(LAB\_7) ASSESMENT

# Write a program that creates two threads. Each Thread should pript its thread ID (TID) and a unique message to the console. Ensure that the output from both threads is interleaved.

**package** Lab.com;

**public class** Thread **implements** Runnable {

**private** String message; **public** Hari(String message) { **this**.message = message;

}

**public void** run() {

**for** (**int** i = 0; i < 5; i++) { System.***out***.println(Thread.*currentThread*().getId() + ": " + message); **try** {

Thread.*sleep*(100); // Optional delay to increase interleaving chances

} **catch** (Exception e) { System.***out***.println(e);

}

}

}

}

**package** Lab.com;

**public class** Interleaved {

**public static void** main(String[] args) {

Thread thread1 = **new** Thread(**new** Thread("Thread 1")); Thread thread2 = **new** Thread(**new** Thread("Thread 2")); thread1.start();

thread2.start();

}

}

# OUTPUT:

16(TID): Thread 2

15(TID): Thread 1

16(TID): Thread 2

15(TID): Thread 1

16(TID): Thread 2

15(TID): Thread 1

16(TID): Thread 2

15(TID): Thread 1

15(TID): Thread 1

16(TID): Thread 2

# Write a program that creates multiple threads with different priorities. Observe how the operating system schedules threads with different priorities and explain the results.

**package** Lab.com;

**public class** Mercy **implements** Runnable{

**public void** run() {

**for** (**int** i = 0; i < 5; i++) { System.***out***.println(Thread.*currentThread*().getName() + ": Priority "

+ Thread.*currentThread*().getPriority() + ", Count: " + i);

**try** { Thread.*sleep*(100);

} **catch** (InterruptedException e) { e.printStackTrace();

}

}

}

}

**package** Lab.com;

**public class** Priority {

**public static void** main(String[] args) {

Thread Thread1 = **new** Thread(**new** Mercy(), "Low Priority Thread"); Thread Thread2 = **new** Thread(**new** Mercy(), "Normal Priority Thread"); Thread Thread3 = **new** Thread(**new** Mercy(), "High Priority Thread");

// Set thread priorities

Thread1.setPriority(Thread.***MIN\_PRIORITY***); Thread2.setPriority(Thread.***NORM\_PRIORITY***); Thread3.setPriority(Thread.***MAX\_PRIORITY***); Thread1.start();

Thread2.start(); Thread3.start();

}

}

# OUTPUT:

Low Priority Thread: Priority 1, Count: 0 High Priority Thread: Priority 10, Count: 0 Normal Priority Thread: Priority 5, Count: 0 Low Priority Thread: Priority 1, Count: 1 High Priority Thread: Priority 10, Count: 1 Normal Priority Thread: Priority 5, Count: 1 High Priority Thread: Priority 10, Count: 2 Normal Priority Thread: Priority 5, Count: 2 Low Priority Thread: Priority 1, Count: 2 High Priority Thread: Priority 10, Count: 3 Normal Priority Thread: Priority 5, Count: 3 Low Priority Thread: Priority 1, Count: 3 High Priority Thread: Priority 10, Count: 4 Low Priority Thread: Priority 1, Count: 4

Normal Priority Thread: Priority 5, Count: 4

# Write a Java program that creates two threads and prints "Thread A" from the first thread and "Thread B" from the second thread. Make sure both threads run concurrently.

**package** Lab.com;

**public class** Mercy **implements** Runnable{

**private** String message;

**public** Mercy(String message) {

**this**.message = message;

}

**public void** run() {

**for** (**int** i = 0; i < 5; i++) { System.***out***.println(message); **try** {

Thread.*sleep*(100); // Optional delay to increase interleaving chances

} **catch** (Exception e) { System.***out***.println(e);

}

}

}

}

**package** Lab.com;

**public class** Mercyy {

**public static void** main(String[] args) {

Thread threadA = **new** Thread(**new** Mercy("Thread A")); Thread threadB = **new** Thread(**new** Mercy("Thread B")); threadA.start();

threadB.start();

}

}

# OUTPUT:

Thread B Thread A Thread A Thread B Thread A Thread B Thread A Thread B Thread A Thread B