**Question Bank**

**Module 4**

**Object Oriented Programming Using Java (21CIC34)**

**Semester:3rd Session : July-Dec 2022**

1. What is multiple inheritance?Multiple inheritance is not supported through class in java, but it is possible by an interface, why?Explain with an example.(CO4)

**Answer :-** When the child class extends from more than one superclass, it is known as multiple inheritance. However, Java does not support multiple inheritance.

**Or**

**If a child class inherits the of two superclasses then it is said to be multiple inheritance**

Multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.

interface Printable{ void print();

}

interface Showable{ void print();

}

class TestInterface3 implements Printable, Showable

{

public void print()

{

System.out.println("Hello");

}

public static void main(String args[])

{

TestInterface3 obj = new TestInterface3();

obj.print();

} }

Output :- Hello

As we can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface3, so there is no ambiguity.

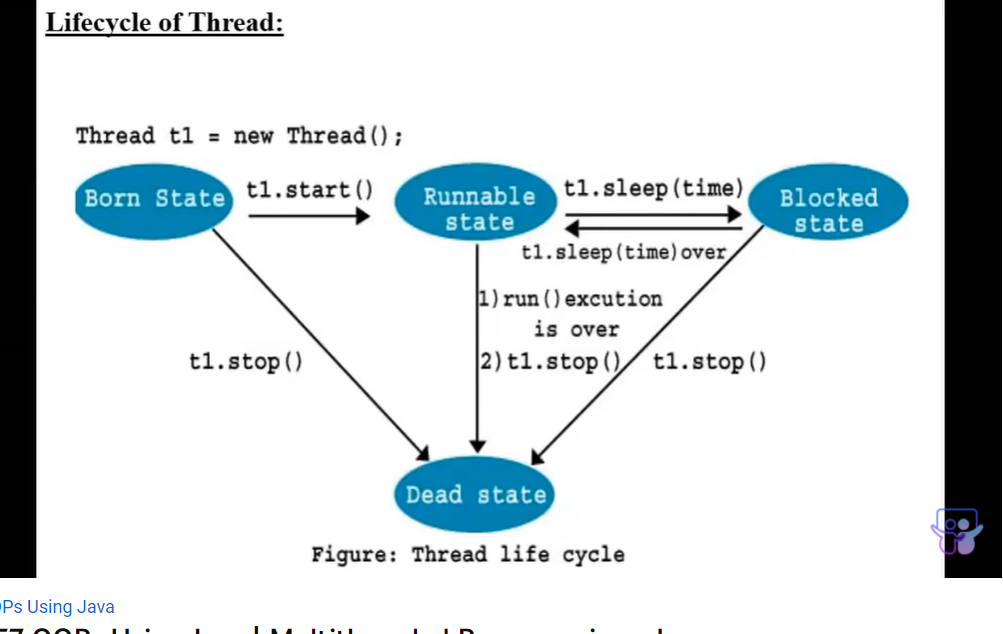
1. Distinguish between the abstract class and interface.(CO4)

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| 5) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 6) An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are publicand final by default. |
| 9)**Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

## What is thread and multi-threading in Java? Discuss the life cycle of the thread in java. (CO4)

Thread: thread is an individual working component. It is a separate path of execution. Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.

**Multithreading in Java** is a process of executing multiple threads simultaneously.

****

**BORN STATE(new state):** Whenever a new thread is created, it will be in born state(new state)

**Active:** When a thread invokes the start() method, it moves from the new state(born state) to the active state. The active state contains two states within it: one is **runnable**, and the other is **running**. from runnable state u can explicitly kill the thread by thread.stop().

**Blocked or Waiting:** Whenever a thread is inactive for a span of time (not permanently)or whenever we make thread to sleep then, either the thread is in the blocked state or is in the waiting state, whenever sleep time completes the thread again goes to runnable state if u stop the thread explicitly by stop() then thread goes to dead state

**Terminated(Dead state):** A terminated or dead thread means the thread is no more in the system. In other words, the thread is dead, and there is no way one can respawn the dead thread.

