Answers

1. **Synchronization** in java is to stop two threads from using a synchronized method. Multi-threaded programs may often come to a situation where multiple threads try to access the same resources and finally produce erroneous and unforeseen results.

Eg.

class Sender

{

    public void send(String msg)

    {

        System.out.println("Sending\t"  + msg );

        try

        {

            Thread.sleep(1000);

        }

        catch (Exception e)

        {

            System.out.println("Thread  interrupted.");

        }

        System.out.println("\n" + msg + "Sent");

    }

}

class ThreadedSend extends Thread

{

    private String msg;

    private Thread t;

    Sender  sender;

      ThreadedSend(String m,  Sender obj)

    {

        msg = m;

        sender = obj;

    }

    public void run()

    {

        synchronized(sender)

        {

            // synchronizing the snd object

            sender.send(msg);

        }

    }

}

class SyncDemo

{

    public static void main(String args[])

    {

        Sender snd = new Sender();

        ThreadedSend S1 =

            new ThreadedSend( " Hi " , snd );

        ThreadedSend S2 =

            new ThreadedSend( " Bye " , snd );

        S1.start();

        S2.start();

        try

        {

            S1.join();

            S2.join();

        }

        catch(Exception e)

        {

            System.out.println("Interrupted");

        }

    }

}

**2. Lamda Expressions example( lambda operator -> body )**

import java.util.ArrayList;

class Test

{

    public static void main(String args[])

    {

        ArrayList<Integer> arrL = new ArrayList<Integer>();

        arrL.add(1);

        arrL.add(2);

        arrL.add(3);

        arrL.add(4);

        arrL.forEach(n -> System.out.println(n));

        arrL.forEach(n -> { if (n%2 == 0) System.out.println(n); });

    }

}

**3. Collections class methods :**

addAll() :   
 It is used to adds all of the specified elements to the specified collection.

copy() :   
It is used to copy all the elements from one list into another list.

fill() :   
It is used to replace all of the elements of the specified list with the specified elements.

list() :   
It is used to get an array list containing the elements returned by the specified enumeration in the order in which they are returned by the enumeration.

max() :   
It is used to get the maximum value of the given collection, according to the natural ordering of its elements.

min():  
It is used to get the minimum value of the given collection, according to the natural ordering of its elements.

rotate() :   
It is used to rotate the elements in the specified list by a given distance.

**4. Collections hierarchy** 

**5. Custom list creation**

   public static void main(String args[])

    {

        int roll[] = {1, 2, 3, 4};

        String name[] = {"Shubham", "Atul", "Ayush", "Rupesh"};

        int marks[] = {100, 99, 93, 94};

        long phone[] = {8762357381L, 8762357382L, 8762357383L,

                        8762357384L

                       };

        CustomArrayList custom = new CustomArrayList();

        custom.addValues(roll, name, marks, phone);

    }

class CustomArrayList

{

    int n=4;

    class Data

    {

        int roll;

        String name;

        int marks;

        long phone;

        Data(int roll, String name, int marks, long phone)

        {

            this.roll = roll;

            this.name = name;

            this.marks = marks;

            this.phone = phone;

        }

    }

public void addValues(int roll[], String name[], int marks[],

                          long phone[])

    {

        ArrayList<Data> list=new ArrayList<>();

        for (int i = 0; i < n; i++)

        {

            list.add(new Data(roll[i], name[i], marks[i],

                                              phone[i]));

        }

        printValues(list);

    }

**6. Difference between ArrayList and LinkedList**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1) ArrayList internally uses a **dynamic array** to store the elements. | LinkedList internally uses a **doubly linked list** to store the elements. |
| 2) Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory. |
| 3) An ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. |
| 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data. |

**7. Thread class methods**

**clone(), currentThread(), getPriority(), interrupt(), isAlive(), join()**

**8. Advantages of generic classes**

**1) Type-safety:** We can hold only a single type of objects in generics. It doesn?t allow to store other objects.

**2) Type casting is not required:** There is no need to typecast the object.

**3) Compile-Time Checking:** It is checked at compile time so problem will not occur at runtime. The good programming strategy says it is far better to handle the problem at compile time than runtime.

9) Recursion will be filled in data segment.

**10. Streams**

Filter()

class GFG {

    public static void main(String[] args)

    {

        List<Integer> list = Arrays.asList(3, 4, 6, 12, 20);

        list.stream().filter(num -> num % 5==0).forEach(System.out::println);

    }

}

Reduce() :

class GFG {

      public static void main(String[] args)

    {

        System.out.println("The stream after applying "

                           + "the function is : ");

        List<Integer> list = Arrays.asList(3, 6, 9, 12, 15);

        list.stream().map(number -> number \* 3).forEach(System.out::println);

    }

}

**11. JDBC | JDBC template**

Spring **JdbcTemplate** is a powerful mechanism to connect to the database and execute SQL queries. It internally uses JDBC api, but eliminates a lot of problems of JDBC API.

**12. Starters in Spring Boot**

1. The Web Starter

## **The Test Starter**

## **The Data JPA Starter**

## **The Mail Starter**

**13. Applying external configuration**

properties files, YAML files, environment variables, and command-line arguments.

**14. Difference between CRUD and JPA**

* PagingAndSortingRepository extends CrudRepository
* JpaRepository extends PagingAndSortingRepository

The **CrudRepository** interface provides methods for CRUD operations, so it allows you to create, read, update and delete records without having to define your own methods.

The **PagingAndSortingRepository** provides additional methods to retrieve entities using pagination and sorting.

Finally the **JpaRepository** add some more functionality that is specific to JPA.

**15. HTTP response status codes**

1. Informational responses (100–199),
2. Successful responses (200–299),
3. Redirects (300–399),
4. Client errors (400–499),
5. and Server errors (500–599).