Data Structure

What is a Data Structure?

A data structure is a way to organize data in a computer so that it can be used effectively. To measure how effective a data structure is, we can look at the time it takes to perform operations on it this is called time complexity, and the space it takes to store the data this is called space complexity.

A part of the java standard library is the collection framework, this is a set of interfaces and classes that implement common data structures.

Arrays are also a part of the Java standard library. In Java, arrays are considered fundamental data structures and are supported directly by the language itself. They are not considered separate libraries but are included as a built-in feature.

Java arrays provide a way to store multiple values of the same type in a contiguous block of memory. Arrays in Java have a fixed length, meaning that once created, their size cannot be changed.

Arrays

Array - This is a Sequence of things, and has a fixed size. Faster then linkedlist and arraylist because you can access the items from memory directly. The index starts from 0. This is a continues piece of memory.

When you declare an array, you are creating a reference variable that can point to an array object in memory. The type of this reference variable is always an array type. The array type is derived from the element type by adding brackets ([]).

```java
public class LetsArray{
  //The are two ways you could initialize and array

  // 1)
  int arr [] = new int [10] //Max items are 10
  arr[0] = 1;

  //2)
  int arr[] = {1,2,4,5}
}
```

In java an array can only be one type but all the types have a wrapper class, an array of objects can be made and you can store any data type. This is a bad practice but it still can be done.

```java
public class LetsArrayObj{
    Object arr [] = new Object [10];
    arr[0] = 5;
    arr[1] = "Hello";
    arr[2] = 6.4;
}
```

You can also have a multidimensional array, this is an array of arrays. This is a 2D array. Multidimensional arrays are not limited to two indexes, you can have as many as you want.

This is a data structure that organizes elements in multiple dimensions. Unlike one-dimensional arrays that store elements in a linear sequence, multidimensional arrays allow for storing data in a tabular or grid-like format with rows and columns, and additional dimensions beyond that.

```
public class LetsArray2D{
    int arr [][] = new int [10][10];
    arr[0][0] = 1;
    arr[0][1] = 2;
    arr[1][0] = 3;
    arr[1][1] = 4;
}
```

Generic Types

What is a Generic Type?

A generic type is a generic class or interface that is parameterized over types. These are used to create classes that can work with any data type. This allows you to reuse the same code with different inputs.

If a generic type has multiple ways of doing something they usually have a interface that they implement.

Using the most generic type like `List<Integer> arr = new ArrayList<>()` instead of `ArrayList<Integer> arr = new ArrayList<>()` is generally considered a good practice because it promotes coding to interfaces rather than implementations. This approach provides several benefits:

1. Flexibility: Declaring the variable as `List<Integer>` allows you to switch the implementation to any other class that implements the `List` interface without needing to change the rest of the code. For example, you can easily change it to `LinkedList<Integer>` if the requirements of your code change.

2. Polymorphism: Declaring the variable as `List<Integer>` allows you to treat it as a more general type, enabling you to write more generic and reusable code. You can pass the variable to methods that accept `List<Integer>` without being tightly coupled to a specific implementation.

```
public class LetsGeneric{
  public static void main(String[] args) {
    //This is a generic type
    List <Integer> list = new ArrayList<>();

    list.add(1);

    list.get(0);

```

```
      list.add(0 , 5) // Index 0 store the value 5
    }
  }
```