ASSIGNMENT – 8

1. Write a java program that will create the reference of the thread and display the details of the reference thread. (It should display class, name of thread, priority, group name)

```
class ThreadData extends Thread {
	public void run() {
		System.out.println("Class: "+getClass());
		System.out.println("Name: "+getName());
		System.out.println("Priority: "+getPriority());
		System.out.println("Thread Group: "+getThreadGroup().getName());
	}
}

class ThreadDetails {
	public static void main(String args[]) {
		ThreadData t = new ThreadData();
		t.start();
	}
}
```

```
C:\24bcsa08\Assignment_8\q1>javac ThreadDetails.java

C:\24bcsa08\Assignment_8\q1>java ThreadDetails
Class: class ThreadData
Name: Thread-0
Priority: 5
Thread Group: main
```

2. Write a java program that will create a thread and set the thread name, display the thread name, get the thread id, check the thread is currently alive or not.

```
class MyThread extends Thread {
	public void run() {
		Thread t = Thread.currentThread();
		System.out.println("The new name of the thread: "+t.getName());
		System.out.println("Id: "+t.getId());
		System.out.println("Is Alive: " +t.isAlive());
	}
}

class SetThreadName {
	public static void main(String args[]) {
		MyThread t1 = new MyThread();
		t1.setName("Special Thread");
		t1.start();
	}
}
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
C:\24bcsa08\Assignment_8\q2>javac SetThreadName.java
Note: SetThreadName.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\24bcsa08\Assignment_8\q2>java SetThreadName
The new name of the thread: Strong Thread
Id: 26
Is Alive: true
```

3. Write a java program that will create one thread [using Runnable interface]. The main thread will read a number and the newly created thread checks the number is Armstrong number or not.

```
import java.util.Scanner;

class ArmstrongCheck implements Runnable {
	int n;
	ArmstrongCheck(int n) {
		this.n = n;
	}

	public void run() {
		int temp = n;
		int sum = 0, rem;

		while (temp != 0) {
			rem = temp % 10;
			sum += rem * rem * rem;
			temp /= 10;
		}

		if (sum == n)
			System.out.println(n + " is an Armstrong number.");
		else
			System.out.println(n + " is not an Armstrong number.");
	}
}

class ThreadArmstrong {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter the number: ");
		int n = sc.nextInt();
		ArmstrongCheck a = new ArmstrongCheck(n);
		Thread t = new Thread(a);
		t.start();
	}
}
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
C:\24bcsa08\Assignment_8\q3>javac ThreadArmstrong.java

C:\24bcsa08\Assignment_8\q3>java ThreadArmstrong
Enter the number: 371
371 is an Armstrong number.
```

4. Write a java program that will create one thread [using extends]. The main thread will read a number and check the number is prime or composite and the same time the new thread will check the number palindrome or not.

```
import java.util.Scanner;

class CheckPalindrome extends Thread {
	int n;

	CheckPalindrome(int n) {
		this.n = n;
	}

	public void run() {
		int temp = n;
		int rem, rev = 0;

		while (temp != 0) {
			rem = temp % 10;
			rev = rev * 10 + rem;
			temp = temp / 10;
		}

		if (rev == n)
			System.out.println(n + " is a palindrome number.");
		else
			System.out.println(n + " is not a palindrome number.");
	}
}

class PrimeCheck {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter the number: ");
		int n = sc.nextInt();

		CheckPalindrome t = new CheckPalindrome(n);
		t.start();

		boolean isPrime = true;

		if (n <= 1)
		isPrime = false;
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
            else {
                for (int i = 2; i <= n / 2; i++) {
                    if (n % i == 0) {
                        isPrime = false;
                        break;
                    }
                }
            }

            if (isPrime)
                System.out.println(n + " is a prime number.");
            else
                System.out.println(n + " is not a prime number.");
        }
}
```

```
C:\24bcsa08\Assignment_8\q4>javac PrimeCheck.java

C:\24bcsa08\Assignment_8\q4>java PrimeCheck
Enter the number: 7
7 is a prime number.
7 is a palindrome number.
```

