1. **How does dataset work, Alternative for Dataset.**

A dataset is a collection of data that is organized in a specific way for a particular purpose. Datasets are used in many different fields such as scientific research, machine learning, and business analytics etc. it is a collection of data tables that contains the relational data in table structure.

Dataset is the local copy of the database which exists in the local system and makes the application execute faster and reliable.

Types of Datasets are:

1. Numerical Dataset
2. Categorical Dataset
3. Bivariate Dataset
4. Multivariate Dataset
5. Correlation Dataset

* **Alternatives for Datasets**

**Data Streams**: A data stream is a continuous flow of data that is generated in real-time, rather than being collected and stored in a dataset. Data streams are often used in different fields like finance, social media, and internet of things (IoT), where the data is constantly changing and needs to be analyzed in real-time.

**Data Lakes**: A data lake is a centralized repository that stores structured, semi-structured, and unstructured data. a data lake does not require data to be structured or cleaned prior to storage, allowing for more flexibility in analysis.

**Data APIs**: A data API (Application Programming Interface) allows for data to be accessed and retrieved in real-time from an external source, rather than being stored locally in a dataset. APIs are often used in web development, mobile apps, and other software applications.

**Data Simulations**: Data simulations involve generating synthetic data to model real-world scenarios. This approach can be useful in situations where collecting real-world data is difficult or expensive.

**Data Scraping**: Data scraping involves using software to extract data from websites or other online sources. This approach can be useful for collecting data that is not available in a structured dataset, or for collecting data in real-time.

**2.How datasets work.**

The working of a dataset depends on the purpose for what it is used. In general, a dataset can be broken down into several key components:

**Data Collection**: Data is collected from various sources, such as surveys, sensors, public databases, or web scraping.

**Data Preprocessing**: Once the data is collected, it needs to be cleaned, transformed, and structured in a way that is suitable for analysis. This process is known as data preprocessing.

**Data Exploration**: After the data is cleaned and structured, it is explored and analyzed to gain knowledge and identify patterns. This process is known as data exploration.

**Data Modeling**: Once the data is explored. it is used to create the model according to the purpose of dataset.

**Model Evaluation**: The model is evaluated to determine its accuracy and effectiveness. This process involves using test data that is separate from the data used for modeling and training.

**Model Deployment**: Once the model is evaluated, it can be deployed and used for its intended purpose.

A dataset is a collection of structured or unstructured data that is organized in a specific way for a particular purpose. The purpose of a dataset could be anything from scientific research to machine learning or business analytics. Datasets are typically created by collecting data from various sources, such as surveys, sensors, public databases, or web scraping.

Once the data is collected, it needs to be cleaned, transformed, and structured in a way that is needed for analysis. After that data is cleaned and structured, it is explored and analyzed to gain knowledge and identify patterns which is known as data exploration.

Once the data is explored, it can be modeled to create a model, depending on the purpose of the dataset. The model is evaluated against some test cases to determine its accuracy and effectiveness using test data that is separate from the data used for modeling. Once the model is evaluated and looks effective, it can be deployed and used for its intended purpose.

It is important to ensure that the dataset is accurate, representative, and unbiased. The working of a dataset is iterative and ongoing, with the dataset being updated and refined as new data becomes available.

**3.How does Array work with Memory**

an array is a data structure that is used to store a collection of elements of the same type. It is the collection of Homogenous data elements. The elements of an array are stored in contiguous memory locations, which means that the memory for the array is allocated as a single block.

When an array is created, a block of memory is reserved for the array, and each element of the array is assigned a unique memory address within that block. The memory address of the first element of the array is used as a reference point for accessing the other elements of the array. This means that elements in the array can be accessed easily, as the memory addresses of the elements are known and can be used to retrieve their values.

Arrays can be statically or dynamically allocated.

**Statistically Allocated Array**:

Statically allocated arrays are allocated at compile time, and the size of the array is fixed and cannot be changed during runtime.

