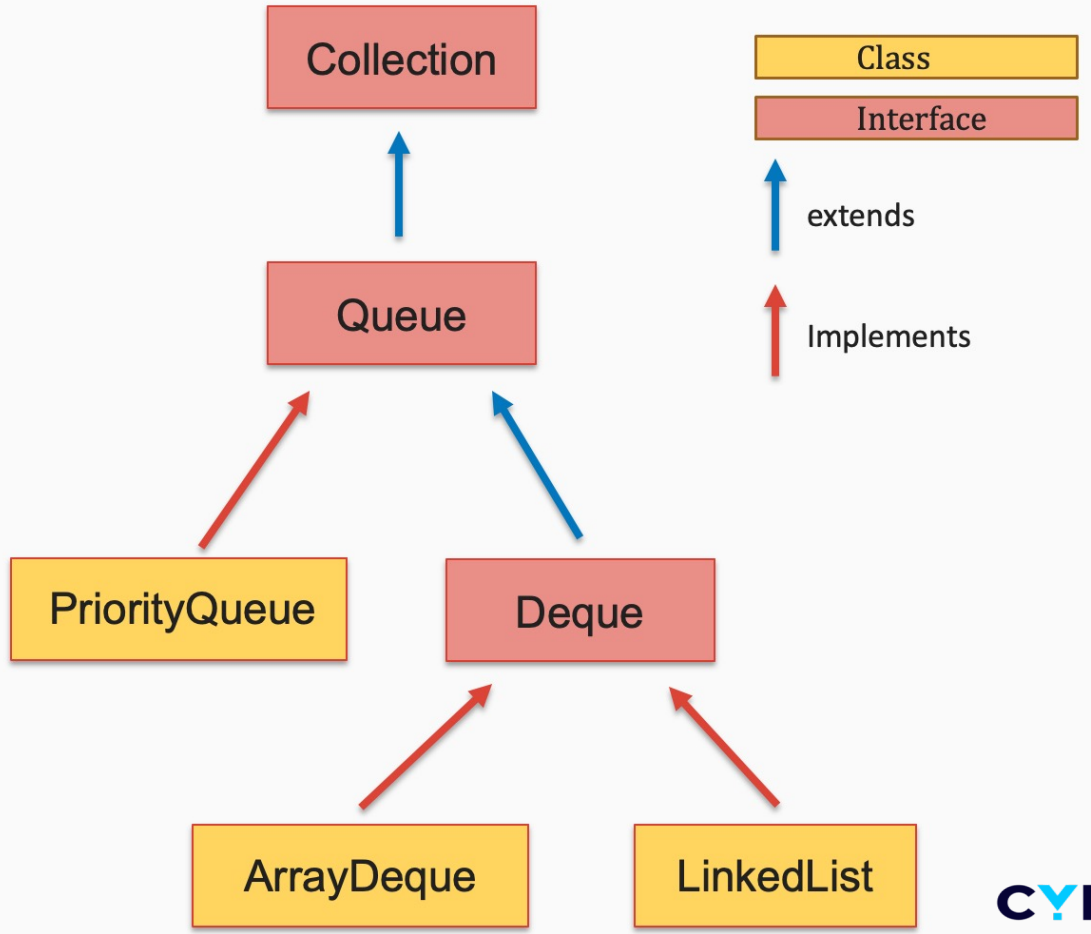

```
import java.util.TreeSet;
```

CYDEO

Queue Interface

- Child interface of Collection
- Duplicates are allowed
- Insertion order not preserved
- First-in, First out order (FIFO)

