Collections:

1. Narmalloy to store the data what we use?

variables

1. But if the data has large values then it is difficult to create so many variables. For this large or so many variables what we will use?

Array

1. Brief things about arrays:
   1. What is array?

An array in Java is a collection of similar data types stored in contiguous memory locations. It allows you to store multiple values in a single variable, which can be accessed and manipulated using an index.

* 1. How can an array declared?

dataType[] arrayName;

or

dataType arrayName[];

For example, the following code declares an array numbers of type int:

int[] numbers;

* 1. How do you initialize an array, you can use the following syntax:

arrayName = new dataType[arraySize];

For example, the following code initializes

the numbers array with a size of 5:

numbers = new int[5];

You can also initialise an array with values at the time of declaration, like this:

int[] numbers = {1, 2, 3, 4, 5};

Arrays are used in many applications where you need to store and manipulate large amounts of data. They are especially useful when you need to store data of the same type, and when you know how many items you need to store ahead of time.

1. Can you please explain what are the disadvantages of arrays

Arrays have the following disadvantages in Java:

1. Fixed size: Once an array is created, its size cannot be changed. If you need to add or remove elements, you have to create a new array and copy the elements from the old array to the new array.
2. Inefficient for inserting or deleting elements in the middle: Inserting or deleting elements in the middle of an array is not efficient as it requires shifting all elements to accommodate the change.
3. Limited functionality: Arrays do not provide many built-in methods for searching, sorting, or manipulating elements, so you have to write your own algorithms or use other data structures for these tasks.
4. Homogeneous data only: Arrays can store only homogeneous elements, i.e., elements of the same type. If you need to store elements of different types, you have to use an array of objects.
5. No built-in data structure to store key-value pairs: Arrays do not provide a built-in data structure for storing key-value pairs, so you have to use other data structures such as Maps for these tasks.

Overall, arrays are useful for simple tasks, but for more complex data storage and manipulation, other data structures like ArrayLists, LinkedLists, HashMaps, etc., are usually more appropriate.

6. From which version collection api’s added in java?

Ans : from java 1.2 version onwards - joshua created collections and suggested to the sunmicrosystems to add in java

7. what are the classes in collections used in java

Java Collections Framework provides several classes for storing and manipulating collections of data. Some of the commonly used classes in the Java Collections Framework are:

**ArrayList:** A resizable array implementation of the List interface that allows us to store and retrieve elements in a dynamic array.

ArrayList is a class in Java that implements the List interface and provides an array-backed implementation of the List interface. The ArrayList class has several built-in methods that can be used to manipulate the elements of the list. Some of the most commonly used methods are:

1. add(E e) - Adds the specified element to the end of the list.
2. add(int index, E element) - Inserts the specified element at the specified position in the list.
3. clear() - Removes all elements from the list.
4. contains(Object o) - Returns true if the list contains the specified element.
5. get(int index) - Returns the element at the specified position in the list.
6. indexOf(Object o) - Returns the index of the first occurrence of the specified element in the list.
7. isEmpty() - Returns true if the list is empty, false otherwise.
8. remove(int index) - Removes the element at the specified position in the list.
9. set(int index, E element) - Replaces the element at the specified position in the list with the specified element.
10. size() - Returns the number of elements in the list.

These are just a few of the methods available in the ArrayList class. There are several others that you can use to perform operations on the list.

