Advanced Programming - ICT3112
Threads

Dinoo Gunasekera
Lecturer (Prob)
Department of ICT
Faculty of Technology
University of Ruhuna

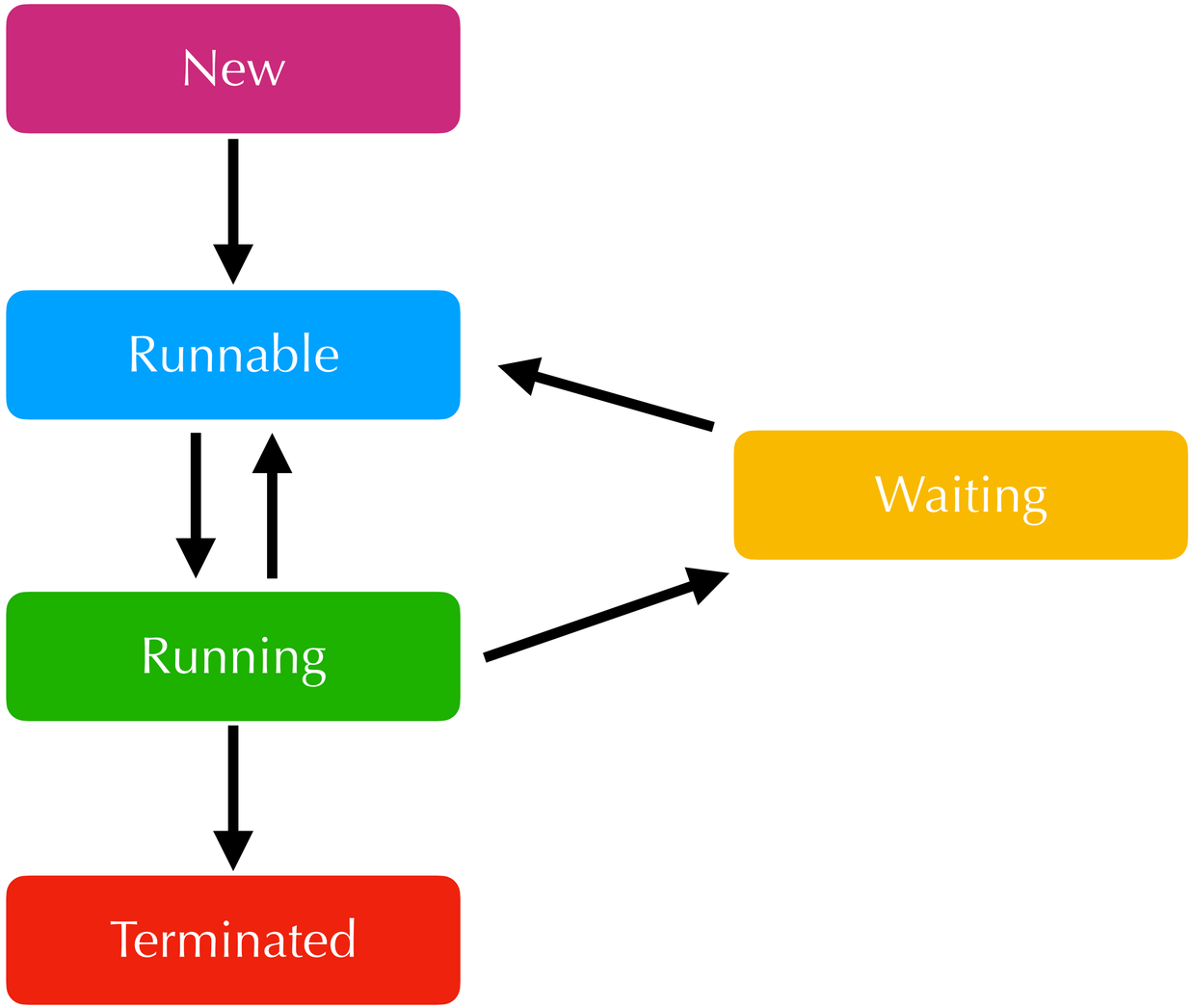
Outline

- What is a Java Thread?
- Thread Lifecycle
- Creating a Thread
- Main Thread
- Multi Threading
- Thread Synchronisation

What is a Java Thread?

- Thread is a lightweight sub process
- It is the smallest independent unit of a programme
- Contains a separate path of execution
- Every Java programme contains at least one thread
- A thread is created and controlled bu the java.lang.Thread class

Java Thread Lifecycle



Java Thread Lifecycle cont.

New

- A new thread begins its life cycle in this state and remains here until the programme starts the thread
- It is also known as a born thread

Runnable

- Once a newly born thread starts, the thread comes under runnable state. A thread stays in this state, until it is executing its task

Running

- A thread starts executing by entering run() method and the yield() method can send them to of back to the runnable state

Java Thread Lifecycle cont.