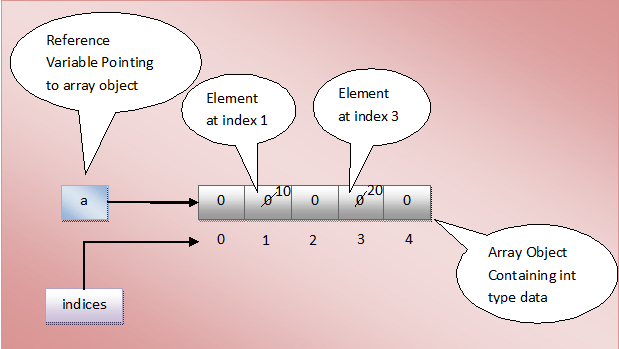
**Dynamically Allocated Array:**

Dynamically allocated arrays are allocated at runtime, and the size of the array can be changed dynamically during program execution.

When an array of a given size, say N, and of a given type is declared, the compiler allocates enough memory to hold all N pieces of data. If M bytes of memory is required for each piece of data of the type specified, then a total of N\*M bytes of contiguous memory are allocated to that array.

Arrays can be multidimensional, which means that they can have more than one index. In this case, the memory for the multidimensional array is allocated as a block of contiguous memory locations, and the elements are accessed using multiple indices. The array indices start from 0.

This allows for efficient access to the elements of the array by their memory addresses.



**4.How does ArrayList work with Memory**

* an ArrayList is a dynamic data structure that is used to store a collection of elements of the same type. an array has a fixed size and cannot be modified, an ArrayList can grow or shrink dynamically during runtime. Elements in ArrayList can be added and removed whenever we want.
* ArrayLists uses contiguous memory. All elements in the ArrayList are located next to each other in the same memory space.
* ArrayList uses an array to store its elements. When an ArrayList is created, an initial array is created with a certain capacity, and elements are added to the array as needed. If the capacity of the array is exceeded, a new, larger array is created, and the elements from the old array are copied into the new array.
* When elements are added to an ArrayList, they are appended at the end of the array. If the ArrayList needs to be resized, a new, larger array is created, and the elements from the old array are copied into the new array. This process of resizing the array can be expensive in terms of memory and processing time, especially if the ArrayList is large.
* In terms of memory usage, an ArrayList uses more memory than a regular array, as it needs to allocate additional memory for the internal array when it needs to be resized. It is flexible and it is very useful tool in programming.

There are different methods in java through which we can access, modify, add , delete the elements in the ArrayList.

**Get ()** – used to access the element.

**Set ()** – used to modify the element.

**Remove ()** – used to remove a particular element from the ArrayList.

**Add ()** – used to add the elements in the ArrayList.

**Clear ()** – used to remove all the elements from the ArrayList.

**Size ()** – it returns the size of the ArrayList.

**5.Array VS ArrayList**

Arrays and ArrayLists are both used to store collections of elements of the same type.

The key differences between Array and ArrayList:

**Size**: Arrays have a fixed size that is determined when they are created, whereas ArrayLists are of variable Length.

**Type**: Arrays are static i.e. it cannot be modified , but ArrayList are dynamic . it means they can be changed dynamically when needed.

**Memory**: Arrays use contiguous memory locations to store their elements, while ArrayLists use internal array that may need to be resized during runtime, potentially using more memory.

**Performance**: Accessing elements in an array is generally faster than accessing elements in an ArrayList because arrays use contiguous memory locations. However, ArrayLists provide more flexibility in terms of size and are easier to work with in many situations.

**Type safety**: Arrays are type-safe, meaning that the type of the elements is checked at compile-time. ArrayLists are not type-safe, meaning that errors may only be detected at runtime.

**Functionality**: Arrays provide limited functionality, such as sorting and searching, whereas ArrayLists provide additional functionality, such as the ability to easily add or remove elements, and to resize dynamically.