## What is Multi-threading? What are the ways to create thread and multiple threads in java.(CO4)

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is an individual working element . thread is independent, it has separate path of execution,

Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

**way to create a thread:**

**1)by extending Thread**

**2)by implementing Runnable**

**1)by extending Thread**

|  |
| --- |
| class x extends Thread  {      public void run()      {          System.out.println("from x thread");          for(int i=0;i<5;i++)          {              System.out.println(i);          }      }  }  class threadbyextendingthread  {      public static void main(String[] args)      {          System.out.println("from main thread");          x obj=new x();          obj.start();      }  } |

**Output:**

**from main thread**

**from x thread**

**0**

**1**

**2**

**3**

**4**

**2)by implementing Runnable**

class vr implements Runnable

{

    public void run()

    {

        System.out.println("from thread created by runnable");

    }

}

public class threadingbyimplementingrunnabel

{

    public static void main(String[] args) {

        vr obj=new vr();

        Thread th=new Thread(obj);

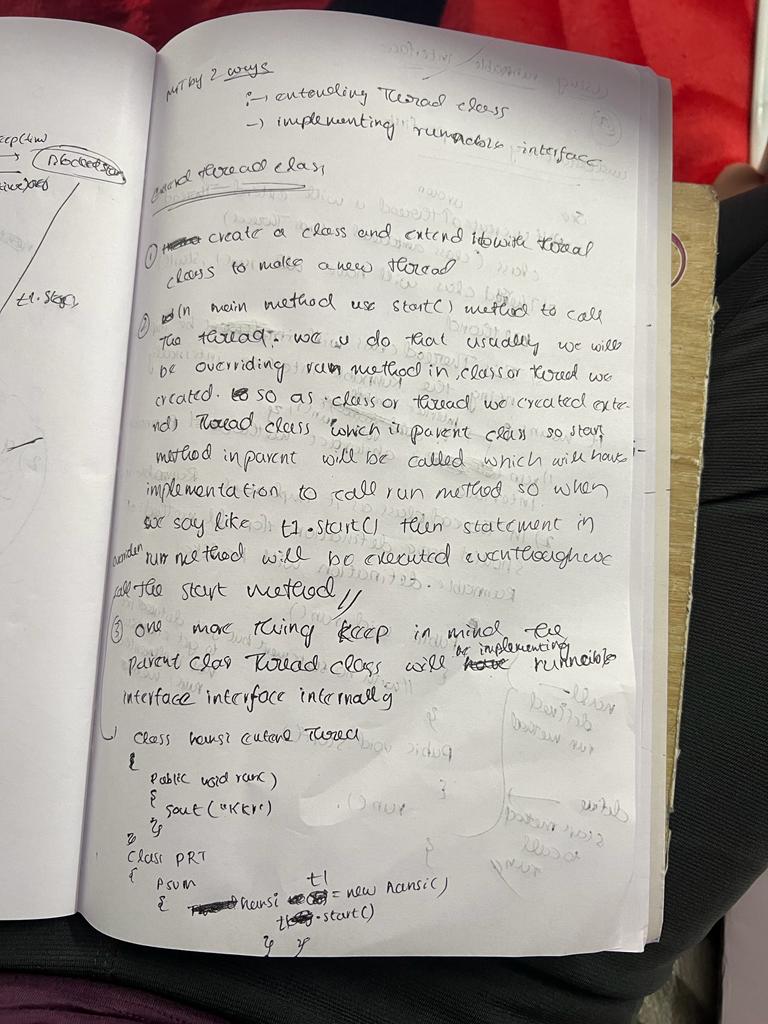
        th.start();