Waiting

- When a thread is temporarily in an inactive state i.e. it is still alive but is not eligible to run
- It can be in waiting, sleeping or blocked state

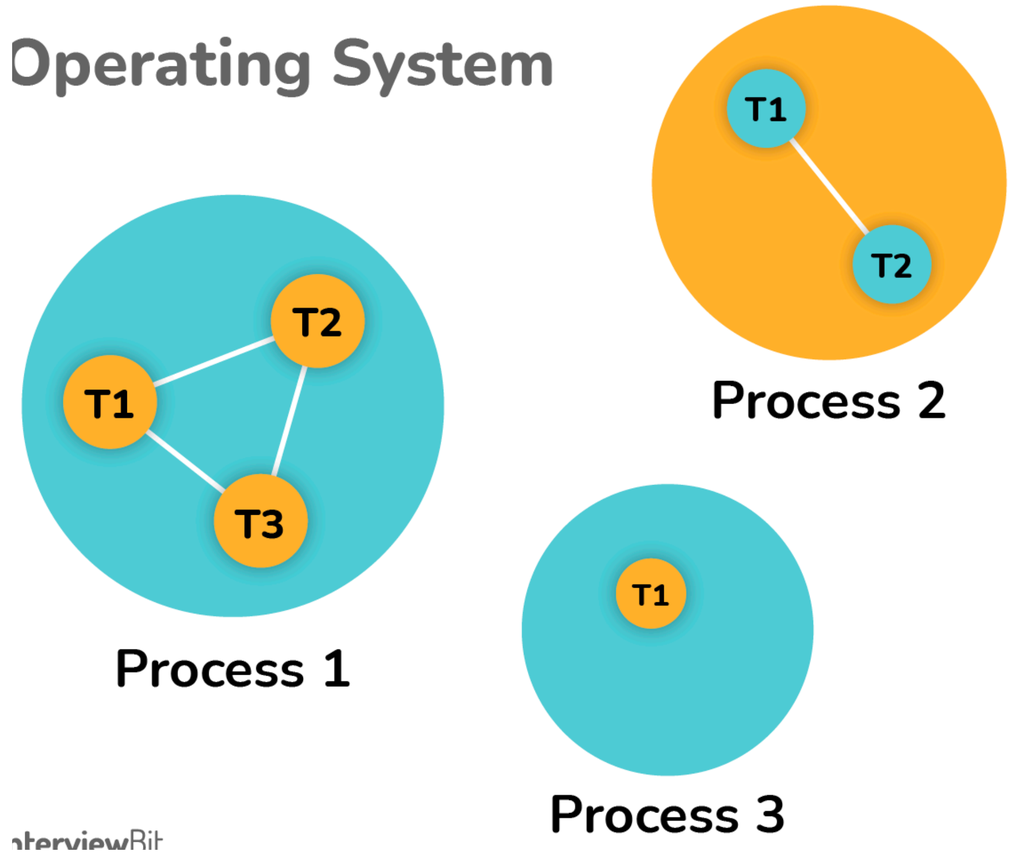
Terminated

- Runnable thread enters the terminated state when it completes its task or otherwise terminates

Thread vs Process

- Thread
  - Smallest units of the particular process
  - It has the ability to execute different parts of the program at the same time

- Process
  - A program that is in execution i.e., an active program
  - A process can be handled using Programme Control Block



Creating a Thread

Creating a Thread

A thread in Java can be created using two ways

Thread Class

```
public class Thread
      extends Object
            implements Runnable
```

Runnable Interface

```
public interface Runnable
```

Thread Class

Runnable Interface

1. Create a Thread class
2. Override run() method
3. Create object of the class
4. Invoke start() method to execute the custom thread run()

```java
public class MyThread extends Thread{
    public void run(){
        System.out.println("Thread is running");
    }

    public static void main(String[] args) {
        MyThread obj = new MyThread();
        obj.start();
    }
}
```

Thread Class

Runnable Interface

1. Create a Thread class implementing Runnable interface
2. Override run() method
3. Create object of the class
4. Invoke start() using the object

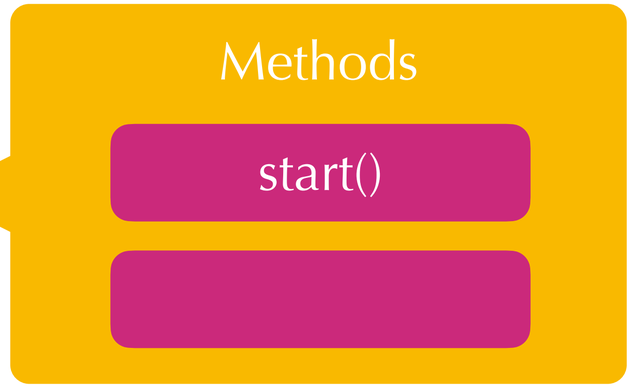
```
public class MyThread implements Runnable{
    public void run(){
        System.out.println("Thread is running");
    }

    public static void main(String[] args) {
        Thread obj = new Thread(new MyThread());
        obj.start();
    }
}
```

