1. int[] arr = Arrays.stream(sc.nextLine().split(" "))

.mapToInt(e -> Integer.parseInt(e))

.toArray();

1. List<Double> n = Arrays.stream(sc.nextLine().split(" "))

.map(Double::parseDouble) .collect(Collectors.toList());

1. String[] words = Arrays.stream(sc.nextLine().split(" "))

.filter(w -> w.length() % 2 == 0)

.toArray(String[]::new)

1. ArrayDeque<Integer> aQueue = new ArrayDeque<>();

Arrays.stream(scanner.nextLine().split("\\s+")

.map(Integer::parseInt)

.forEach(aQueue::offer);

1. List<String> names = new ArrayList<>(Arrays.asList(

"Peter", "Michael", "George", "Victor"));

1. Collections.sort(names);
2. for (Map.Entry<K, V> entry : fruits.entrySet()) {

System.out.printf("%s -> %.2f%n",

entry.getKey(), entry.getValue());

}

1. int sum = Arrays.stream(new int[]{15, 25, 35}).sum();
2. int sum = Arrays.stream(new int[]{15, 25, 35}).sum();
3. int sum = nums.stream() .mapToInt(Integer::intValue).sum();
4. int min = nums.stream().mapToInt(Integer::intValue)

.min().getAsInt();

1. int min = nums.stream()

.min(Integer::compareTo).get();

1. double avg = nums.stream()

.mapToInt(Integer::intValue)

.average()

.getAsDouble();

1. String regex = "\\b(?<day>\\d{2)”

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(dates);

while (matcher.find()) {

System.out.println(String.format("Day: %s, Month: %s, Year: %s", matcher.group("day"), matcher.group("month"), matcher.group("year")));

1. STACK – PUSH(), POP(),PEEK()
2. QUEUE – add(), offer() REMOVE, POLL()

add()–throws exception if queue is full

offer()–returns false if a queue is full

1. public void traverseInOrder(Node node) {

if (node != null) {

traverseInOrder(node.left);

visit(node.value);

traverseInOrder(node.right);

} }

1. Function<T,R>is a function that returns R type

Consumer<T>is a void function

Supplier<T>gets no parameters

Predicate<T>evaluates a condition

BiFunction<T, U, R> accepts two parameters

1. public class Library<T> implements Iterable<T> {

public class Library<Book> implements Iterable<Book> {

private Book[] books;

public Library(Book... books) {

this.books = books;

}

**Question 6: Explain the main method in Java.**

Answer: The main method is the entry point for any Java program. It must be public, static, return no value (void), and accept a String array as a parameter. It signature is:

**Question 7: What are literals in Java?**

Answer: Literals refer to the fixed values assigned to variables in Java. For example, 100, -90, 3.14F, 'A', and "Hello" are all literals.

**Question 8: What is a constructor in Java?**

Answer: A constructor in Java is a block of code similar to a method that's called when an instance of an object is created. Unlike methods, constructors have no explicit return type and have the same name as the class itself.

**Question 9: Explain method overloading in Java.**

Answer: Method overloading is a feature that allows a class to have more than one method having the same name, if their parameter lists are different. It is related to compile-time (or static) polymorphism.

2. Object-Oriented Programming (OOP)

**Question 11: What is Object-Oriented Programming?**

Answer: Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data in the form of fields (often known as attributes or properties) and code in the form of procedures (often known as methods).

**Question 12: What are the main principles of OOP?**

**Encapsulation**: The binding (or wrapping) of data and methods that operate on the data into a single unit called a ‘class’. It also means hiding data (i.e., private variables) from direct access.

**Abstraction**: Hiding the complex implementation details of an operation while exposing a simple interface.

**Inheritance**: Allows a new class to inherit properties and methods of an existing class.

**Polymorphism**: The ability of different classes to provide a unique interface by exposing a method that can behave differently.

**Question 14: What is an interface?**

Answer: An interface in Java is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types. Interfaces cannot contain instance fields. The methods in interfaces are abstract by default.

**Question 16: What is polymorphism?**

Answer: Polymorphism in Java is the ability of an object to take on many forms. Most commonly, it is when a parent class reference is used to refer to a child class object.

**Question 18: What is the "super" keyword?**

Answer: The super keyword in Java is a reference variable that is used to refer to parent class objects. The keyword can be used to call superclass methods and to access the superclass constructor.

**3. Exception Handling and Assertions**

**Question 21: What is exception handling?**

Answer: Exception handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. The core advantage of exception handling is to maintain the normal flow of the application.

**Question 23: What is the finally block?**

Answer: The finally block in Java is used to place important code such as clean-up code, e.g., closing the file, database connections, etc. The finally block executes whether an exception is handled or not.

**Question 24: What is an exception?**

Answer: An exception is a problem that arises during the execution of a program. Exceptions are caught and handled in Java with try-catch blocks**.**

**Question 25: What are checked exceptions?**

Answer: Checked exceptions are checked at compile-time. It means if a method is throwing a checked exception, then it should handle the exception using the try-catch block, or it should declare the exception using the throws keyword, e.g., IOException, SQLException, etc.