    }

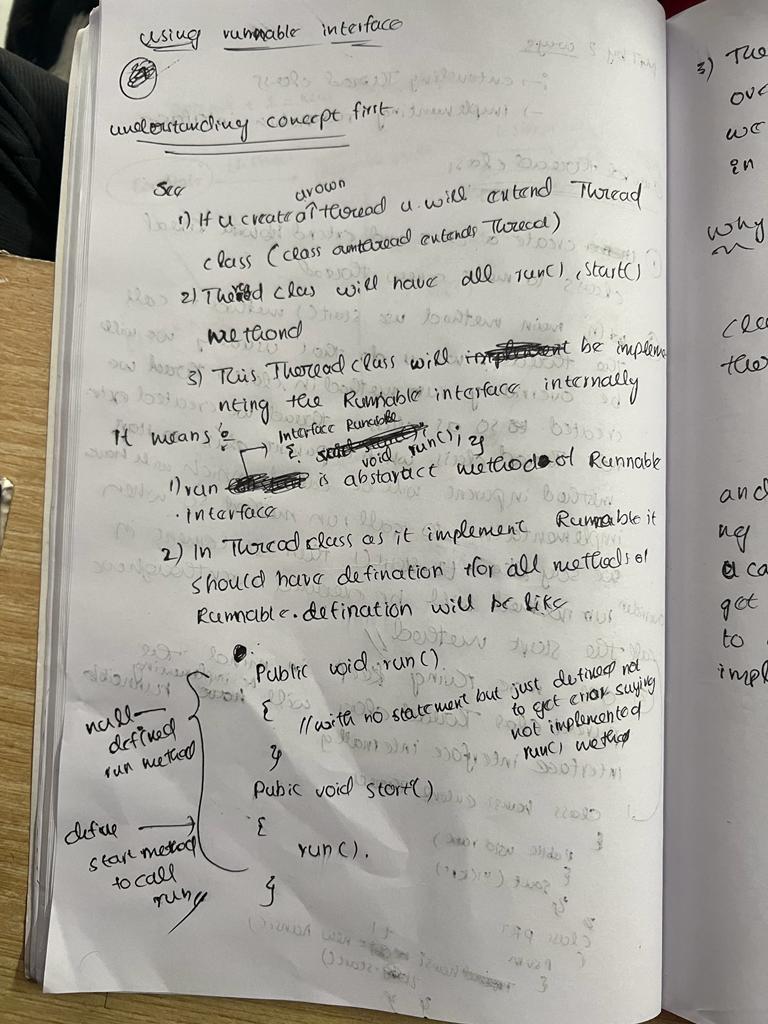
}

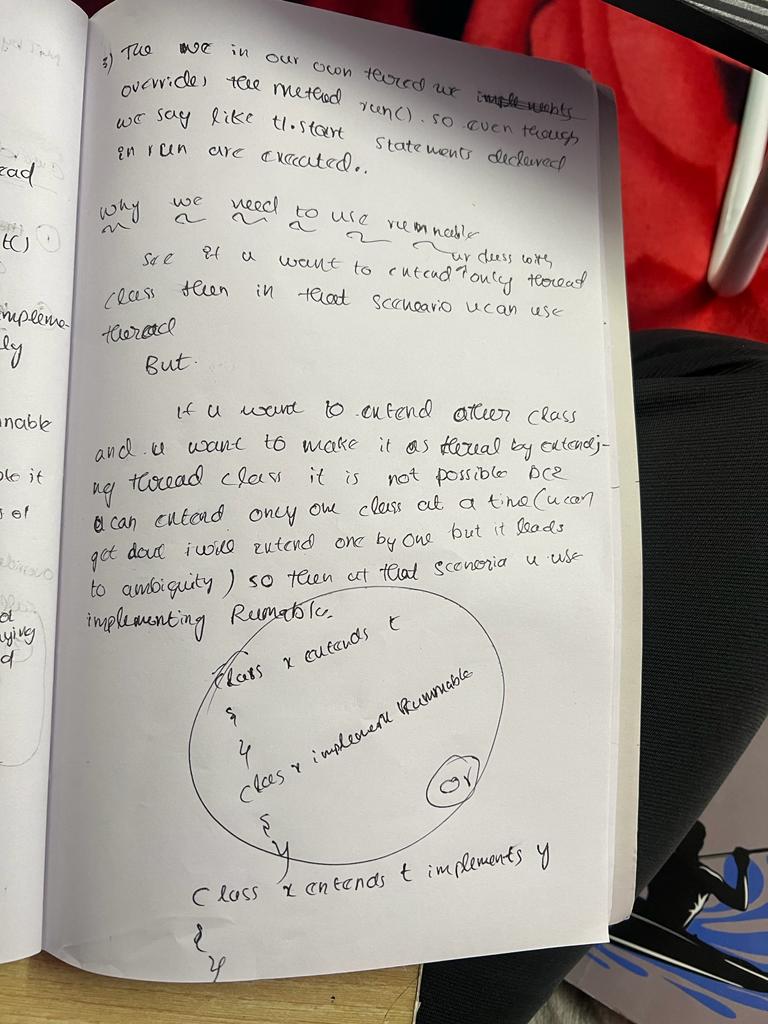
**Output:**

**from thread created by runnable**

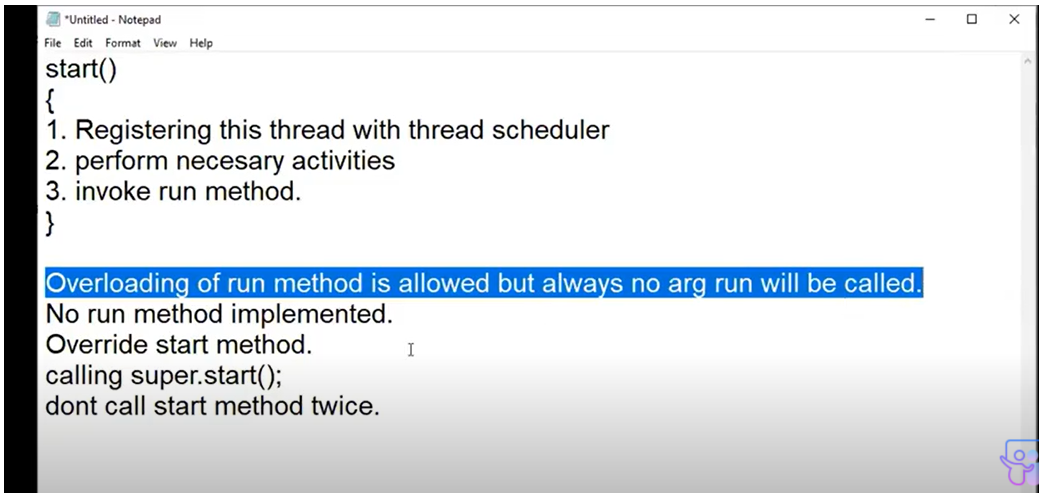


**2)by implementing Runnable**





**How start method works and what all start method does**

****

**CREATING MULTIPLE THREADS IN JAVA**

**CREATE MULTIPLE OBJ OF SAME THREAD LEADS TO CREATION OF MULTIPLE THREADS**

class vr implements Runnable

