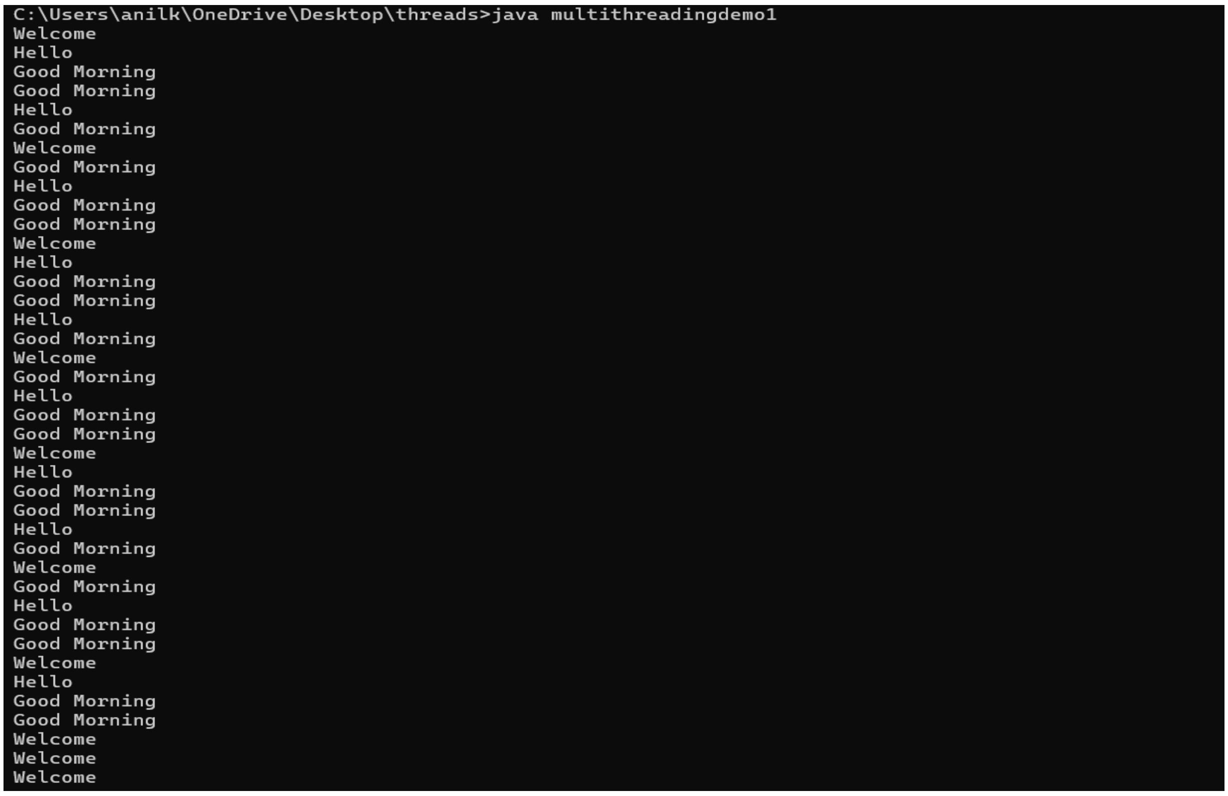
Experiment - 7

a) Develop a JAVA program that creates threads by **extending Thread class**. First thread display "Good Morning" every 1 sec, the second thread displays "Hello" every 2 seconds and the third thread display "Welcome" every 3 seconds.

Source Code

```
class thread1 extends Thread
{
    public void run()
    {
             for(int i=0;i<10;i++)
            {
               try
               {
                    System.out.println("Good Morning");
                    Thread.sleep(1000);
               }

               catch(InterruptedException e)
               {
                   System.out.println("Thread is in sleep state  - cant be interrupted");
               }
           }
    }
}
class thread2 extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try
            {

                System.out.println("Hello");
                Thread.sleep(2000);  // 2 second
            }
            catch (InterruptedException e)
            {
                 System.out.println("Thread is in sleep state  - cant be interrupted");
            }
        }
    }
}
class thread3 extends Thread
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try
            {
                 System.out.println("Welcome");
                Thread.sleep(3000);  // 3 second
            }
            catch (InterruptedException e)
            {
                 System.out.println("Thread is in sleep state  - cant be interrupted");
            }
        }
    }
}
class multithreadingdemo1
{
    public static void main(String[] args)
    {
        // Creating threads
        thread1 t1 = new thread1();
        thread2 t2 = new thread2();
        thread3 t3 = new thread3();
       //starting threads
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output



