Programming Foundation

JAVA CLASSES

Java Classes

- Date and Time
- Iterable + Collection Class
  - ArrayList / Vector
  - LinkedList
  - HashMap
  - HashSet
- Exception
- RegEx
- Threads
- Lambda
- File
- Serializer

Date & Time

- LocalDate
- LocalTime
- LocalDateTime
  - now()
  - plus (days, hours, minutes, seconds) depending on the object
  - parse(string) construct an object based on the format
    - LocalDate.parse(“2020-11-11”);
- DateTimeFormatter
  - dd MM (month) yyyy hh mm(min) MMM(month short name) a(AM/
    (Mon, Tue)

Date & Time

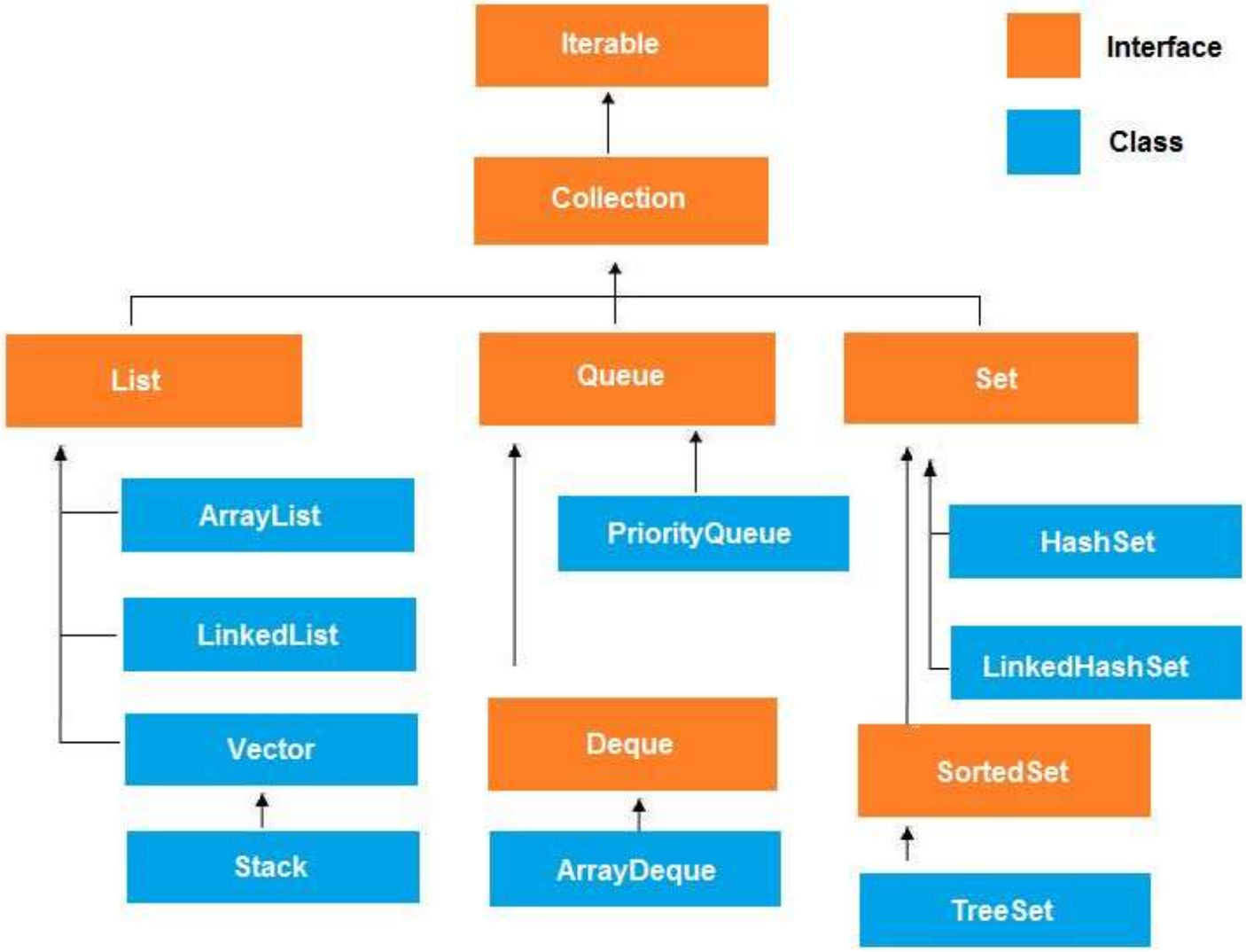
```
LocalDate ldate = LocalDate.now() ;
LocalDateTime ldatetime = LocalDateTime.now();
LocalTime ltime = LocalTime.now();
DateTimeFormatter dtf =
DateTimeFormatter.ofPattern("dd/MM/yyyy") ;
DateTimeFormatter dtf1 =
DateTimeFormatter.ofPattern("E,dd/MMM/yyyy hh:mm a");
System.out.println(ldate);
System.out.println(ldate.format(dtf));
System.out.println(ldatetime.format(dtf1));
System.out.println(ltime);
System.out.println(ldate.plusDays(30));
System.out.println(LocalDate.parse("2020-11-11"));
```

```
2021-03-30
30/03/2021
Tue,30/Mar/2
03:13 pm
15:13:21.081
2021-04-29
2020-11-11
```

