# NON PREMITIVE DATA TYPE IN JAVA

## 1. What are non-primitive data types in Java?

## Answer.

- Non-primitive data types are complex data structures built upon primitive data types. They provide more functionality and flexibility than the basic types.

## 2. List some common non-primitive data types in Java.

## Answer.

- Arrays: Collections of elements of the same data type, accessed with an index.
- Strings: Sequences of characters, immutable and stored in the String pool.
- Classes: Blueprints for creating objects with attributes and methods.
- Interfaces: Define contracts for classes to implement, promoting abstraction.
- Enums: Represent a fixed set of named constants, enhancing readability and type safety.

## 3. Describe the difference between arrays and ArrayLists.

## Answer.

- Arrays are fixed-size, while ArrayLists are dynamic and can grow/shrink.
- Arrays offer better performance for basic operations, while ArrayLists are more flexible.

## 4. How do you create and access elements in an array?

## Answer.

- Declare an array with size and data type, then use index (starting from 0) to access elements.

## 5. Explain how Strings are immutable in Java.

## Answer.

- Values cannot be changed directly. Modifying a String creates a new object.

## 6. What are the benefits of using classes and objects?

## Answer.

- Encapsulation: Bundling data and behavior together.
- Reusability: Creating multiple instances with the same blueprint.
- Inheritance: Creating specialized classes based on existing ones.

## 7. What is the purpose of interfaces in Java?

## Answer.

- Define common functionalities that different classes can implement.
- Promote loose coupling and polymorphism.

## 8. How do Enums simplify code and improve readability?

## Answer.

- Represent a set of pre-defined constants with explicit names, avoiding magic numbers.
- Enhance type safety and provide compile-time checks.

## 9. Discuss the difference between abstract classes and interfaces.

## Answer.

- Abstract classes can have both abstract and concrete methods, while interfaces only have abstract methods.

- Abstract classes can implement some behavior, while interfaces define pure contracts.

## 10. What are some advanced non-primitive data types in Java?

## Answer

- Collections framework (e.g., HashMap, LinkedHashSet) for complex data structures and algorithms.
- Annotations for meta-data and code behavior extension.
- Generics for creating type-safe and reusable data structures and algorithms.

By mastering these non-primitive data types, you'll unlock powerful tools for structuring your Java programs and creating efficient, maintainable, and readable code.