{

    public void run()

    {

        System.out.println("from thread created by runnable");

    }

}

public class threadingbyimplementingrunnabel

{

    public static void main(String[] args) {

        vr obj=new vr();

        Thread th=new Thread(obj);

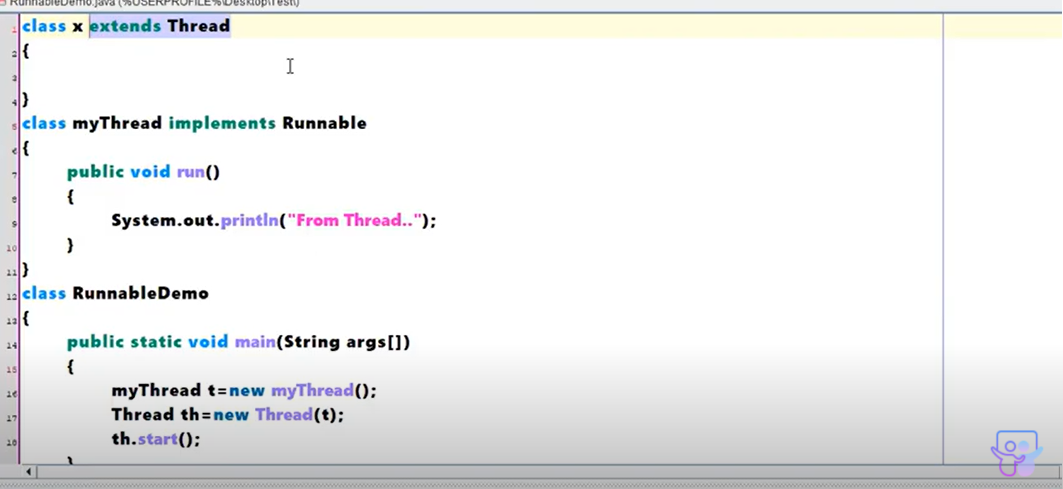
        th.start();