**Dimensions:** Arrays are single and Multidimensional whereas ArrayList are single dimensional Only.

**6.Understanding about memory working with Array and collection objects**

**Array:** It is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index.

Arrays can be allocated statically and dynamically.

**Collection objects**: it is an ordered set of items that can be referred to as a single unit.

* Arrays store data in contiguous memory locations, which means that the memory for the array is allocated as a single block. Each element of the array is assigned a unique memory address within that block. Accessing an element in an array is fast and efficient because the memory address of the element is known and can be used to retrieve its value.
* Collection objects can use more complex data structures to store data, such as linked lists or trees. These data structures use more memory than arrays because they require additional memory to store pointers to the next or previous elements in the list or tree. However, collection objects provide more flexibility and functionality than arrays because they can grow or shrink dynamically and provide different methods to manipulate the data.
* When working with arrays or collection objects, it is important to consider the size of the data being stored and how it will impact memory usage. Large arrays or collections can use a significant amount of memory, which can impact the performance of the program. It is also important to consider the efficiency of data access and manipulation, as inefficient algorithms can result in slow program performance.
* Careful consideration of memory usage and efficiency is important when working with arrays and collection objects to ensure that programs are performing optimally.

**7.How does Garbage collector work with Array/ArrayList/Dictonary and how is memory freed of an object and what is the process**

**Garbage collection** is a process in programming languages like c#, java where unused objects and memory are automatically reclaimed by the system. Garbage collectors work by identifying objects in memory that are no longer being used by the program and freeing up the memory they occupy.

In the case of arrays, ArrayLists, and Dictionaries, the garbage collector works by monitoring the objects that are referenced by the array or collection. When an array or collection is no longer being used by the program, the garbage collector will identify the objects it references and mark them as eligible for garbage collection.

Once an object is marked as eligible for garbage collection, the garbage collector will free up the memory it occupies by removing it from memory. The process of garbage collection can be triggered automatically by the system when memory usage reaches a certain threshold or can be triggered manually by the programmer.

The garbage collector is an important component of modern programming languages because it helps to prevent memory leaks and reduces the risk of running out of memory. Garbage collection allows programs to allocate and deallocate memory dynamically, which can make them more flexible and efficient.

the garbage collector works by monitoring the objects that are referenced by arrays, ArrayLists, and Dictionaries. When an array or collection is no longer being used by the program, the garbage collector will identify the objects it references and mark them as eligible for garbage collection. Once an object is marked as eligible for garbage collection, the garbage collector will free up the memory it occupies by removing it from memory.

Basically It implements Mark and Sweep algorithm

**Memory freeing of an object and its process**

In most modern programming languages, including Java, Python, and C++, memory is managed by a mechanism called "garbage collection".

The process of freeing memory of an object typically involves the following steps:

**Marking**: The garbage collector identifies objects that are no longer being used by the program, i.e., objects that are no longer reachable through any reference variable.

**Deallocating**: The garbage collector deallocates the memory that was used by the identified objects.

The exact process of garbage collection can vary depending on the language and the implementation of the garbage collector. In some cases, garbage collection may occur automatically in the background, while in other cases, it may be triggered manually or on a scheduled basis.

**8. Connection of Garbage Collector with Threading process**.

* Garbage collection and threading are two important aspects of memory management in programming. In some programming languages, garbage collection is performed automatically by the runtime environment to free up memory that is no longer being used by the program. Threading allows programs to execute multiple threads of execution concurrently.
* Garbage collection in java is carried out by a daemon thread called Garbage Collector.
* Daemon threads are low priority threads which performs their tasks in background such as garbage collection.
* Before removing an object from memory garbage collection thread invokes finalize() method of that object and gives a warning to perform any sort of clean up required.
* Garbage Collection can be done manually and automatically.
* In the case of automatic GC , it will only trigger if JVM thinks it needs garbage collection based on java heap size.