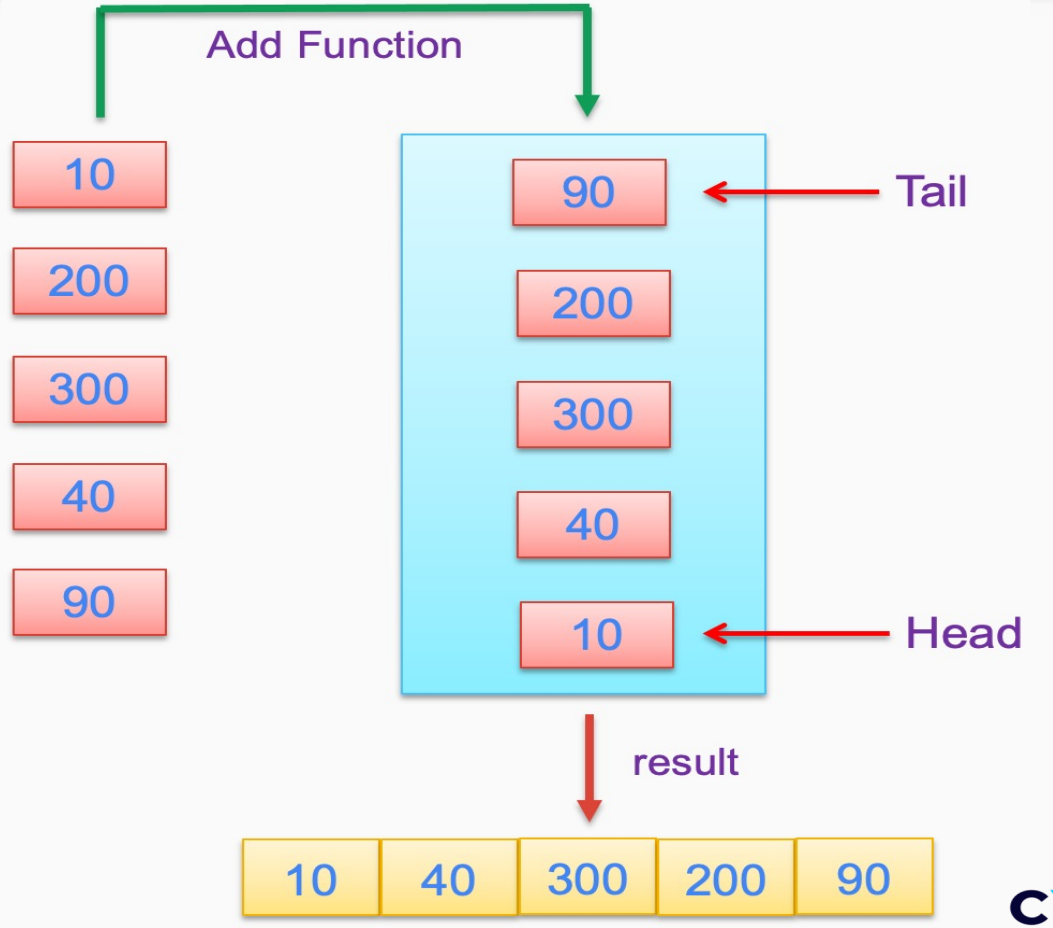
```
import java.util.Queue;
```



CYDEO

PriorityQueue Class

- Implements the Queue interface
- Maintains the random order
- Does not accept null values

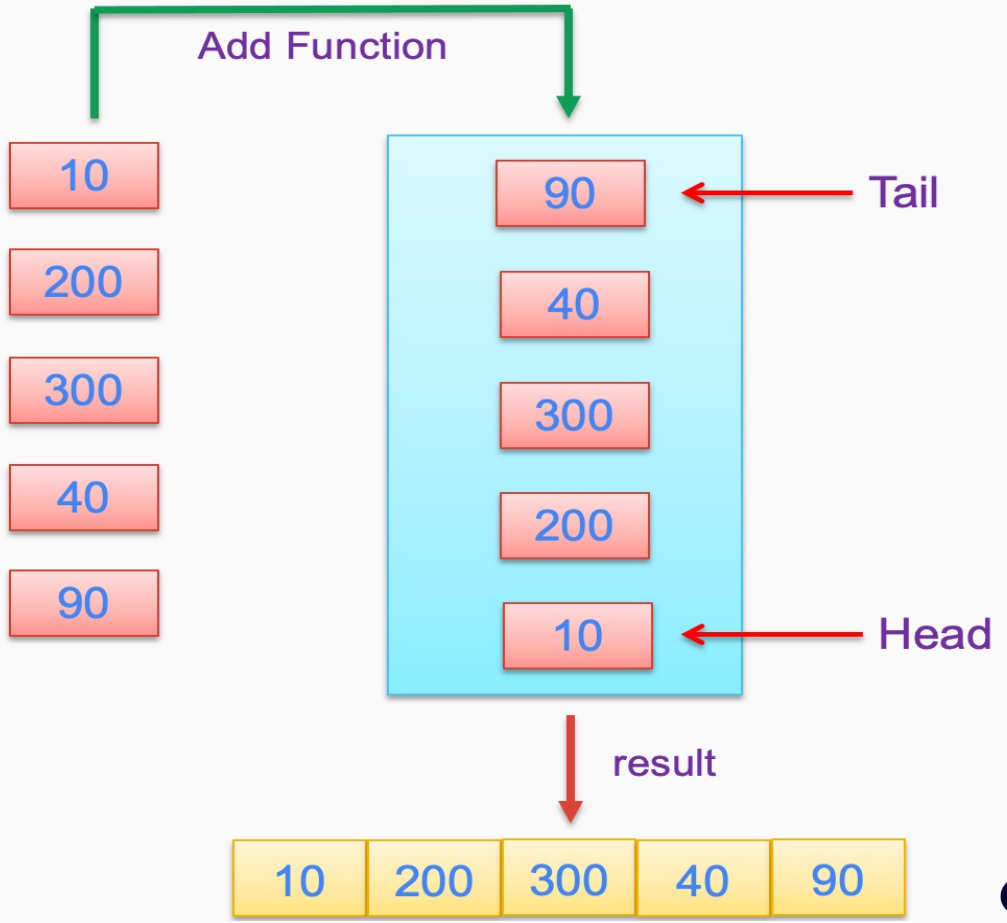



```
import java.util.PriorityQueue;
```