        Thread th1=new Thread(obj);

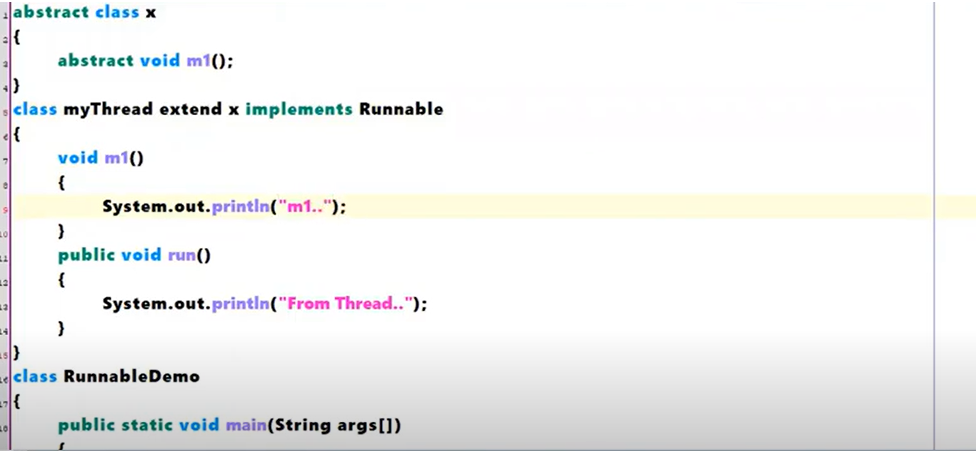
        th1.start();

    }

}

****

****

****

## Define thread priorities in Java with suitable example in Java .(CO4)

# Priority of a Thread (Thread Priority)

Each thread has a priority. Priorities are represented by a number between 1 and 10. In most cases, the thread scheduler schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses. Note that not only JVM a Java programmer can also assign the priorities of a thread explicitly in a Java program.

## Setter & Getter Method of Thread Priority

Let's discuss the setter and getter method of the thread priority.

**public final int getPriority():** The java.lang.Thread.getPriority() method returns the priority of the given thread.

**public final void setPriority(int newPriority):** The java.lang.Thread.setPriority() method updates or assign the priority of the thread to newPriority. The method throws IllegalArgumentException if the value newPriority goes out of the range, which is 1 (minimum) to 10 (maximum).

## 3 constants defined in Thread class:

1. public static int MIN\_PRIORITY
2. public static int NORM\_PRIORITY
3. public static int MAX\_PRIORITY

Default priority of a thread is 5 (NORM\_PRIORITY). The value of MIN\_PRIORITY is 1 and the value of MAX\_PRIORITY is 10.