**Question 26: What are unchecked exceptions?**

Answer: Unchecked exceptions are not checked at compiled time. It means if your program is throwing an unchecked exception, and even if you didn’t handle/declare that exception, the program won’t give a compilation error. Most of the time, these are programming errors, e.g., Logic errors or improper use of an API. Examples are ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc.

**Question 27: What is the throws keyword?**

Answer: The throws keyword is used to declare an exception. It gives information to the programmer that there may be an exception, so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

**Question 28: What is the difference between throw and throws?**

Answer: throw keyword is used to explicitly throw an exception. throws is used to declare an exception. Checked exceptions cannot be propagated with throws.

**Question 29: What is throw used for?**

Answer: The throw keyword is used within a method. It is used to throw an exception explicitly.

**Question 30: Explain custom exceptions.**

Answer: Sometimes, it is necessary to create a custom exception. This is done by extending the Exception class.

4. Core APIs

**Question 31: What is the String API?**

Answer: The String API is a set of classes and methods that operate on strings, including manipulating characters, comparing strings, searching strings, extracting substrings, and creating copies of strings with alterations.

**Question 32: Explain "String", "StringBuilder", and "StringBuffer".**

String: Immutable sequence of characters.

StringBuilder: Mutable sequence of characters. Not thread-safe.

StringBuffer: Mutable sequence of characters. Thread-safe.

Question 33: What is an immutable object?

Answer: An immutable object is an object whose state cannot be modified after it is created. Strings are a good example of immutable objects.

**Question 34: What are wrapper classes?**

Answer: Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they "wrap" the primitive data type into an object of that class. Examples include Integer, Character, Double etc.

**Question 35: Explain autoboxing and unboxing.**

Answer: Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, etc. Unboxing is the reverse process, where the object is converted back to a primitive type.

**Question 36: What is the "Collections" API?**

Answer: The Collections API is a set of classes and interfaces that support operations on collections of objects. These classes and interfaces are grouped under java.util package. The collections framework provides both interfaces that define various collections and classes that implement them.

**Question 37: What are generics?**

Answer: Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces, and methods. Much like the more familiar formal parameters used in method declarations, type parameters provide a way for you to reuse the same code with different inputs. The inputs to formal parameters are values, while the inputs to type parameters are types.

**Question 38: What is the "Iterator" interface?**

Answer: The Iterator interface provides methods to iterate over any Collection. It works similarly to a "looping" mechanism, and it allows you to traverse through a collection and remove elements from the collection selectively if desired.

**Question 39: Explain the difference between "Iterator" and "ListIterator".**

Iterator: Enables you to traverse through a collection in the forward direction only, for obtaining or removing elements.

ListIterator: Extends Iterator to allow bidirectional traversal of the list and also to modify elements.

**5. Concurrency**

**Question 45: What is the "synchronized" keyword?**

Answer: The synchronized keyword is used to control the access of multiple threads to any shared resource. It can be used to define a method or a block of code that can only be accessed by one thread at a time.

**Question 46: What is deadlock?**

Answer: Deadlock is a part of multithreading. It occurs when two or more threads get blocked forever, waiting for each other. Deadlock occurs due to improper thread synchronization.

**Question 48: What is the "volatile" keyword?**

Answer: The volatile keyword is used to indicate that a variable's value will be modified by different threads. Declaring a volatile Java variable means:

The value of this variable will never be cached thread-locally: all reads and writes will go straight to "main memory";

Access to the variable acts as though it is enclosed in a synchronized block, synchronized on itself.

**Question 49: What is thread safety?**

Answer: Thread safety means that a method or class instance can be used by multiple threads at the same time without any problem. An example of thread safety is a synchronized method.

6. Stream API

**Question 51: What is the Stream API in Java?**

Answer: The Stream API in Java provides a new abstraction called Stream, which allows you to process data in a declarative way. It supports operations like map, filter, limit, reduce, find, match, and sort, on collections of objects.

**Question 52: What are the benefits of using Streams?**

Answer: Streams can make the code more concise and readable. They can simplify code to perform bulk operations sequentially or parallelly. Streams don't store data and, instead, operate on the source data structures (e.g., collections) directly.

Question 54: What is the difference between 'map' and 'flatMap' in Streams?

map: It transforms each element in the stream using the given function. It is a one-to-one mapping.

flatMap: It helps in converting one type of stream element into another type and flattens the Stream of Streams into a Stream.

**Question 55: What is the "filter" method in a "Stream"?**

Answer: The filter method is used to evaluate each element in a stream using a predicate. If the predicate evaluates to true, the element is included in the resulting stream.

**Question 57: What are terminal operations on "Streams"?**

Answer: Terminal operations produce a result from a stream pipeline. Terminal operations include operations like forEach, reduce, collect, and sum.

**Question 58: What is the "collect" method in "Streams"?**

Answer: The collect method is a terminal operation that transforms the elements of a stream into a different kind of result, e.g., a List, a Map, or even an Integer.

**Question 60: Explain the "reduce" operation in "Streams".**

Answer: The reduce operation combines all elements of the stream into a single result by applying a binary operator. This operation takes two parameters: an initial value, and a binary operator function.

