BUZZ CHRONICLES > MULTITHREADING
Saved by @IamKishor_P
See On Twitter

Twitter Thread by Vikas Rajput

Vikas Rajput
@vikasrajputin

Java: Beginner Guide to Multithreading

a thread...

Multithreading is a concept of applying multitasking in Java.

Java supports thread-based multitasking.

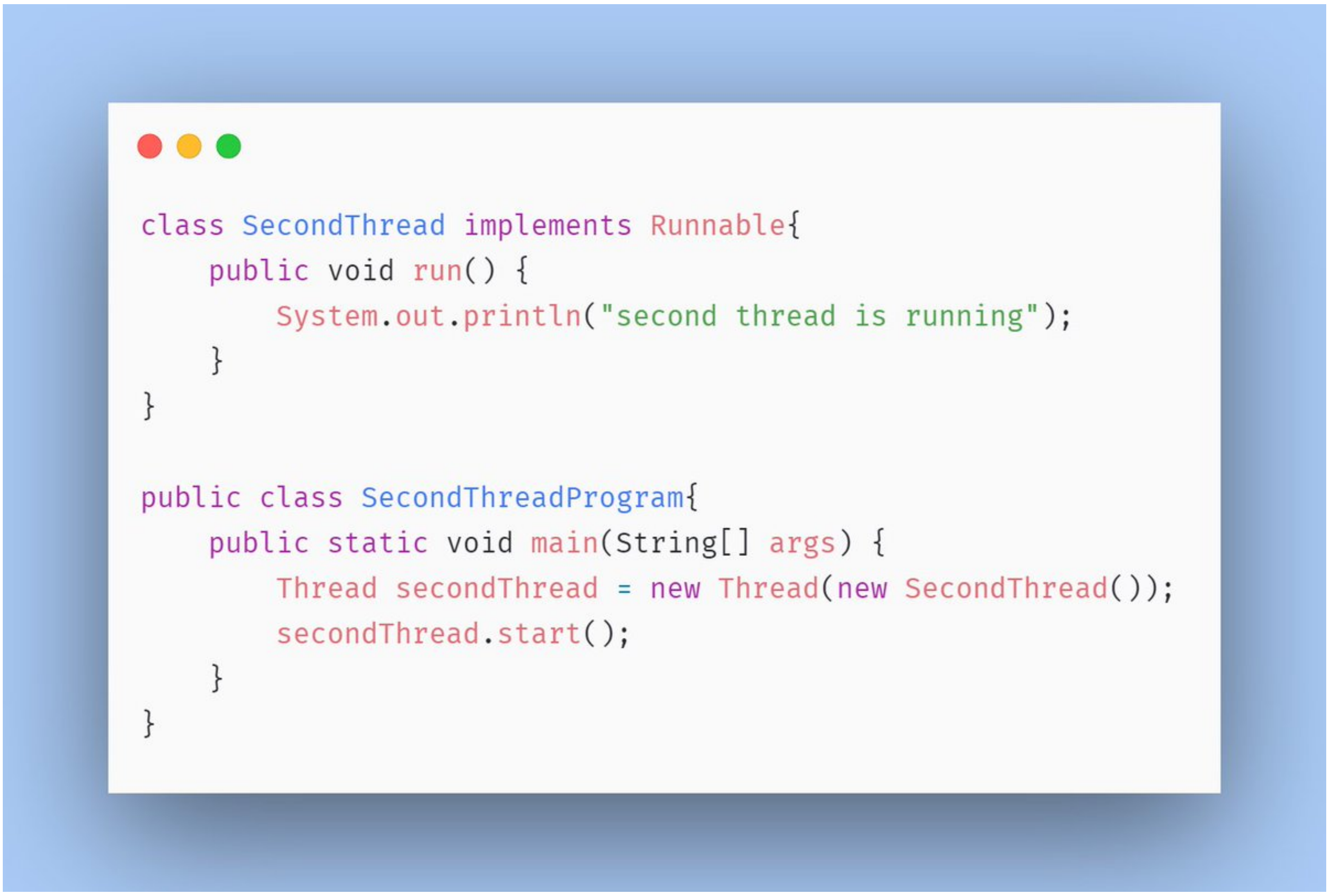
Java program can be divided into several threads and those threads can be executed in parallel to support multi-tasking.

Two ways to create a thread in Java:

1. By Implementing Runnable Interface
2. By extending the Thread class

Let's create a thread by implementing the "Runnable" interface:

Steps:
1. Implement Runnable Interface
2. Override the run() method, and put your code inside it.
3. Pass the instance of your class to the Thread class constructor.
4. Call the start() method to run your thread.

