Short Notes

1) String is immutable in Java.
A String is an immutable object which means we cannot change them after creating the objects. Whenever we change any String, a new instance is created. ~~In~~ The term "immutable string" in Java refers to a string object that cannot be altered, but the reference to the object ~~to~~ can be changed.

2) Array and ArrayList
Array:
- Fixed size
- Declared with Square brackets [ ]
- Elements are of the same data type.
- Memory allocation is static
- Faster access and manipulation
- No built-in methods for adding/removing elements.

ArrayList:
- Dynamic size
- Implemented as a class java.util.ArrayList
- Elements can be of different data types (Object)
- Memory allocation is dynamic
- slower access and manipulation compared to arrays
- Built-in methods for adding/removing elements

3) The need of overriding equals and Hashcode methods as a standared context of collection Framework.
overriding the equals() and hashcode() methods in Java is important for defining object equality and ensuring the correct behavior of collections. This is especially true when working with custom data types

and collections. You must override hashcode() in every class that overrides equals(). Failure to do so will result in a violation of the general contract for object. Hashcode(), which will prevent your class from functioning properly in conjunction with all hash-based collections, including HashMap, Hashset, and Hashtable.

Why overriding equals() and hashcode()?
1. Uniqueness : Ensure unique elements in collection
2. Lookup : Efficiently retrieve elements from collections.
3. Removal : Correctly remove elements from collections.

collection framework
1. Equals contract : a.equals(b) implies a.hashcode() == b.hashcode()
2. HashCode contract : If a.hashcode() == b.hashcode(), a.equals(b) may or may not be true.

equals()
• check null and getclass()
• compare relevant Fields
• Use Objects.equals() for simplicity

Hashcode()
• Use prime numbers to minimize collisions
• combine hashcodes of relevant Fields
• Use ~~Objects.equals() for~~ Object.hash() For simplicity.

4) Database Connectivity in Java.
Steps to connect to a database in Java.
1. Import necessary libraries : import java.sql package and database-specific driver

2. Register the driver : Register the database driver using ~~class.Nor~~ class.forName() or the Driver Manager class.

3. Establish a connection : Create a Connection object using DriverManager.getConnection().

4. Create a Statement : Create a statement object to execute SQL queries.

5. Execute queries : Execute SQL queries using Statement. executeQuery() or Statement.executeUpdate();

6. Process results : Process the ~~raw~~ result using Resultset object

7. close Resource : close the statement, connections and Resultset objects.

Database Connectivity API's in Java
1. JDBC (Java Database Connectivity) : Built-in API for interacting with relational database.

11. JPA (Java Persistence API) : Standard API for object-relational mapping. (ORM):

111. Hibernate : Popular ORM framework.

5) Interfaces and types of interfaces and advanced Feature of interfaces.
Interfaces in java define a contract or a set of methods that must be implemented by any class that implements it.

Types of Interfaces :
1. Normal Interfaces : Defines methods and constants
2. Marker Interfaces : Empty interface used for tagging or marking classes.

Page No.:
Date: YOUVA

3. Functional Interface : single abstract method (SAM) interface, used for lambda expressions
4. Nested Interface : Defined inside another interface or class.
5. Inner Interface : Defined inside a class, but not static.

Advanced Features :
1. Default Methods : Introduced in Java 8, allows interfaces to provide default implementations.
2. Static Methods : Introduced in Java 8, allows interfaces to have static methods.
3. Private Methods : Introduced in Java 9, allows interfaces to have private methods.
4. Functional Programming : Interfaces can be used as target types for lambda expression.
5. Generics : Interfaces can be generic, allowing type parameters.

6) MySQL Data Types : char, varchar and Nvarchar
• char : Fixed-length character string
• varchar : variable-length character string
• Nvarchar : variable-length character string with Unicode support.

7) Different types of Joins in MySQL and Different Types of Constraints
I. Inner Join : Returns records with matching values in both tables.

II. left Join (or Left Outer Join) : Returns all records from the left table, with matching records from the right table.

III. Right Join (or Right Outer Join) : Returns all recods from the right table, with matching records from the left table.

(iv) Full Outer Join : Returns all records from both tables. with null values where no match exists.

v) Cross Join : Returns the cartesian product of both tables

Different Types of constraints
1. Primary key (PK) : Unique identifier for each record.
2. Foreign key (FK) : References the PK of another table
3. Unique : Ensures unique values for a column or set of columns.
4. Not Null : Ensures a column cannot contain null values.
5. check : Ensures data meets specific conditions.
6. Default : Provides a default values for a column

7) ACID properties of Databases
1. Atomicity : Ensures database transactions are treated as a single, indivisible unit.
2. Consistency : Ensures database remains in a consistent state, even after multiple transactions
3. Isolation : Ensures concurrent transactions do not interface with each other,
4. Durability : Ensures committed transactions are permanently stored.