```
C:\Users\anilk\OneDrive\Desktop\threads>java multithreadingdemo1
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Welcome
Welcome
Welcome
```

b) Develop a JAVA program that creates threads by **implementing Runnable interface**. First thread display “Good Morning” every 1 sec, the second thread displays “Hello” every 2 seconds and the third thread display “Welcome” every 3 seconds.

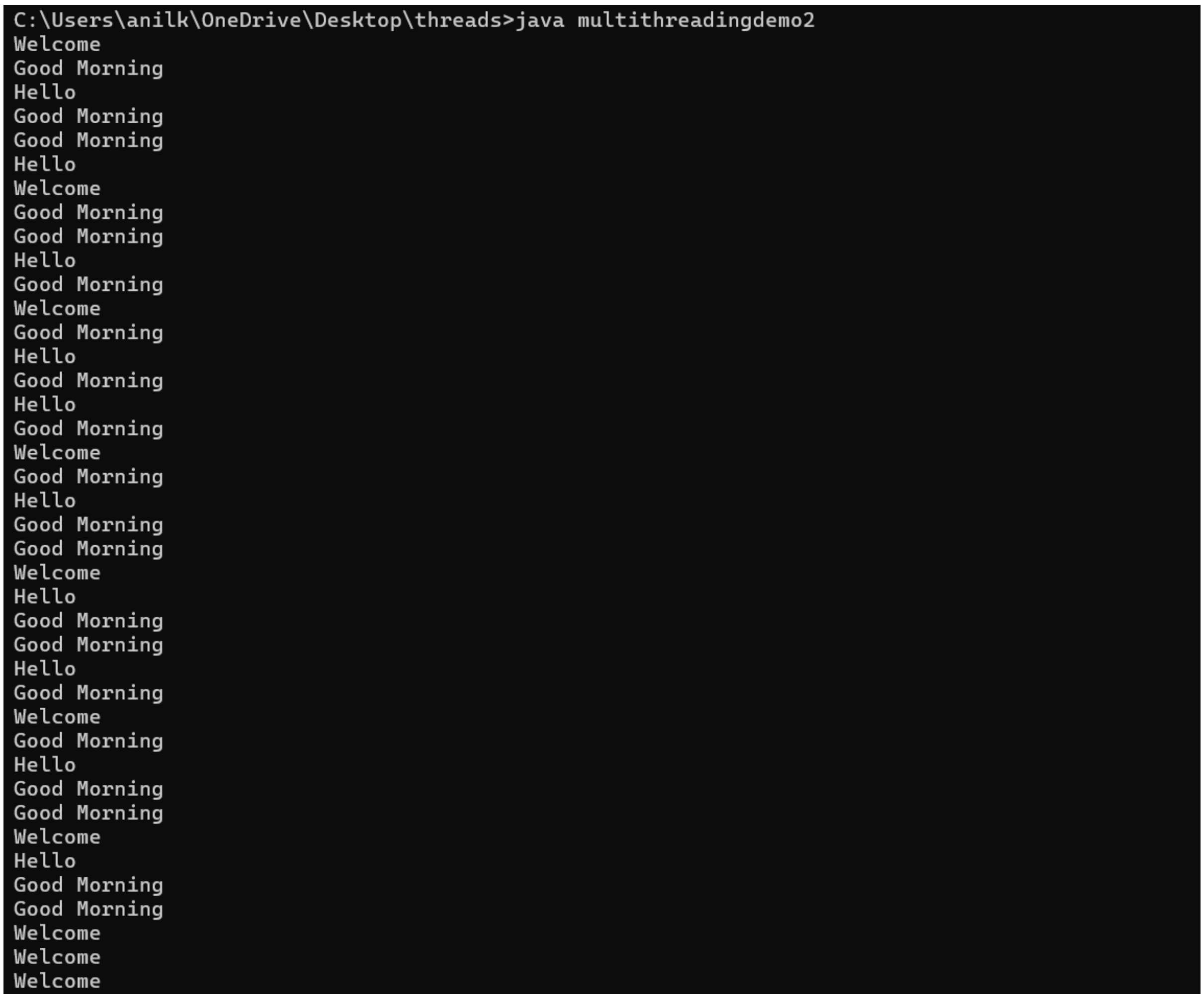
Source Code