### Example of priority of a Thread:

**FileName:** ThreadPriorityExample.java

1. // Importing the required classes
2. **import** java.lang.\*;
4. **public** **class** ThreadPriorityExample **extends** Thread
5. {
7. // Method 1
8. // Whenever the start() method is called by a thread
9. // the run() method is invoked
10. **public** **void** run()
11. {
12. // the print statement
13. System.out.println("Inside the run() method");
14. }
16. // the main method
17. **public** **static** **void** main(String argvs[])
18. {
19. // Creating threads with the help of ThreadPriorityExample class
20. ThreadPriorityExample th1 = **new** ThreadPriorityExample();
21. ThreadPriorityExample th2 = **new** ThreadPriorityExample();
22. ThreadPriorityExample th3 = **new** ThreadPriorityExample();
24. // We did not mention the priority of the thread.
25. // Therefore, the priorities of the thread is 5, the default value
27. // 1st Thread
28. // Displaying the priority of the thread
29. // using the getPriority() method
30. System.out.println("Priority of the thread th1 is : " + th1.getPriority());
32. // 2nd Thread
33. // Display the priority of the thread
34. System.out.println("Priority of the thread th2 is : " + th2.getPriority());
36. // 3rd Thread
37. // // Display the priority of the thread
38. System.out.println("Priority of the thread th2 is : " + th2.getPriority());
40. // Setting priorities of above threads by
41. // passing integer arguments
42. th1.setPriority(6);
43. th2.setPriority(3);
44. th3.setPriority(9);
46. // 6
47. System.out.println("Priority of the thread th1 is : " + th1.getPriority());
49. // 3
50. System.out.println("Priority of the thread th2 is : " + th2.getPriority());
52. // 9
53. System.out.println("Priority of the thread th3 is : " + th3.getPriority());
55. // Main thread
57. // Displaying name of the currently executing thread
58. System.out.println("Currently Executing The Thread : " + Thread.currentThread().getName());
60. System.out.println("Priority of the main thread is : " + Thread.currentThread().getPriority());
62. // Priority of the main thread is 10 now
63. Thread.currentThread().setPriority(10);
65. System.out.println("Priority of the main thread is : " + Thread.currentThread().getPriority());
66. }
67. }

**Output:**

Priority of the thread th1 is : 5

Priority of the thread th2 is : 5

Priority of the thread th2 is : 5

Priority of the thread th1 is : 6

Priority of the thread th2 is : 3

Priority of the thread th3 is : 9

Currently Executing The Thread : main

Priority of the main thread is : 5

Priority of the main thread is : 10

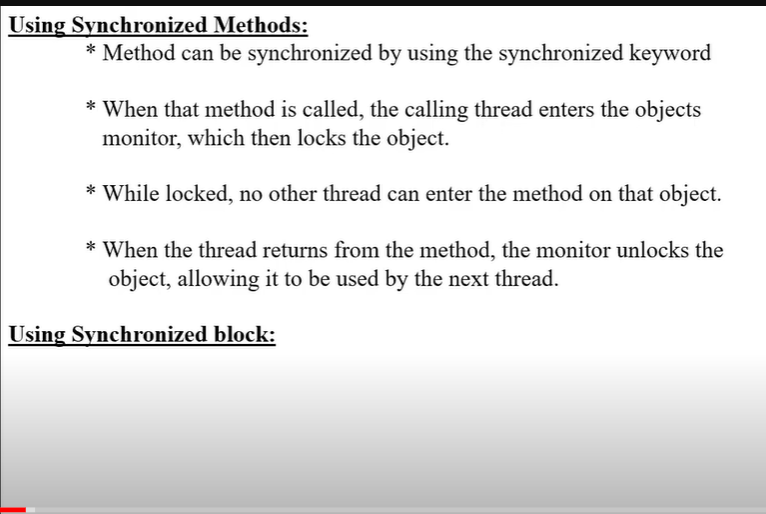
## Discuss briefly Runnable interface .How can a thread be created? Explain with an example .(CO4)

## Write a program to create single thread and multiple thread in Java. (CO4)

## What is synchronization? What are different ways of synchronizing threads? Give example for each. (CO4)

[Multi-threaded](https://www.geeksforgeeks.org/multithreading-in-java/)programs may often come to a situation where multiple threads try to access the same resources and finally produce erroneous and unforeseen results.

So it needs to be made sure by some synchronization method that only one thread can access the resource at a given point in time. Java provides a way of creating threads and synchronizing their tasks using synchronized blocks. Synchronized blocks in Java are marked with the synchronized keyword. A synchronized block in Java is synchronized on some object. All synchronized blocks synchronize on the same object can only have one thread executing inside them at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.