Collection Classes

- To learn how to use the collection classes supplied in the library
- To use iterators to traverse collections
- To choose appropriate collections for solving programmin problems
- To study applications of stacks and queues

Collection Class



Collection Class

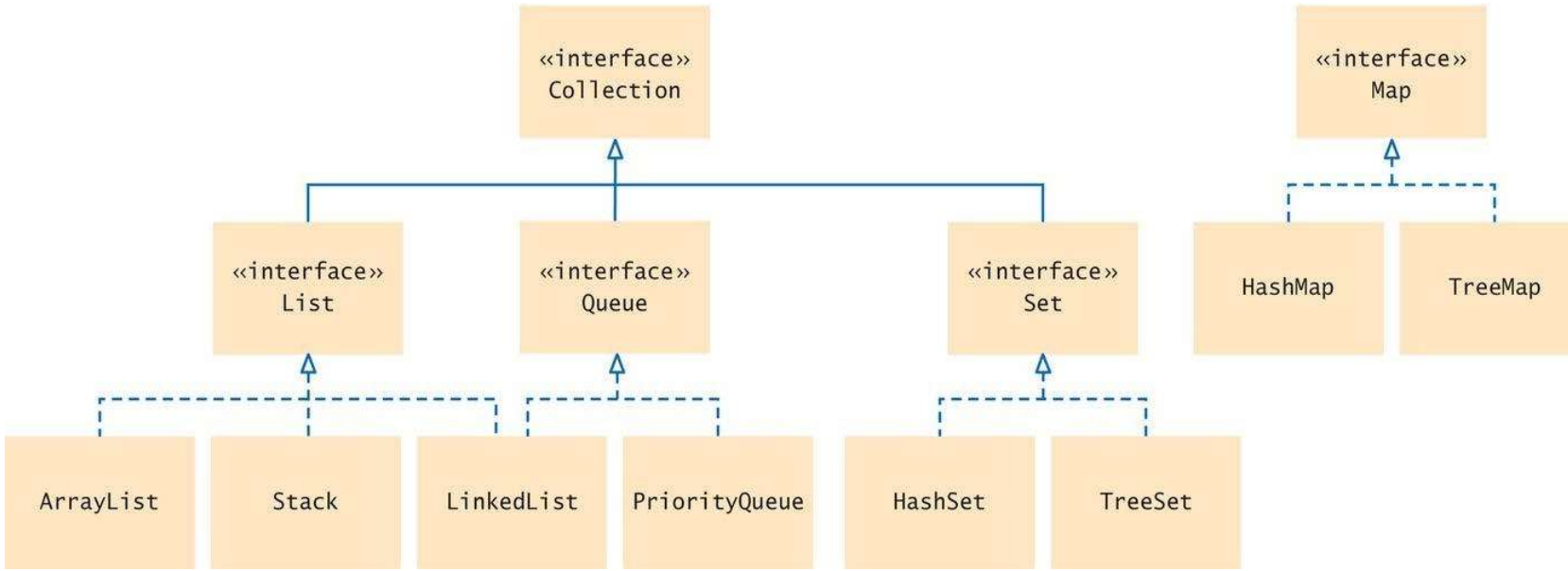
- Java collections framework: a hierarchy of interface types and for collecting objects.
  - Each interface type is implemented by one or more classes
- The Collection interface is at the root
  - All Collection class implement this interface
  - So all have a common set of methods