```
class thread1 implements Runnable
{
   public void run()
```

```
    {

            for(int i=0;i<10;i++)
            {
               try
               {
                    System.out.println("Good Morning");
                    Thread.sleep(1000);
               }
               catch(InterruptedException e)
               {
                   System.out.println("Thread is in sleep state  - cant be interrupted");
               }
           }
    }
}
class thread2 implements Runnable
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try
            {
                System.out.println("Hello");
                Thread.sleep(2000);  // 2 second
            }
            catch (InterruptedException e)
            {
                 System.out.println("Thread is in sleep state  - cant be interrupted");
            }
        }
    }
}
```

```
class thread3 implements Runnable
{
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            try
            {

                System.out.println("Welcome");
                Thread.sleep(3000);  // 3 second
            }
            catch (InterruptedException e)
            {
                 System.out.println("Thread is in sleep state  - cant be interrupted");
            }
        }
    }
}
class multithreadingdemo2
{
    public static void main(String[] args)
    {
        // Creating threads
        thread1 t1 = new thread1();
        thread2 t2 = new thread2();
        thread3 t3 = new thread3();

       Thread one = new Thread(t1);
       Thread two=new Thread(t2);
       Thread three=new Thread(t3);

        //starting threads
        one.start();
        two.start();
        three.start();


    }
}
```

Output:



```
C:\Users\anilk\OneDrive\Desktop\threads>java multithreadingdemo2
Welcome
Good Morning
Hello
Good Morning
Good Morning
Hello
Welcome
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Welcome
Welcome
Welcome
```

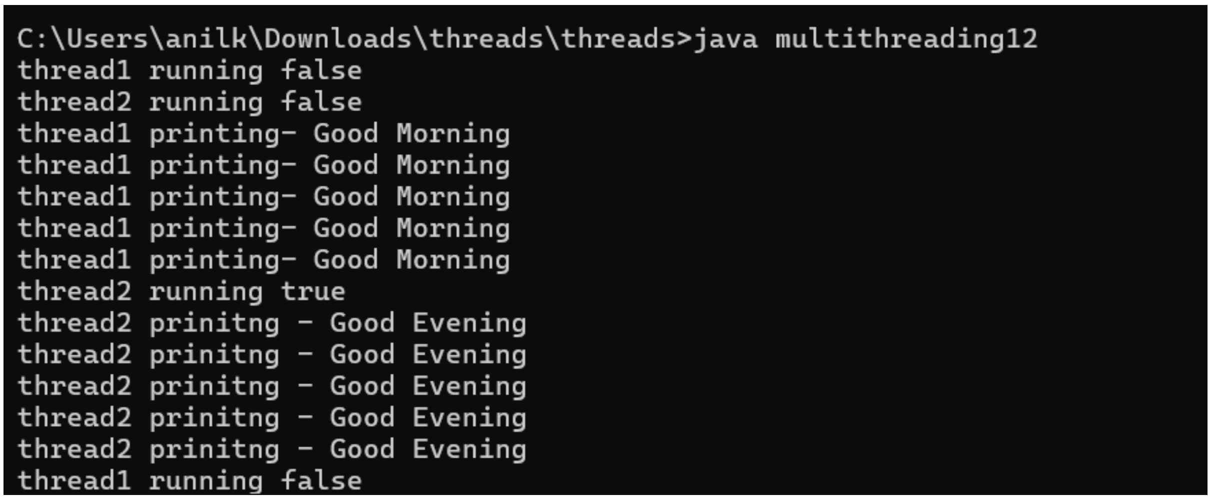
C. Develop a program on isAlive() and join().

Source Code