Thread Class vs Runnable Interface

Thread Class	Runnable Interface
Each thread creates its unique object	Each thread creates its unique object
More memory consumption	More memory consumption
A class extending Thread class can't extend any other class	Along with runnable a class can implement any other interface
Thread class is extended only if there is a need of overriding other methods of it	Runnable is implanted only if there is a need of special run method
Enables tight coupling	Enables loose coupling

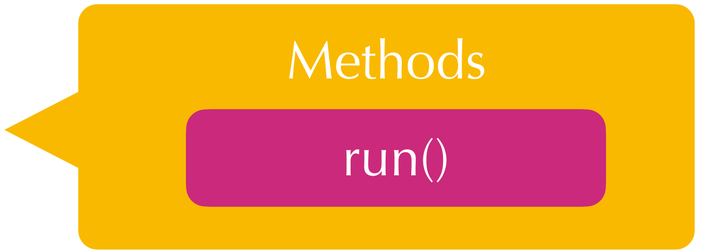
Class Hierarchy

- Object
  - Boolean
  - Character
  - Math
  - Number
  - Thread
  - …



Interface Hierarchy

- Appendable
- AutoCloseable
- CharSequence
- Cloneable
- Comparable
- Interable
- Readable
- Runnable

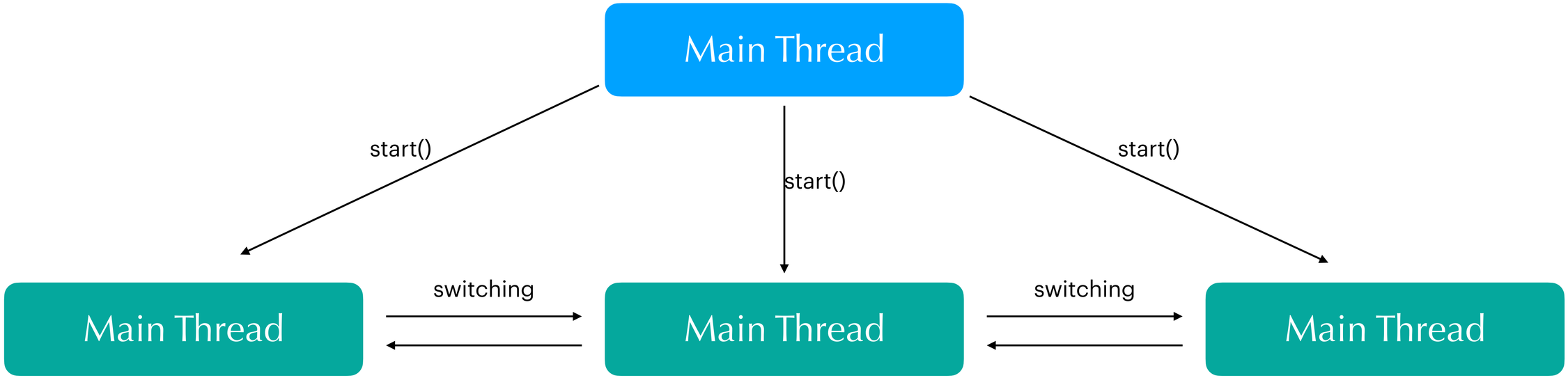


Java Main Thread

- The most important thread of a Java programme
- It is executed whenever a Java programme starts
- Every programme must contain this thread for its execution to take place
- Java main thread is needed because:
  - From this other ‘child’ elements are spawned
  - It must be the last thread to finish execution i.e. when the main thread stops programme terminates

Multi Threading in Java

- Multi threading is the ability of a programme to run two or more threads concurrently, where each thread can handle a different task at the same time making optimal use of the available resources