Iterable

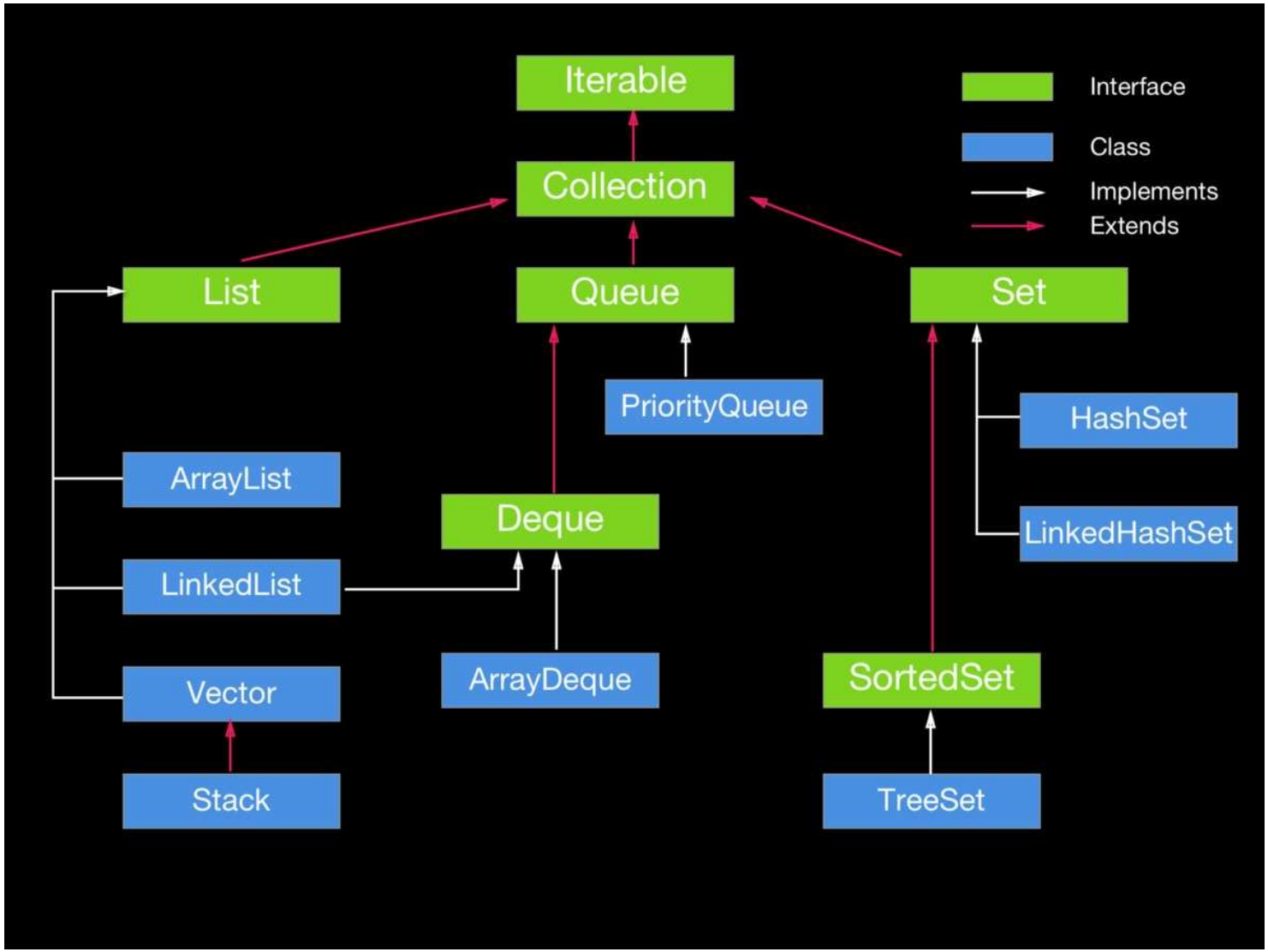
- The *Java Iterable* interface represents a collection of objects w is *iterable* - meaning which can be iterated.
- This means, that a class that implements the Java Iterable inte can have its elements iterated.
- You can iterate the objects of a Java Iterable in three ways: Via for loop, by obtaining a **Java Iterator** from the Iterable, or by c the Java Iterable forEach() method.