If we talk about garbage collection in java then JVM provides four different garbage collectors

* **Serial GC:** This is the simplest garbage collector, designed for single threaded systems and small heap size.
* **Parallel/Throughput GC:** it uses multiple threads to scan through the heap space and perform compaction. Its drawback is that it pauses all the application threads while it is performing its task.
* **The CMS collector:** Concurrent Mark Sweep uses multiple threads to scan through the heap for unused objects that can be recycled.
* **G1 collector:** Garbage first collector is designed for heap size greater then 4GB. It divides the heap size and perform marking operation to determine the active objects and then it collects unreachable objects first from the empty region.

In a multithreaded environment, the garbage collector must be designed to handle concurrent access to memory by multiple threads. This can be a challenging task.

To address this issue, modern garbage collectors use a technique called "concurrent garbage collection" or "concurrent marking". In this technique, the garbage collector runs concurrently with the program's threads, periodically interrupting them to identify the objects that are no longer being used and can be garbage collected. By doing this work concurrently with the program's threads, the garbage collector minimizes the impact on program performance and ensures that the program's memory usage remains within reasonable bounds.

**9. How garbage collector works with multiple threads**

In a multi-threaded environment, the garbage collector (GC) works with multiple threads by ensuring that it collects garbage that is no longer being used by any thread. This is done by employing various techniques and algorithms to manage memory, identify and track objects that are no longer in use, and recover their memory.

One common approach used by GCs in a multi-threaded environment is to use **a mark-and-sweep** algorithm. In this algorithm, the garbage collector first marks all live objects in memory. This process is typically performed concurrently with the execution of the program's threads, so that the GC can keep track of which objects are in use at any given time.

Once all the live objects have been marked, the garbage collector **sweeps** through memory, reclaiming the memory occupied by any objects that are not marked. This process is also typically performed concurrently with the program's threads, so that the GC can minimize any impact on the program's performance.

In addition to mark-and-sweep, there are other algorithms used by garbage collectors that are better suited for multi-threaded environments. For example, the **concurrent copying algorithm** is designed to work efficiently with multiple threads by dividing the memory into regions and allowing multiple threads to work on different regions simultaneously.

the garbage collector in a multi-threaded environment needs to be carefully designed and optimized to ensure that it can manage memory effectively without unduly impacting the performance of the program's threads.

**10.  what do you think why Time and space complexity is important explain it with an own example**

Time and space complexity are two fundamental concepts in computer science that are used to evaluate the efficiency of algorithms.

**Time complexity** refers to the amount of time an algorithm takes to run as a function of the input size.

**space complexity** refers to the amount of memory an algorithm requires as a function of the input size.

Understanding time and space complexity is important because it helps us to choose the most efficient algorithm for a given problem.

Time and Space complexities are represented with the help of Asymptotic notations.

**There are three asymptotic notations:**

* **Big-O Notation**
* **Omega Notation**
* **Theta Notation**

Using these notations, the efficiency of the algorithm is represented in different scenarios or cases like Worst, Average and Best Cases.

**There are six major types of complexities:**

* **Constant:** O (1) 🡪Excellent/Best
* **Linear Time:** O(n) 🡪Fair
* **Logarithmic time:** O (n log n) 🡪Bad
* **Quadratic time:** O(n^2)🡪Horrible/Worst
* **Exponential time:** O(2^n) 🡪 Horrible/Worst
* **Factorial time:** O(n!) 🡪 Horrible/Worst

Example: sum of n numbers

|  |  |  |
| --- | --- | --- |
| **statements** | **Frequency** | **Total steps** |
| Algorithm Sum( arr , n)  {Tempsum = 0;  For(i:=1 to n do)  Tempsum+= arr[i];  Return tempsum;  } | ----  1  N+1  N  1  ---- | 0  1  N+1  N  1  0 |
|  | **Total** | 2n+3 |

Time complexity : O(n)