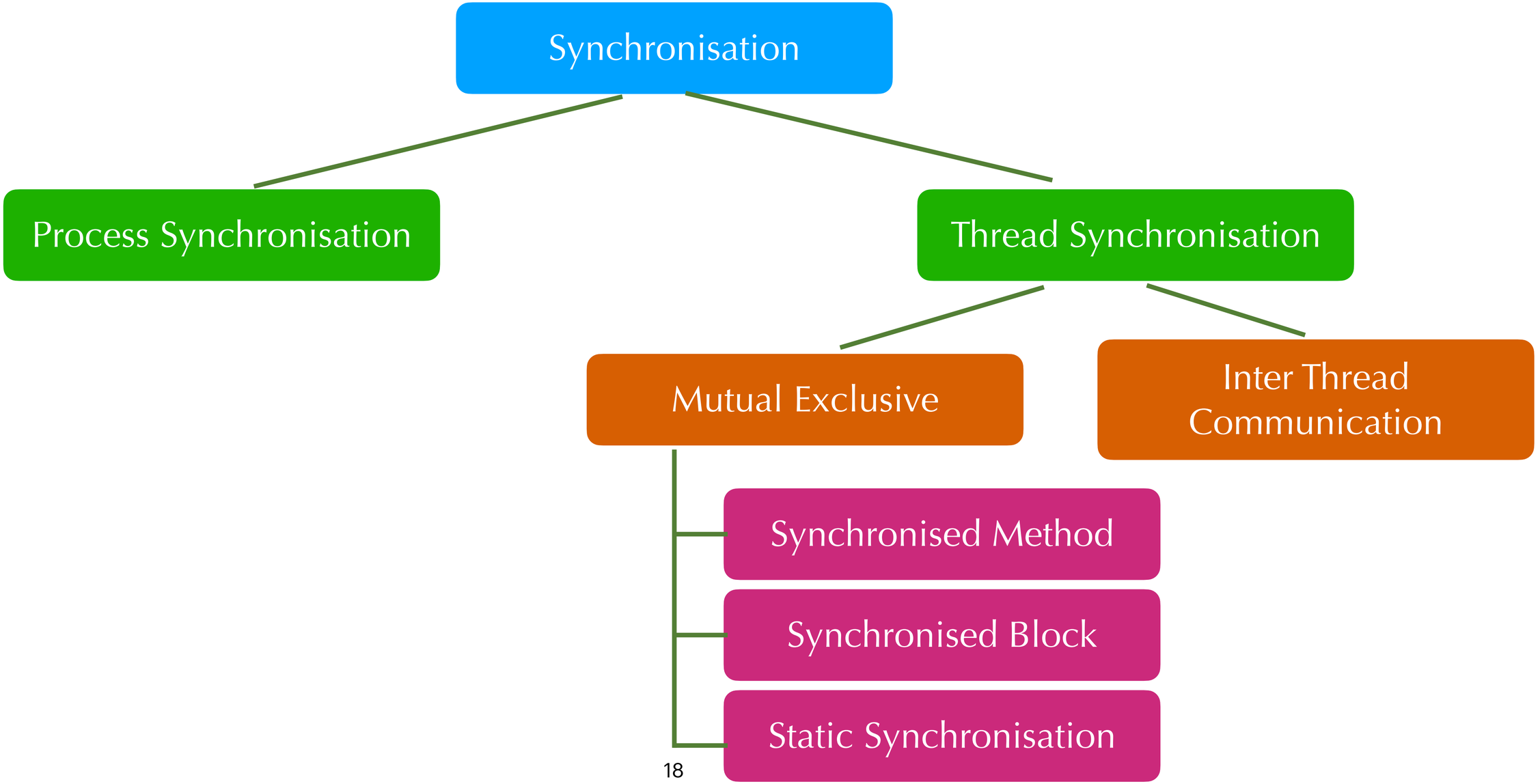
Multitasking vs Multithreading

- Multitasking
  - Computer's ability to perform multiple jobs concurrently
    - More than one programme are running concurrently
- Multithreading
  - Multiple threads to control within a single programme
    - Each programme can run multiple threads to control within it

Synchronisation

- Capability to control the access of multiple threads to any shared resources
- In multi threading, multiple threads try to access the shared resources at a time to produce inconsistent results
- Synchronisation is necessary for reliable communication between threads
- Synchronisations helps:
  - In preventing thread interference
  - To prevent concurrency problems

Synchronisation Methods



Exercises

Tread Creation and Management:

- Write a Java program to create a thread by extending the Thread class. The thread should print numbers from 1 to 10 with a delay of 500 milliseconds between each number.

Runnable Interface

- Create a Java program to implement the Runnable interface. Define a class that implements the Runnable interface to print even numbers from 2 to 20 with a delay of 300 milliseconds between each number.

Thread Synchronisation

- Write a Java program that demonstrates thread synchronization using synchronized methods. Define a class with a synchronized method that prints odd numbers from 1 to 10, and another synchronized method that prints even numbers from 2 to 10. Ensure that the odd and even numbers are printed in sequence.

Inter-Thread Communication

- Implement a Java program to demonstrate inter-thread communication using wait() and notify() methods. Define two threads, one to produce numbers from 1 to 5 and another to consume these numbers and print them. Use wait() and notify() to ensure that the consumer thread waits until the producer thread produces a number.

Thread States and Life-Cycle

- Write a Java program to illustrate the different states in the life cycle of a thread. Create a thread that goes through each state (New, Runnable, Running, Waiting/Blocked, Dead) and prints its current state before transitioning to the next state.