Collection

- A collection groups together elements and allows them to be retrieved later.
- Java collections framework: a hierarchy of interface types and classes for collecting objects.
- Each interface type is implemented by one or more classes



Collection



List Interface

- List is a collection which remember the order of its elements
- Two implementing classes
  - ArrayList
  - LinkedList
  - Vector (Legacy)

Set Interface

- A set is an unordered collection of **unique elements**.
- Arranges its elements so that finding, adding, and removing elements is more efficient.
- Two mechanisms to do this
  - hash tables
  - binary search trees

Queue Interface

- Queue
  - Add items to one end (the tail) and remove them from the other end
    head)
  - A queue of people
- A priority queue
  - an unordered collection
  - has an efficient operation for removing the element with the highest
    priority

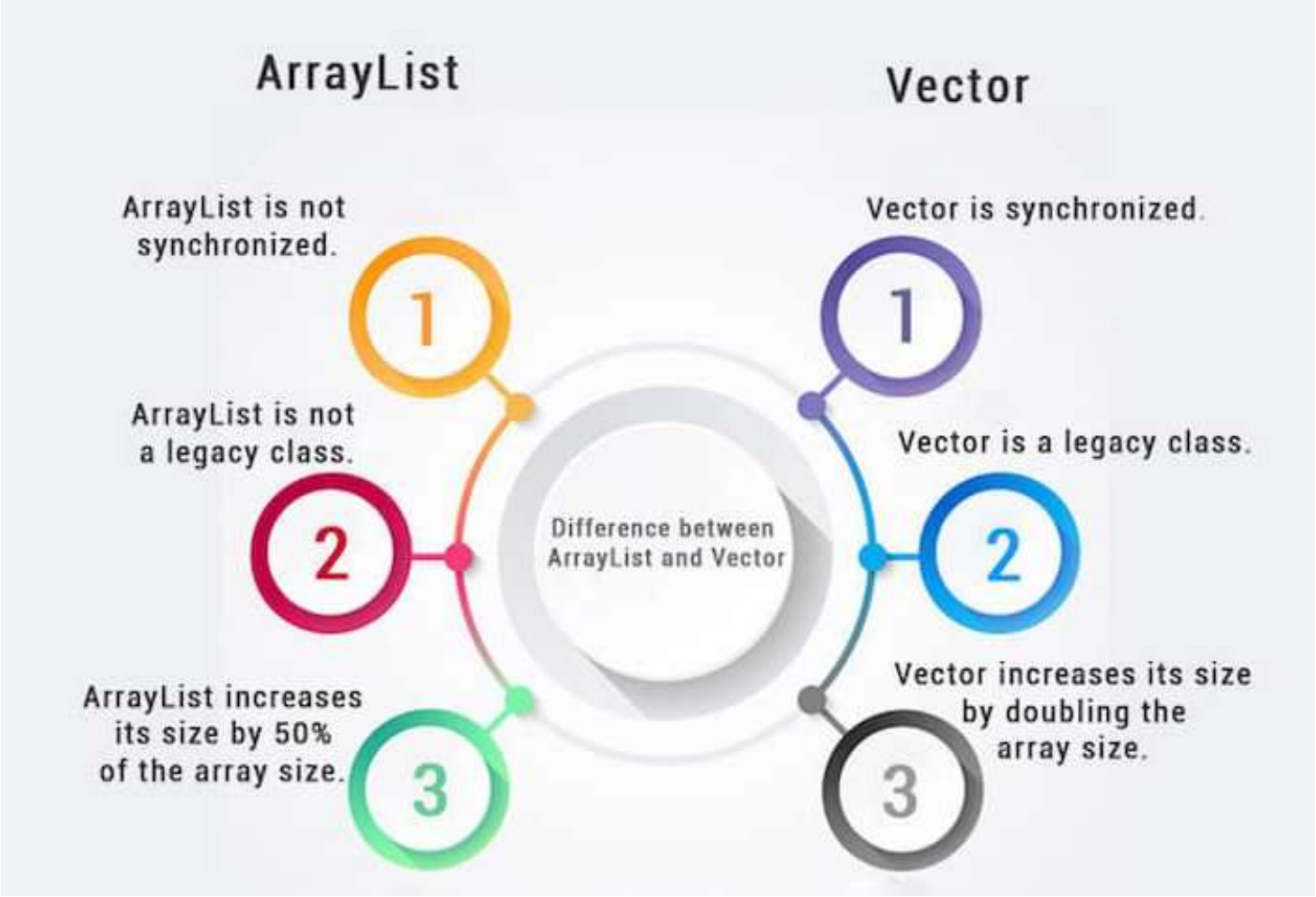
Map Interface

- Map
  - Keeps associations between key and value objects. Every key in the m
    has an associated value.
  - The map stores the keys, values, and the associations between them
  - ISBN to books
  - Have you seen in shops, they scan the barcode to retrieve the inform
    of the product?
- HashMap
- TreeMap