## Explain and Demonstrate the inter thread communication with an example. (CO4)

**Inter-thread communication** or **Co-operation** is all about allowing synchronized threads to communicate with each other

## Create two threads in which one displays “Dept of CSE” for every 50 seconds and the other displays “ Jain University” for every 100 seconds continuously.(CO4)

class cse extends Thread

{

public void run()

{

try

{

int t=10;

while(t!=0)

{

System.out.println("Dept of CSE");

Thread.sleep(50000);

t--;

}

}

catch(InterruptedException e)

{

}

}

}

class jain implements Runnable

{

public void run()

{

try

{

int t=10;

while(t!=0)

{

System.out.println("Jain University");

Thread.sleep(100000);

}

}

catch(Exception e)

{

}

}

}

public class questionbankquestion {

public static void main(String[] args)

{

//initializing first thread

cse obj=new cse();

obj.start();

//initialising second thread

jain obj2=new jain();

Thread obj3=new Thread(obj2);

obj3.start();

    }

}

## With proper syntax and example explain following thread methods : (1) wait( ) (2) sleep( ) (3) resume( ) (4) notify( ) (CO4)

wait() method is a part of java.lang.Object class. When wait() method is called, the calling thread stops its execution until notify() or notifyAll() method is invoked by some other Thread.

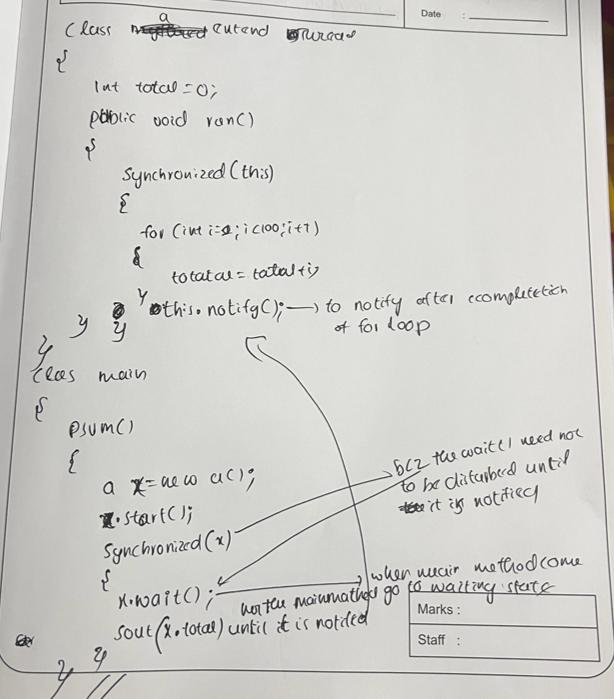
**Syntax:**

**Public final void wait() throws InterruptedException**

p The *notify() method* is defined in the Object class, which is Java’s top-level class. It’s used to wake up only one thread that’s waiting for an object, and that thread then begins execution. The thread class notify() method is used to wake up a single thread.

Or

Notify is used is to wake up a thread which is went for waiting by thread.wait)(



# ava Thread resume() method

The **resume()** method of thread class is only used with suspend() method. This method is used to resume a thread which was suspended using suspend() method. This method allows the suspended thread to start again.

## Syntax

1. **public** **final** **void** resume()

## Return value

This method does not return any value.