**7. JDBC**

**Question 61: What is JDBC?**

Answer: JDBC (Java Database Connectivity) is an API that enables Java programs to execute SQL statements. This allows Java applications to interact with any SQL-compliant database.

**Question 62: What are the core components of JDBC?**

Answer: The core components of JDBC include DriverManager, Driver, Connection, Statement, ResultSet, and SQLException.

**Question 63: How do you connect to a database in Java?**

Connection conn = DriverManager.getConnection("jdbc:subprotocol:subname", "user", "password");

**Question 64: What are "Statement" and "PreparedStatement"?**

Statement: Used to execute a simple SQL query with no parameters.

PreparedStatement: Used for executing SQL statements multiple times or when you need to bind parameters to the query.

Question 65: How do you execute a query in JDBC?

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM myTable");

**Question 66: What is "ResultSet" in JDBC?**

Answer: ResultSet is a table of data representing a database result set, which is generated by executing a statement that queries the database.

**Question 67: What do you mean by batch processing in JDBC?**

Answer: Batch processing in JDBC is used to execute multiple SQL statements as a single batch, which reduces the number of communication rounds between the application and the database server.

**Question 68: Explain the types of JDBC drivers.**

Answer: There are four types of JDBC drivers:

Type 1: JDBC-ODBC bridge

Type 2: Native-API/partly Java driver

Type 3: Net-protocol/all-Java driver

Type 4: Native-protocol/all-Java driver

**Question 70: What is Connection Pooling?**

Answer: Connection pooling is a technique used to improve performance in applications that need to make calls to a database by reusing the connections instead of creating a new one each time.

**Question 78: What is the 'subList' method in List?**

Answer: The subList method creates a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

**Question 79: How can you synchronize a "List"?**

List<String> syncList = Collections.synchronizedList(new ArrayList<String>());

Question 80: What are the benefits of using "Generics" with "List"?

Answer: Generics provide type safety. They ensure that you insert only the specified type of objects into your list. This reduces bugs and eliminates the need for typecasting.

drop database exam;

create database exam;

TRUNCATE  database

use exam;

1. CREATE TABLE workers

(

id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(40) NOT NULL,

last\_name VARCHAR(40) NOT NULL,

age INT,

personal\_number VARCHAR(20) UNIQUE,

salary DECIMAL(19, 2),

is\_armed BOOLEAN NOT NULL,

start\_date DATE,

preserve\_id INT,

FOREIGN KEY (preserve\_id) REFERENCES preserves (id),

);

1. SELECT CONCAT(name, ' is in South Hemisphere'),

latitude,

longitude,

area \* id,

LOWER(type),

established\_on

FROM preserves

WHERE latitude < 0;

-- 3

UPDATE workers

SET salary=salary + 500

where position\_id = 5

OR position\_id = 8;

-- 4

DELETE

FROM preserves

WHERE preserves.established\_on IS NULL;

-- 5

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name,

DATEDIFF('2024-01-01', start\_date) AS days\_of\_experience

FROM workers

WHERE DATEDIFF('2024-01-01', start\_date) > 5 \* 365

ORDER BY days\_of\_experience DESC

LIMIT 10;

-- 6

SELECT w.id AS worker\_id,

w.first\_name,

w.last\_name,

p.name AS preserve\_name,

c.country\_code

FROM workers w

JOIN preserves p ON w.preserve\_id = p.id

JOIN countries\_preserves cp ON p.id = cp.preserve\_id

JOIN countries c ON cp.country\_id = c.id

WHERE w.age < 50

AND w.salary > 5000

ORDER BY c.country\_code ASC;

-- 7

SELECT p.name AS name,

COUNT(CASE WHEN w.is\_armed = TRUE THEN 1 END) AS armed\_workers

FROM preserves p

LEFT JOIN workers w ON p.id = w.preserve\_id

GROUP BY p.name

HAVING COUNT(CASE WHEN w.is\_armed = TRUE THEN 1 END)

ORDER BY armed\_workers DESC, p.name ASC;

;

-- 8

SELECT pr.name AS preserve,

co.country\_code,

YEAR(pr.established\_on) AS founded\_in

FROM preserves pr

JOIN countries\_preserves cp ON pr.id = cp.preserve\_id

JOIN countries co ON cp.country\_id = co.id

WHERE MONTH(pr.established\_on) = 5

ORDER BY pr.established\_on ASC

LIMIT 5;

-- 9

SELECT id,

name,

CASE

WHEN area <= 100 THEN 'very small'

WHEN area <= 1000 THEN 'small'

WHEN area <= 10000 THEN 'medium'

ELSE 'very large'

END AS category

FROM preserves

ORDER BY area DESC;

public interface Notification{

public void notify();

}

public class Employee{

private Notification notification;

public Employee(Notification notification){

this.notification = notification;

}

public void notifyUser(){

notification.notify();

}

}

public class EmailNotification implements Notification{

public void notify(){

//implement notification via email

}

}

public static void main(String [] args){

Notification notification = new EmailNotification();

Employee employee = new Employee(notification);

employee.notifyUser();

}