CYDEO

ArrayDeque Class

- Implements the Deque interface
- Maintain the insertion order
- Does not accept null values




```
import java.util.ArrayDeque;
```

CYDEO

List vs Set vs Queue

List	Set	Queue
Duplicates are allowed	Duplicates are not allowed	Duplicates are allowed
Insertion order preserved	Insertion order not preserved	Insertion order not preserved
Has index	Does not have index	Does not have index

When to use List, Set, and Queue?

- List: If we want to represent a group of individual objects as a single entity where duplicates are allowed, and insertion order preserved
- Set: If we want to represent a group of individual objects as a single entity where duplicates are NOT allowed, and insertion order NOT preserved
- Queue: If we want to represent a group of individual objects prior to processing

Iterable

- Parent interface of Collection interface
- Allows the object to be Iterated by using Iterator
- Iterator is used when we want to remove elements whilst you iterate over a collection

How to use Iterator?

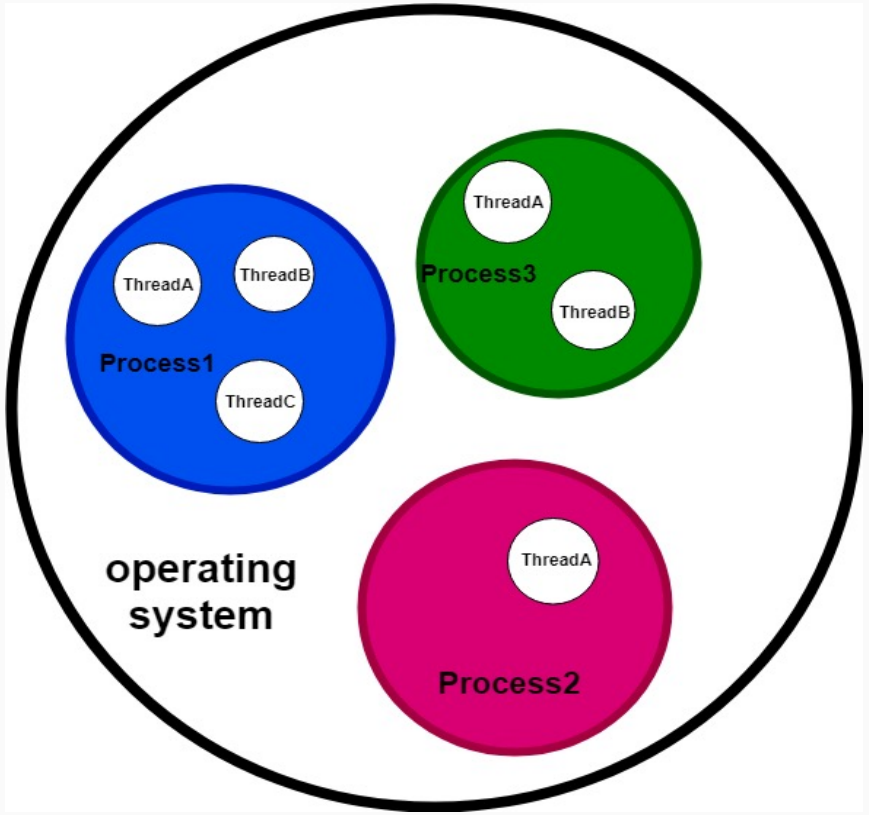
- We can obtain the Iterator by using the iterator() method of an Iterable
- Iterator provides the ready methods that can be used while traversing the collections:
    - hasNext( )
    - next( )
    - remove( )

CYDEO

Process, Thread and Synchronized

Process

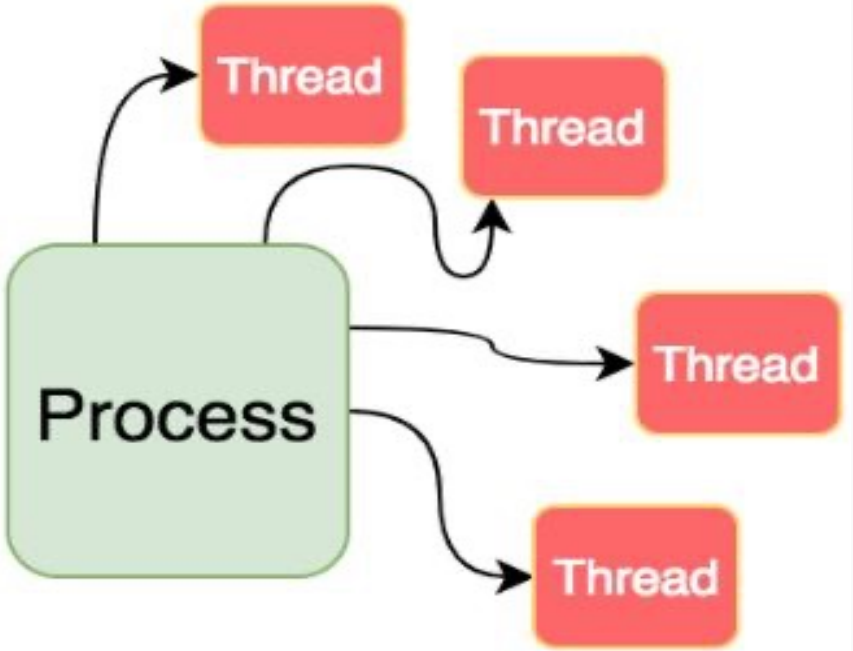
- Programs that are scheduled for execution in the CPU
- An instance of a program running on a computer.
- A process has code, data, heap and other segments



CYDEO

Thread

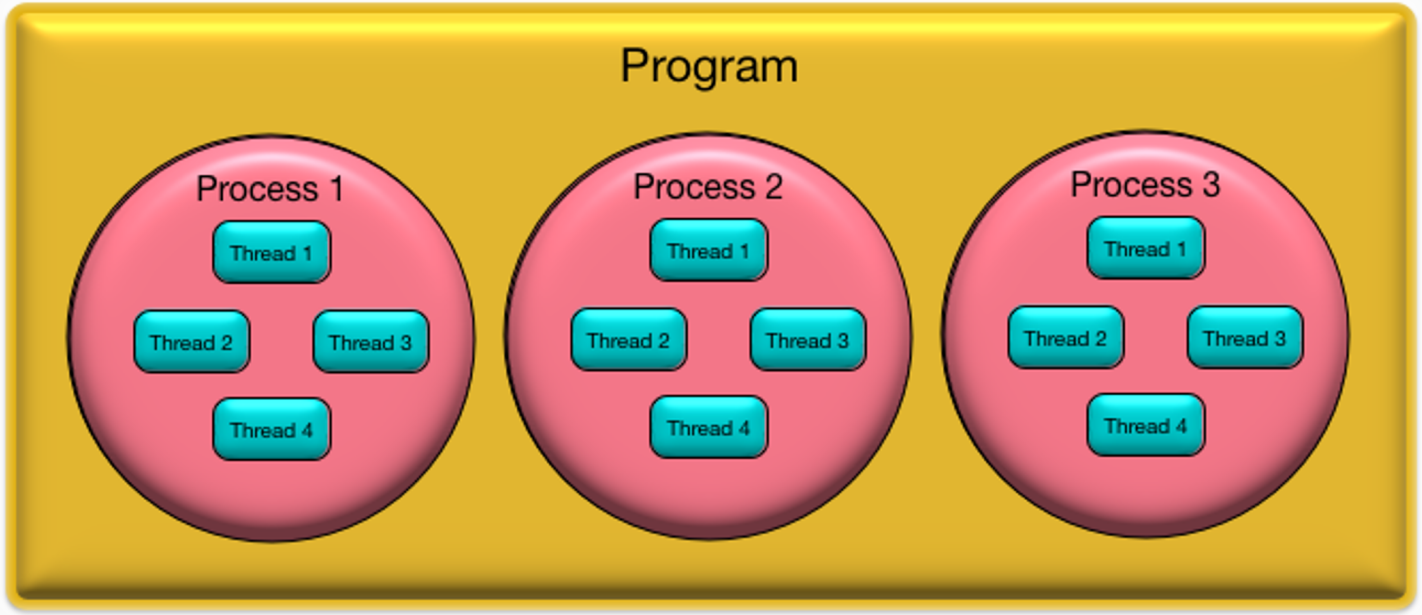
- A subset of a process
- A sequence of execution within a process
- Every single process at least have one thread
- Shares all the resources of process