## Exception

**SecurityException:** If the current thread cannot modify the thread.

## Example

1. **public** **class** JavaResumeExp **extends** Thread
2. {
3. **public** **void** run()
4. {
5. **for**(**int** i=1; i<5; i++)
6. {
7. **try**
8. {
9. // thread to sleep for 500 milliseconds
10. sleep(500);
11. System.out.println(Thread.currentThread().getName());
12. }**catch**(InterruptedException e){System.out.println(e);}
13. System.out.println(i);
14. }
15. }
16. **public** **static** **void** main(String args[])
17. {
18. // creating three threads
19. JavaResumeExp t1=**new** JavaResumeExp ();
20. JavaResumeExp t2=**new** JavaResumeExp ();
21. JavaResumeExp t3=**new** JavaResumeExp ();
22. // call run() method
23. t1.start();
24. t2.start();
25. t2.suspend(); // suspend t2 thread
26. // call run() method
27. t3.start();
28. t2.resume(); // resume t2 thread
29. }
30. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=JavaResumeExp)

**Output:**

Thread-0

1

Thread-2

1

Thread-1

1

Thread-0

2

Thread-2

2

Thread-1

2

Thread-0

3

Thread-2

3

Thread-1

3

Thread-0

4

Thread-2

4

Thread-1

4

## What is package ? How do we create it ? Give the example to create and to access package.(CO4)

What is package

In Java, a **package** is a group of classes, interfaces, enumeration, and annotations. Java contains many pre-defined packages such as **java.lang, java.io, java.net,** etc. When we create any Java program the **java.lang package is imported by default**. We need not to write the package name at the top of the program. We can also create our own package by providing the name that we want. In this section, we will learn **how to create a package in Java**.

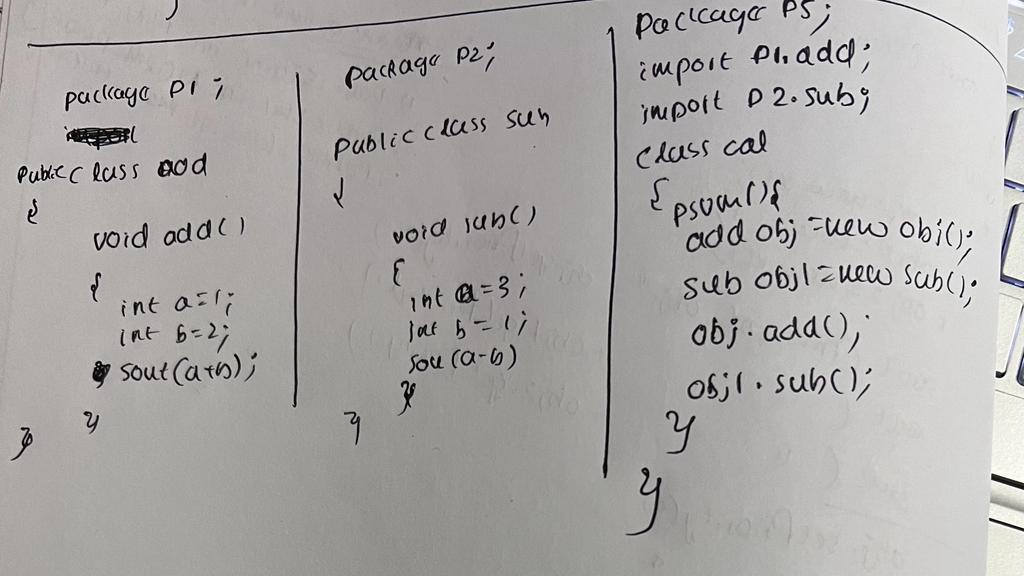
Creating a package

1)think a name of package

2)on the top of the program include ur pack name with syntax: package packname;

3)if u want to import another package use import packname

4)if u want to import



## Explain the visibility of each Access specifier with respect to package.(CO4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| public | The fields with public access specifier are accessible for all classes inside and outside of a package | | | | [Try it »](https://www.w3schools.com/java/tryjava_multi.asp?filename=demo_mod_public2&multi=demo_mod_public2_multi) |
| private | The fields with private are only accessible within the declared class | | | | [Try it »](https://www.w3schools.com/java/tryjava.asp?filename=demo_access_mod) |
| *default* | The fields with default only accessible by classes in the same package. This is used when you don't specify a modifier. | | | | [Try it »](https://www.w3schools.com/java/tryjava.asp?filename=demo_mod_default2) |
| protected | The fields with protected are accessible in the same package and **subclasses**  **Of other packages also** . | | | |  |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside**  **package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |