Print "Good morning" and "Welcome" Continuously Using Threads:

public class ContinuousPrinting {

    public static void main(String[] args) {

        Thread morningThread = new Thread(() -> {

            while (true) {

                System.out.println("Good morning");

                try {

                    Thread.sleep(1000);

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        });

        Thread welcomeThread = new Thread(() -> {

            while (true) {

                System.out.println("Welcome");

                try {

                    Thread.sleep(1000);

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        });

        morningThread.start();

        welcomeThread.start();

    }

}

**Output**

Good morning

Welcome

Welcome

Good morning

Welcome

Good morning

Welcome

Good morning

Welcome

Good morning

Welcome

Add Step Method to Delay Execution of Welcome Thread:

public class WelcomeThreadWithDelay {

    public static void main(String[] args) {

        Thread welcomeThread = new Thread(() -> {

            while (true) {

                System.out.println("Welcome");

                try {

                    Thread.sleep(200); // Delay execution for 200ms

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        });

        welcomeThread.start();

    }

}

**Output**

Welcome

Welcome

Welcome

Welcome

Demonstrate getPriority() and setPriority() Methods:

public class PriorityExample {

    public static void main(String[] args) {

        Thread thread1 = new Thread(() -> {

            System.out.println("Thread 1 priority: " + Thread.currentThread().getPriority());

        });

        Thread thread2 = new Thread(() -> {

            System.out.println("Thread 2 priority: " + Thread.currentThread().getPriority());

        });

        thread1.setPriority(Thread.MIN\_PRIORITY);

        thread2.setPriority(Thread.MAX\_PRIORITY);

        thread1.start();

        thread2.start();

    }

}

**Output**

Thread 2 priority: 10

Thread 1 priority: 1

Get State of a Given Thread:

public class ThreadStateExample {

    public static void main(String[] args) {

        Thread thread = new Thread(() -> {

            System.out.println("Thread state: " + Thread.currentThread().getState());

        });

        thread.start();

    }

}

**Output**

Thread state: RUNNABLE

Get Reference to Current Thread:

public class CurrentThreadExample {

    public static void main(String[] args) {

        Thread currentThread = Thread.currentThread();

        System.out.println("Current thread: " + currentThread);

    }

}

**Output**

Current thread: Thread[#1,main,5,main]

Create and Start Multiple Threads Incrementing a Shared Counter:

public class SharedCounterThreads {

    private static int counter = 0;

    public static void main(String[] args) {

        for (int i = 0; i < 5; i++) {

            Thread thread = new Thread(() -> {

                for (int j = 0; j < 1000; j++) {

                    incrementCounter();

                }

            });

            thread.start();

        }

        try {

            Thread.sleep(2000); // Wait for threads to finish

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

        System.out.println("Final counter value: " + counter);

    }

    private synchronized static void incrementCounter() {

        counter++;

    }

}

**Output**

Final counter value: 5000

Producer-Consumer Scenario Using wait() and notify():