Multithreading

Thread

➨Threads in Java:

A thread is actually a lightweight process. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread and each thread defines a separate path of the execution.

Every thread in Java is created and controlled by a unique object of the java.lang.Thread class.

Java Multithreading is mostly used in games, animation, etc.

Threads are independent. If there occurs exception in one thread, it doesn't affect other threads.

- There are two ways to define a thread in java:-

  - By extending Thread class
  - By implementing Runnable interface

1. By extending Thread class-

Thread class provide constructors and methods to create and perform operations on a thread. The first way to create a thread is to create a new class that extends Thread class.

```java
class MyThread extends Thread{
    public void run() // override the run()
    {
        for(int a=0;a<10;a++) {
            System.out.println("Child thread");
        }
    }
}
public class Demo_Thread2 {
    public static void main(String[] args) {
        MyThread t = new MyThread();
        t.start(); // start the thread
        for(int i=0;i<10;i++) {
            System.out.println("main thread");
        }

    }

}
```

- **Case I: Thread Schedular**

Thread schedular is the part of JVM and it is responsible to schedule thread execution i.e if multiple threads are waiting to get the chance of execution then, in which order threads will be executed is decided by thread schedular

- **Case II: Overloading of run() is possible or not?**

Overloading of run() is possible but Thread class start() method invoke only no argument run().

We have to call explicitly the other overloaded method like normal method call.

- **Case III: Overriding of run() is possible or not?**

  Overriding of run() is not possible

- **Case IV:**

  After starting a thread if we are trying to restart the same thread once again, so we get run time exception

2. By implementing Runnable Interface

Runnable interface is present in java.lang package and it contain only one method i.e run() method

public void run()-> is used to perform action for a thread.

- The easiest way to create a thread is to create a class that implements the Runnable interface.
- To implement Runnable, a class need only implement a single method called run( )
- We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.
- After the new thread is created, it will not start running until we call its start( ) method, which is declared within Thread. In essence, start( ) executes a call to run( ).

The start( ) method is shown as:

void start( )

```java
class MyThread implements Runnable{
    public void run() {
        for(int a=0;a<10;a++) {
            System.out.println("Child thread through Runnable interface");
        }
    }
}
public class Demo_Thread2 {

    public static void main(String[] args) {
        MyThread r = new MyThread();
        Thread t = new Thread(r);
        t.start();
        for(int i=0;i<10;i++) {
            System.out.println("main thread");
        }

    }

}
```

- **Various Constructor with in Thread class : -**

  1. Thread t = new Thread();

  2. Thread t = new Thread(Runnable r);

Thread Priorities

- Every thread in java has some priority i.e it may be default priority which is generated by JVM or customized priority which is provided by programmer explicitly
- Range of thread priority (1 to 10)
- Thread class defines some constants to represent the priority

  Thread.MIN_PRIORITY -> 1

  Thread.NORM_PRIORITY -> 5

  Thread.MAX_PRIORITY -> 10

➡ Get & set the priority of Thread :-

Thread class defines the following methods to get & set priority of a thread

(i) public final int getPriority()

(ii) public final void setPriority(int p)

```java
class Mythrea extends Thread{
    public void run() {
        for(int i=0;i<10;i++) {
            System.out.println("child thread ");
        }
    }
}
public class Demo_Thread2{

    public static void main(String[] args) {
        Mythrea t1 = new Mythrea();
        t1.setPriority(9);  // set the priority of thread
        t1.start();
        for(int i=0;i<10;i++) {
            System.out.println("main thread");
        }
    }
}
```

- **But some OS or platform don’t provide the proper support for thread priority**

Commonly used Constructors of Thread class:-

- Thread()
- Thread(String name)
- Thread(Runnable r)
- Thread(Runnable r, String name)

Property	yield()	join()	sleep()
Purpose	If a thread wants to pass its execution to give the chance for remaining threads of same priority then we use yield() method	If a thread wants to wait until completing some other thread then we should go for join() method	If a thread don't want to perform any operation for a particular amount of time then we use sleep() method
Is it overloaded ?	No	Yes	Yes
Is it final ?	No	Yes	No
Is it static ?	Yes	No	Yes
Is it throws Interrupted Exception	No	Yes	Yes

How a thread can interrupt another thread ?