CYDEO

Multi-threading

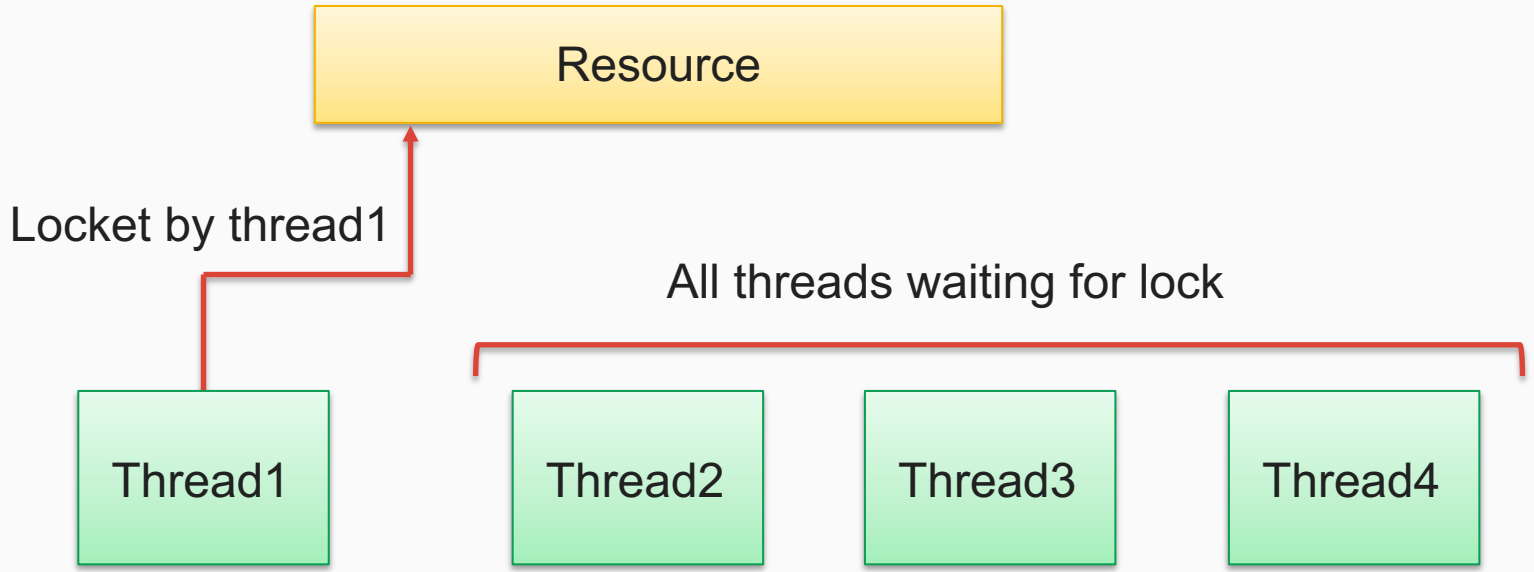
- Two or more threads are being executed within the process