| import java.util.\*;  public class First {  public static void main(String[] args) {  ArrayList al1 = new ArrayList();  al1.add(10);  al1.add(20);  al1.add(30);  System.*out*.println(al1);  // \*\*with this we can conclude that in arraylist we can store the homogeneous data  // \*\*remember that in collection if you add any data that will be store as object.  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*");  ArrayList al2 = new ArrayList();  al2.add("ineuron");  al2.add(20);  al2.add('b');  al2.add(10.5);  System.*out*.println(al2);  // with this we can conclude that in arrylist we can store the hetroginious data  ArrayList al3 = new ArrayList();  al3.addAll(al2);  System.*out*.println("vlaue of al3" + al3);  // with this we can confirm that we can add entaire collection into existing collection or new collection.  ArrayList al4 = new ArrayList(); // ArryList is the class in the colecgtions  al4.add(11);  al4.add(12);  al4.add(13);  al4.add(14);  System.*out*.println("value of al4" + al4);  al4.add(2, 28); // means we are saying at 2nd index add 28  System.*out*.println("after adding in secodn index" + al4); // // but not recomended because it will shift the object to next indenx  al4.add(0,5);  System.*out*.println("after adding in the 0 index" + al4); // but not recomended  al4.add(55);  System.*out*.println("after adding at the rare end " + al4); // adding the data at rate end is recomended.  }  }  —-----------------------  [10, 20, 30]  \*\*\*\*\*\*\*\*\*\*\*\*  [ineuron, 20, b, 10.5]  vlaue of al3[ineuron, 20, b, 10.5]  value of al4[11, 12, 13, 14]  after adding in secodn index[11, 12, 28, 13, 14]  after adding in the 0 index[5, 11, 12, 28, 13, 14]  after adding at the rare end [5, 11, 12, 28, 13, 14, 55] |
| --- |

LinkedList: A doubly-linked list implementation of the List interface that allows us to store and retrieve elements in a linked list.

what are the methods in linked list in java?

The Java LinkedList class implements the List interface and is backed by a doubly-linked list data structure. Some of the commonly used methods in LinkedList are:

1. add(E element) - Adds the specified element to the end of the list.
2. add(int index, E element) - Inserts the specified element at the specified position in the list.
3. clear() - Removes all elements from the list.
4. contains(Object o) - Returns true if the list contains the specified element.
5. get(int index) - Returns the element at the specified position in the list.
6. indexOf(Object o) - Returns the index of the first occurrence of the specified element in the list.
7. isEmpty() - Returns true if the list is empty.
8. remove(int index) - Removes the element at the specified position in the list.
9. set(int index, E element) - Replaces the element at the specified position in the list with the specified element.
10. size() - Returns the number of elements in the list.

These are just a few examples, there are many more methods available in LinkedList to perform various operations.

| import java.util.\*;  public class Doublylinnked {  public static void main(String[] args) {  LinkedList ll1 = new LinkedList();  ll1.add(10);  ll1.add("ineuron");  ll1.add(20);  ll1.add(20);  System.*out*.println(ll1);  ll1.addFirst("ashraf");  System.*out*.println("afteradding with addfirst method" + ll1);  ll1.add(3, "bangaloure");  System.*out*.println("after adding the 3rd insdex"+ ll1);  ll1.addLast("wonder");  System.*out*.println("after adding wonder" + ll1);  }  }  Ans:  [10, ineuron, 20, 20]  afteradding with addfirst method[ashraf, 10, ineuron, 20, 20]  after adding the 3rd insdex[ashraf, 10, ineuron, bangaloure, 20, 20]  after adding wonder[ashraf, 10, ineuron, bangaloure, 20, 20, wonder] |
| --- |

* Internally Linkedlist datastructer will fallows the Doubleylinked data structure.

HashMap: A class that implements the Map interface, allowing us to store and retrieve key-value pairs.

HashSet: A class that implements the Set interface, allowing us to store unique elements in a hash table.

TreeMap: A class that implements the Map interface and uses a tree data structure for storing key-value pairs.

TreeSet: A class that implements the Set interface and uses a tree data structure for storing elements.

Queue: A collection that implements the Queue interface and provides functionality for adding elements at the end of the queue and removing elements from the front of the queue.

Deque: A collection that implements the Deque interface and provides functionality for adding elements at the front and end of the queue and removing elements from the front and end of the queue.

PriorityQueue: A collection that implements the Queue interface and provides functionality for adding elements in order based on priority and removing elements from the front of the queue.

These are just a few of the classes in the Java Collections Framework. There are many other classes available to suit different data storage and manipulation requirements.

All this are will be placed in the util package, if you want to use them you have to import them.

When to use array over array list?

When ever size of the data is known to you and if you are sure data is homogeneous or similar type then you have to with arry only. Because array is faster then arrylist.