- A thread can interrupt sleeping thread or waiting thread by using interrupt() method of Thread class.

```
public void interrupt( )
```

```java
//implement the use of interrupt() method
class MyThread extends Thread{
    public void run() {
        try {
            for(int i=0;i<5;i++) {
                System.out.println("thread");
                Thread.sleep(1000);
            }
        }catch(InterruptedException e) {
            System.out.println("interrupted");
        }
    }
}
public class Demo_Thread2{

    public static void main(String[] args) {
        MyThread t = new MyThread();
        t.start();
        t.interrupt();  // line 1
        System.out.println("end of main thread");
    }

}
```

- Thread Class vs Runnable Interface

i. If we extend the Thread class, our class cannot extend any other class because Java doesn't support multiple inheritance. But, if we implement the Runnable interface, our class can still extend other base classes.

ii. We can achieve basic functionality of a thread by extending Thread class because it provides some inbuilt methods like yield(), interrupt() etc. that are not available in Runnable interface.

➡ Life Cycle of a Thread:-

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

i. New

ii. Ready/ Runnable

iii. Running

iv. Terminated

Synchronization

- Synchronized is a modifier which is applicable for methods & block but not for classes and variables .
- If multiple threads are trying to operate simultaneously on the same java object then there may be chance of data inconsistency problem .To overcome this problem we should go for synchronized keyword.
- So, if a method or block declared as synchronized then at a time only one thread is allowed to execute that method or block on the given object . So that data inconsistency problem will be resolved.

Synchronized Block

- If very few lines of the code required synchronization , then it is not recommended to declare entire method as synchronized i.e we have to enclosed those few lines of the code by using synchronized block .
- The benefit of synchronized block over synchronization method is that it reduces waiting time of threads and improves performance of the system

How to declare synchronized block ?

(i) Lock the current object

```
synchronized(this)
{
    ////////
}
```


(ii) Lock the particular object

```
synchronized(object )
{
    ////////
}
```

FAQ

- What is synchronized keyword and where we can apply?
- Explain advantage & disadvantage of synchronized keyword
- What is race condition ?

Inter Thread Communication

- Two threads can communicate with each other by using **wait() , notify()** and **notifyAll()** methods.
- We can call wait() , notify(), notifyAll() methods only from synchronized area , otherwise we will get runtime exception

Difference between notify() & notifyAll()

- We use notify() method to give the notification for only one waiting thread , if multiple threads are waiting then only one thread will be notify and remaining threads have to wait for further notification .
- We use notifyAll() to give notification for all waiting threads of a particular object

Deadlock

- If two threads are waiting for each other forever such type of infinite waiting is called deadlock .
- Because of synchronized keyword only the program will entered into deadlock suitable .If we are not using synchronized keyword then our program never entered into deadlock situation

How to stop a Thread ?

- Start a thread [t.start()]
- But if we call stop() method then immediately the thread will entered into dead state
- We can stop a thread execution by using stop() method of thread class


```
public void stop( )
```

How to suspend and resume of a Thread

- Thread class of java contain suspend() & resume() method .

public void suspend( )	public void resume( )

- We can suspend a thread by using suspend() method of Thread class then immediately the thread will be entered into suspended state .
- We can resume a suspended thread by using resume() method of Thread class then suspended thread can continue its execution .

Commonly used methods of Thread class:-

- public void run(): - is used to perform action for a thread.
- public void start(): - starts the execution of the thread. JVM calls the run() method on the thread .
- public void sleep(long miliseconds): -currently running thread to block for the specified number of milliseconds.
- public void join(): - The current thread invokes this method on a second thread, causing the current thread to block until the second thread terminates .
- public void join(long miliseconds):- The current thread invokes this method on a second thread, causing the current thread to block until the second thread terminates (after specified number of milliseconds).

- public void yield():- currently executing thread object temporarily pause and allow other threads to execute.
- public int getPriority(): - returns the priority of the thread.
- public int setPriority(int priority): - changes the priority of the thread.
- public void suspend():- is used to suspend the thread
- public void resume():-: is used to resume the suspended thread
- public void stop():-is used to stop the thread
- public void interrupt():-interrupts the thread.

Advantages of Multithreading

1. It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
2. You can perform many operations together, so it saves time.
3. Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

Note: At a time one thread is executed only.

THANK YOU