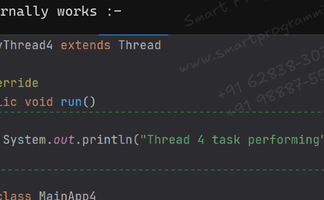
Multithreading

⇒ Process :
  → Process is the program which is executing or performing any task
  → Process is an independent unit because it has its own memory space, system resources etc
  → Examples :-
    = Web Browser
    = Text Editor
    = Media Player
      etc
  → How to achieve process :-
    = Process is created by operating system
    = We can create process programtically:-
      >> Runtime.getRuntime().exec();
      >> ProcessBuilder class

⇒ Thread :-
  → Thread is the smallest unit of process.
  → It is the sub-process which occupies same memory space as that of process.
  → Example :-
    = Tabs in browser
    = Sound, Progress bar etc thread in Media player



⇒ Multithreading :-
  → Multithreading is the way by which multiple threads interact with each other
    to perform a particular task or process.
  → Multithreading is the way by which we can achieve Multitasking

⇒ How to create threads :-
  → There are 2 ways to create threads in java :-
    1. Using Runnable Interface
    2. Using Thread class

```java
class MyThread3 implements Runnable
{
    @Override
    public void run()
    {
        System.out.println("Thread task executed");
    }
}
public class MainApp3
{
    public static void main(String[] args)
    {
        MyThread3 mt = new MyThread3();
        Thread thread = new Thread(mt);
        thread.start();
    }
}
```

```java
class MyThread4 extends Thread
{
    @Override
    public void run()
    {
        System.out.println("Thread 4 task performing");
    }
}
public class MainApp4
{
    public static void main(String[] args)
    {
        MyThread4 mt = new MyThread4();
        mt.start();
    }
}
```

→ Which is better way to create thread ?
  = using Runnable interface because we can also achieve multiple inheritance

⇒ How it internally works :-

```java
class MyThread4 extends Thread
{
    @Override
    public void run()
    {
        System.out.println("Thread 4 task performing");
    }
}
public class MainApp4
{
    public static void main(String[] args)
    {
        MyThread4 mt = new MyThread4();
        mt.start();
    }
}
```

thread-0

main thread

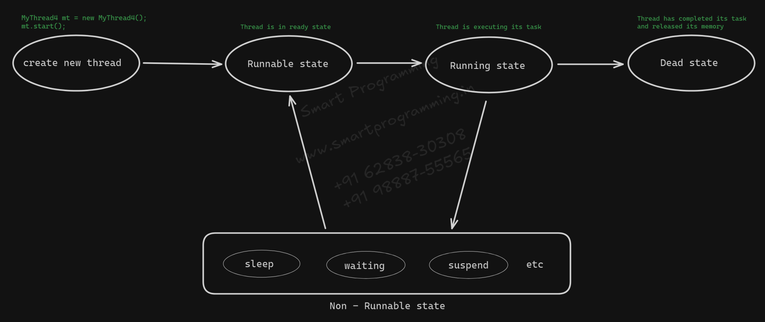
```java
class MyThread4 extends Thread
{
    @Override
    public void run()
    {
        for(int i=1; i<=5; i++)
        {
            System.out.println(i);
        }
    }
}
public class MainApp4
{
    public static void main(String[] args)
    {
        MyThread4 mt = new MyThread4();
        mt.start();

        for(int i=1; i<=5; i++)
        {
            System.out.println(i);
        }
    }
}
```

thread-0

main thread (JVM)

⇒ Thread Life Cycle :-




```java
class MyThread4 extends Thread
{
    @Override
    public void run()
    {
        System.out.println("Thread task executed");
    }
}
public class MainApp4
{
    public static void main(String[] args)
    {
        MyThread4 mt = new MyThread4();
        mt.start();
    }
}
```

⇒ Relationship Between Runnable & Thread class :-

```
interface Runnable
{
    run() { - }
}
class Thread implements Runnable
{
    start() { - }
    isAlive() { - }
    currentThread() { - }
    getName() { - }
    ----etc
}
```

⇒ Thread class Methods :-
  1. thread status
  2. naming a thread
  3. daemon thread

⇒ Daemon Thread :-
  → Thread which is executed in the background of another thread
  → It totally depends on the other thread

this is demo

this is dmo

main thread

spelling checker thread

Interview Questions

⇒ Multithreading Basics
1. What is multithreading in Java?
2. How is multithreading different from multiprocessing?
3. What are the advantages of using multithreading?
4. What are some challenges or drawbacks of multithreading?
5. How does multithreading improve performance in Java applications?

⇒ Thread Life Cycle
6. What are the different states of a thread in Java?
7. Can you explain the Thread life cycle from creation to termination?
8. In which state does a thread enter after calling start()?
9. What is the difference between new, runnable, and running states?
10. What happens if you call start() on a thread that has already finished execution?

⇒ Creating Threads
11. What are the two ways to create a thread in Java?
12. Which approach is better: extending Thread class or implementing Runnable interface? Why?
13. Can we directly call the run() method instead of start()? What will happen?
14. Why do we override the run() method when creating a thread?
15. Can a class implement both Runnable and extend another class at the same time?

⇒ Naming a Thread
16. How do you assign a name to a thread in Java?
17. What is the default name of a thread if we don't set it?
18. How can you retrieve the name of the currently executing thread?
19. Why might naming threads be useful in debugging or logging?

⇒ Daemon Threads
20. What is a daemon thread in Java?
21. What is the difference between a user thread and a daemon thread?
22. How do you create a daemon thread in Java?
23. Can we change a thread to daemon after it has started? Why or why not?
24. Give some examples of daemon threads in Java applications.
25. What happens when all user threads finish but daemon threads are still running?