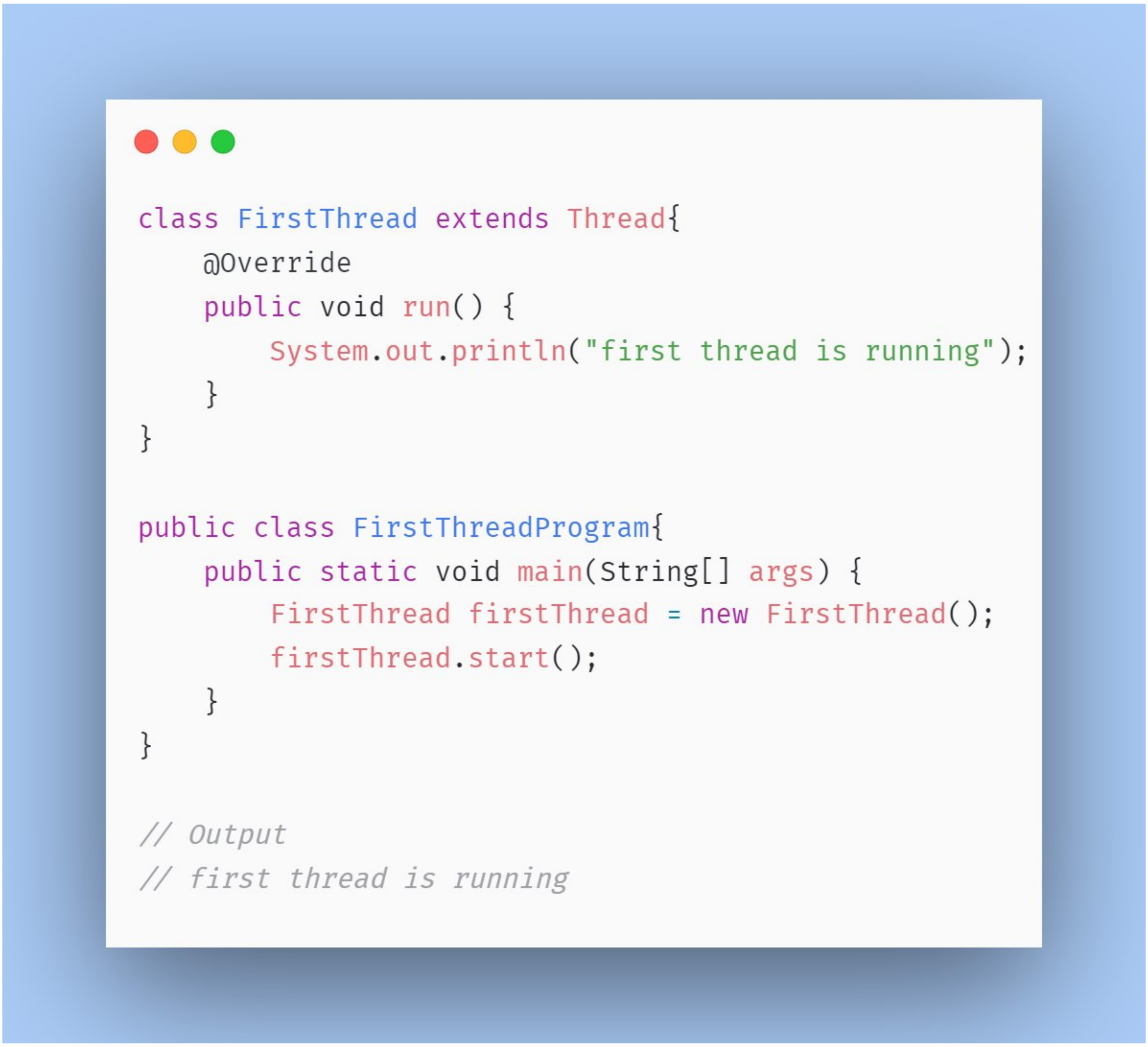


```java
class SecondThread implements Runnable{
    public void run() {
        System.out.println("second thread is running");
    }
}

public class SecondThreadProgram{
    public static void main(String[] args) {
        Thread secondThread = new Thread(new SecondThread());
        secondThread.start();
    }
}
```

Let's create a thread by extending the "Thread" class:

Steps:

1. Extend your class with Thread class
2. Override the run() method, and put your code inside it.
3. Instantiate your class.
4. Call the start() method to run your thread.