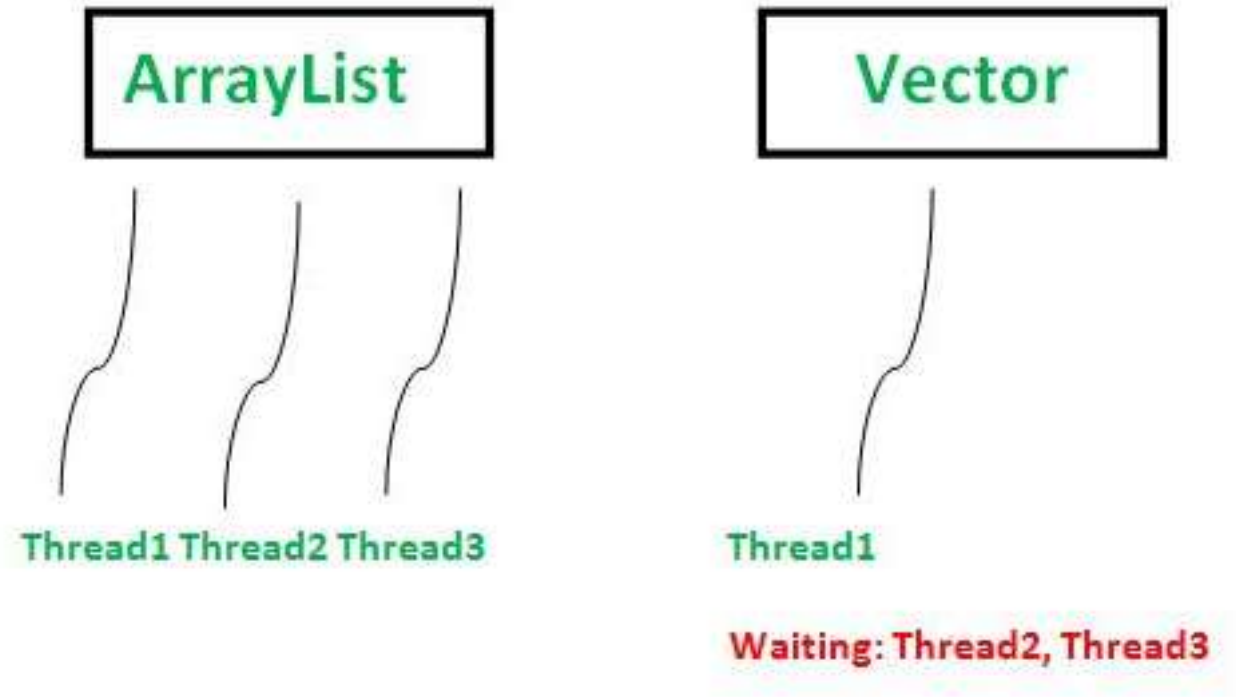
Array List & Vector

- The ArrayList and Vectors are resizable Array
- Found in java.util package
- Array size is not modifiable in java
- Syntax is different
  - String[] cars = new String[120];
  - ArrayList<String> carsList = new ArrayList<Strings>();
  - Vector<Integer> v = new Vector<Integer>();
- Can it be ArrayList or Vectors of Author Class?

Array List & Vector



Array List & Vector



Array List & Vector

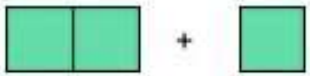
ArrayList vs Vector

ArrayList

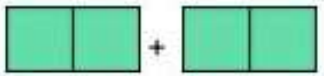
- Asynchronous.
- Not Thread Safe.
- High performance.
- Grows by half of its size.
- Used in single-user application.

Vector

- Synchronous
- Thread Safe
- Slow performance
- Double the size when grow
- Used in multi-user application.



Grows by half of its size



Grows by double of its size

Working with collection

- add and addAll (default or at index)
- remove and removeAll (object or index)
- contains and containsAll (contains)
- indexOf(object)
- get(index)
- forEach(object)
- sort(collection)
- etc..

How do you iterate?

- for loop
- Iterator
- By Indexing
- forEach loop

Example

```
ArrayList<String> carList = new ArrayList<String>();
carList.add("Car1");
carList.add("Car2");
carList.add("Car3");
carList.add("Car4");
```

Example

```
System.out.println("By for (String car: carList)");
for (String car : carList) { // iterator
    System.out.println(car);
}
System.out.println("By indexing - classic array index");
for (int i = 0; i < carList.size(); i++) { // iterator
    System.out.println(carList.get(i));
}
```

Example

```
System.out.println("By Iterator");
Iterator<String> it = carList.iterator();
while (it.hasNext()){