```
class thread1 extends Thread
{
    public void run()
    {
        for(int i=0;i<5;i++)
        {
            System.out.println("thread1 printing- Good Morning");
        }
    }
}

class thread2 extends Thread
{
    public void run()
    {
        for(int i=0;i<5;i++)
        {
            System.out.println("thread2 prinitng - Good Evening");
        }
    }
}


public class multithreading12
{
    public static void main(String[] args) throws InterruptedException
    {
        thread1 t1=new thread1(); //thread created - new born state
        System.out.println("thread1 running "+t1.isAlive());    //returns false since t1 not yet
been in running state

        thread2 t2=new thread2(); //thread created - new born state
        System.out.println("thread2 running "+t2.isAlive());    //returns false since t1 not yet
been in running state


        t1.start();       //t1 execution started(running)
        t1.join();        //all other threads are in waiting stage until t1 finished its execution
        t2.start();       //t2 waits until t1 completes its execution
        System.out.println("thread2 running "+t2.isAlive());
        System.out.println("thread1 running "+t1.isAlive());
          //returns false since t1 has been terminated

    }
```

}

Output



```
C:\Users\anilk\Downloads\threads\threads>java multithreading12
thread1 running false
thread2 running false
thread1 printing- Good Morning
thread1 printing- Good Morning
thread1 printing- Good Morning
thread1 printing- Good Morning
thread1 printing- Good Morning
thread2 running true
thread2 prinitng - Good Evening
thread2 prinitng - Good Evening
thread2 prinitng - Good Evening
thread2 prinitng - Good Evening
thread2 prinitng - Good Evening
thread1 running false
```

D. Demonstrate Producer Consumer Problem.

Source Code

```
import java.util.LinkedList;

public class Threadexample {
	public static void main(String[] args)
		throws InterruptedException
	{
		// Object of a class that has both produce()
		// and consume() methods
		final PC pc = new PC();

		// Create producer thread
		Thread t1 = new Thread(new Runnable() {
			@Override
			public void run()
			{
				try {
					pc.produce();
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		});

		// Create consumer thread
		Thread t2 = new Thread(new Runnable() {
			@Override
			public void run()
			{
				try {
					pc.consume();
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		});

		// Start both threads
		t1.start();
		t2.start();

		// t1 finishes before t2
```

```
        t1.join();
        t2.join();
    }

    // This class has a list, producer (adds items to list
    // and consumer (removes items).
    public static class PC {

        // Create a list shared by producer and consumer
        // Size of list is 2.
        LinkedList<Integer> list = new LinkedList<>();
        int capacity = 2;

        // Function called by producer thread
        public void produce() throws InterruptedException
        {
            int value = 0;
            while (true) {
                synchronized (this)
                {
                    // producer thread waits while list
                    // is full
                    while (list.size() == capacity)
                        wait();

                    System.out.println("Producer produced-"
                                        + value);

                    // to insert the jobs in the list
                    list.add(value++);

                    // notifies the consumer thread that
                    // now it can start consuming
                    notify();

                    // makes the working of program easier
                    // to understand
                    Thread.sleep(1000);
                }
            }
        }

        // Function called by consumer thread
        public void consume() throws InterruptedException
        {
            while (true) {
                synchronized (this)
                {
                    // consumer thread waits while list
                    // is empty
                    while (list.size() == 0)
                        wait();

                    // to retrieve the first job in the list
                    int val = list.removeFirst();

                    System.out.println("Consumer consumed-"
                                                    + val);

                    // Wake up producer thread
                    notify();

                    // and sleep
                    Thread.sleep(1000);
                }
            }
        }
    }
}
```

OUTPUT:

```
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
```

E. Demonstrate Demon Thread

Source Code

```
public class DaemonThreadExample {
    public static void main(String[] args) {
        Thread daemonThread = new Thread(() -> {
            while (true) {
                System.out.println("Daemon thread running...");
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    System.out.println("Daemon thread interrupted");
                }
            }
        });

        // Set the thread as a daemon thread
        daemonThread.setDaemon(true);

        // Start the daemon thread
        daemonThread.start();

        // Create and start a user thread
        Thread userThread = new Thread(() -> {
            System.out.println("User thread running...");
            try {
                Thread.sleep(2000);  // User thread will run for 2 seconds
            } catch (InterruptedException e) {
                System.out.println("User thread interrupted");
            }
            System.out.println("User thread completed.");
        });

        userThread.start();

        // Main thread will wait for userThread to finish
        try {
            userThread.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted");
        }

        System.out.println("Main thread completed.");
    }
}
```

<u>Output</u>

```
User thread running...
Daemon thread running...
Daemon thread running...
Daemon thread running...
Daemon thread running...
User thread completed.
Main thread completed.
```