CYDEO

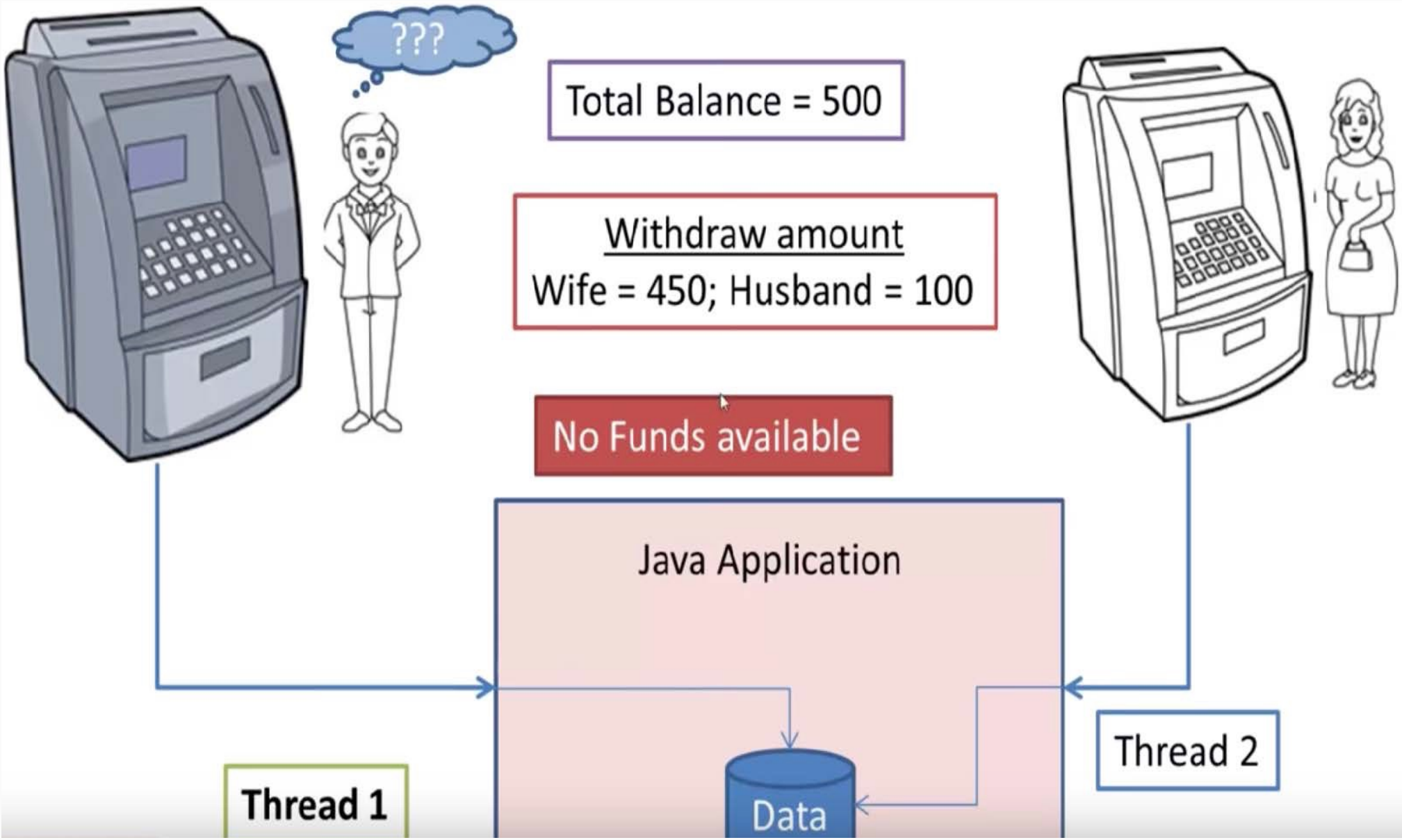
Synchronized (Thread-safety)

- Process of controlling the access of multiple threads to any shared resource
- Only one thread to access the shared resource at a time (Thread-safety)



CYDEO

Synchronized (Thread-Safety)



CYDEO