    System.out.println(it.next());
}
System.out.println("By forEach");
carList.forEach( (element) -> {
    System.out.println( element );
});
```

Problems to Solve

1. Write a Java program to create a new array list, add some colors (string) and pri
collection.
2. Write a Java program to iterate through all elements in a array list.
3. Write a Java program to insert an element into the array list at the first position.
4. Write a Java program to retrieve an element (at a specified index) from a given a
5. Write a Java program to update specific array element by given element.
6. Write a Java program to remove the third element from a array list.
7. Write a Java program to search an element in a array list.
8. Write a Java program to sort a given array list.
9. Write a Java program to copy one array list into another.
10. Write a Java program to shuffle elements in a array list.
11. Write a Java program to reverse elements in a array list.

Problems to Solve

12. Write a Java program to extract a portion of a array list.
13. Write a Java program to compare two array lists.
14. Write a Java program of swap two elements in an array list.
15. Write a Java program to join two array lists.
16. Write a Java program to clone an array list to another array list.
17. Write a Java program to empty an array list.
18. Write a Java program to test an array list is empty or not.
19. Write a Java program to trim the capacity of an array list the current list size.
20. Write a Java program to increase the size of an array list.
21. Write a Java program to replace the second element of a ArrayList with the specified el
22. Write a Java program to print all the elements of a ArrayList using the position of the el