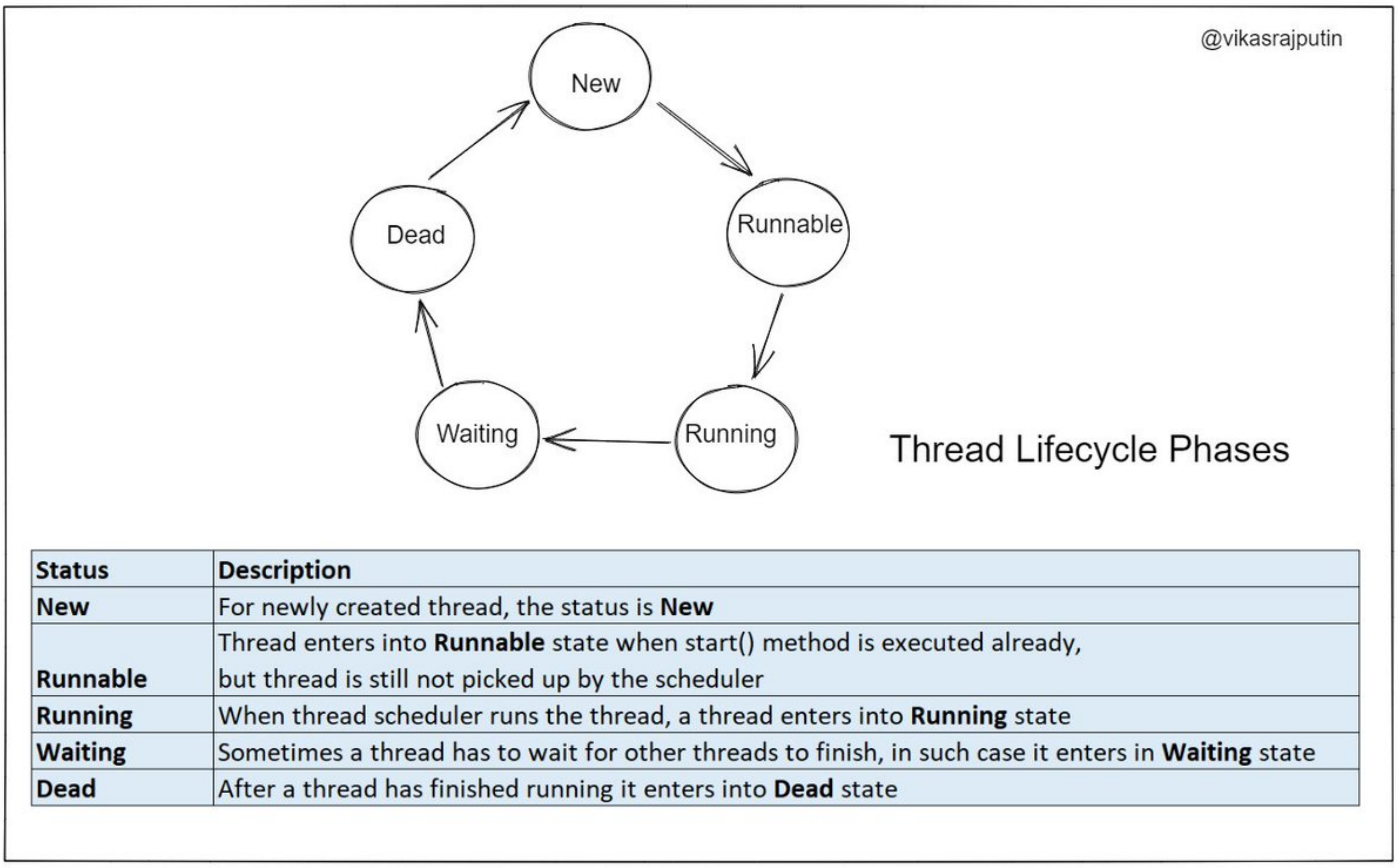
```java
class FirstThread extends Thread{
    @Override
    public void run() {
        System.out.println("first thread is running");
    }
}

public class FirstThreadProgram{
    public static void main(String[] args) {
        FirstThread firstThread = new FirstThread();
        firstThread.start();
    }
}

// Output
// first thread is running
```

Out of the above approaches, it is always advisable to use the first approach(By implementing a Runnable Interface) to create the threads in Java.

Because it's an Interface, you can also extend other classes in the future and also implement other interfaces.

Lifecycle of thread:

It has 5 different phases in its lifecycle:
1. New
2. Runnable
3. Running
4. Waiting
5. Dead



Thread Lifecycle Phases

Status	Description
New	For newly created thread, the status is **New**
Runnable	Thread enters into **Runnable** state when start() method is executed already, but thread is still not picked up by the scheduler
Running	When thread scheduler runs the thread, a thread enters into **Running** state
Waiting	Sometimes a thread has to wait for other threads to finish, in such case it enters in **Waiting** state
Dead	After a thread has finished running it enters into **Dead** state

Few Important Methods of Thread Class:

run() - Actual task of the thread is defined here.
start() - Starts the thread
join() - Wait for thread to die.
setName() - Give name to our thread.
getName() - Returns the thread name.

setPriority() - Sets the priority to thread.
getPriority() - Returns the priority.
getState() - Returns the state of thread.
isAlive() - checks if thread is alive or not

and a few more...

Conclusion:

We can use Multithreading in order to boost the performance of our program.

If our program has independent units, we can run those independent code blocks into a separate thread.

They can run parallel and process faster than normal.