5. Write a java program that will create one child thread. The child thread has to display all odd numbers between m and n, and the main thread will display all the even numbers between m and n.

```
import java.util.Scanner;

class OddThread extends Thread {
    int m, n;
    OddThread(int m, int n) {
        this.m = m;
        this.n = n;
    }

    public void run() {
        for (int i = m; i <= n; i++) {
        if (i % 2 != 0)
            System.out.print(i + " ");
        }
    }
}

class OddEvenThreads {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter m and n: ");
        int m = sc.nextInt(), n = sc.nextInt();
```

```
            OddThread t = new OddThread(m, n);
            t.start();

            for (int i = m; i <= n; i++) {
                if (i % 2 == 0)
                    System.out.print(i + " ");
            }
        }
}
```

```
C:\24bcsa08\Assignment_8\q5>javac OddEvenThreads.java

C:\24bcsa08\Assignment_8\q5>java OddEvenThreads
Enter m and n: 5 20
6 5 8 7 9 10 11 12 13 14 16 18 15 20 17 19
```

6. Write a java program to create two threads. First thread should find the square of the number, second thread should find the sum of the digits of the number.

```
import java.util.Scanner;

class SquareThread extends Thread {
    int num;
    SquareThread(int n) {
        num = n;
    }

    public void run() {
        System.out.println("Square: " + (num * num));
    }
}

class SumThread extends Thread {
    int num;
    SumThread(int n) {
        num = n;
    }

    public void run() {
        int sum = 0, temp = num;
        while (temp != 0) {
            sum += temp % 10;
            temp /= 10;
        }
        System.out.println("Sum of digits: " + sum);
    }
}

class TwoThreads {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
            System.out.print("Enter the number: ");
            int n = sc.nextInt();
            SquareThread s1 = new SquareThread(n);
            SumThread s2 = new SumThread(n);
            s1.start();
            s2.start();
        }
}
```

```
C:\24bcsa08\Assignment_8\q6>javac TwoThreads.java

C:\24bcsa08\Assignment_8\q6>java TwoThreads
Enter the number: 10
Square: 100
Sum of digits: 1
```

7. Write a java program that will create two threads. The main thread will read a number and one thread will print the multiplication table of the entered number and the same time the other thread will find the factorial of the entered number.
Note: Main thread has to wait until other two threads have completed their task.

```
import java.util.Scanner;

class TableThread extends Thread {
        int num;
        TableThread(int n) {
                num = n;
        }

        public void run() {
        for (int i = 1; i <= 10; i++) {
                System.out.println(num + " x " + i + " = " + (num * i));
                }
        }
}

class FactorialThread extends Thread {
        int num;
        FactorialThread(int n) {
                num = n;
        }

        public void run() {
                int fact = 1;
                for (int i = 1; i <= num; i++) {
                        fact *= i;
                }
                System.out.println("Factorial: " + fact);
        }
```

```
}

class ThreadWait {
	public static void main(String[] args) throws InterruptedException {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a number: ");
		int n = sc.nextInt();

		TableThread t1 = new TableThread(n);
		FactorialThread t2 = new FactorialThread(n);

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

```
C:\24bcsa08\Assignment_8\q7>javac ThreadWait.java

C:\24bcsa08\Assignment_8\q7>java ThreadWait
Enter a number: 10
Factorial: 3628800
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

8. Write a java program that will create two threads. Set the priority to each thread and display it.

```
class Priority extends Thread {
	public void run() {
		System.out.println(getName() + " Priority: " + getPriority());
	}
}
class ThreadPriority {
	public static void main(String[] args) {
		Priority t1 = new Priority();
		Priority t2 = new Priority();
		t1.setPriority(8);
		t2.setPriority(5);
		t1.start();
		t2.start();
	}
}
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
C:\24bcsa08\Assignment_8\q8>javac ThreadPriority.java

C:\24bcsa08\Assignment_8\q8>java ThreadPriority
Thread-1 Priority: 5
Thread-0 Priority: 8
```

9. Write a java program that will compute product of two 1D arrays using multithreading. The program should read two 1D arrays of same size from the user. First thread should multiply the corresponding elements present in the odd index position and second thread should multiply the corresponding elements present in the even index position. Main thread should display the result.

```
import java.util.Scanner;

class OddIndexProduct extends Thread {
	int[] a, b, res;
	OddIndexProduct(int[] a, int[] b, int[] res) {
		this.a = a;
		this.b = b;
		this.res = res;
	}

	public void run() {
		for (int i = 1; i < a.length; i += 2) {
			res[i] = a[i] * b[i];
		}
	}
}

class EvenIndexProduct extends Thread {
	int[] a, b, res;
	EvenIndexProduct(int[] a, int[] b, int[] res) {
		this.a = a;
		this.b = b;
		this.res = res;
	}

	public void run() {
		for (int i = 0; i < a.length; i += 2) {
			res[i] = a[i] * b[i];
		}
	}
}

class ArrayProduct {
	public static void main(String[] args) throws InterruptedException {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter size: ");
		int n = sc.nextInt();
```

```
                int[] a = new int[n];
                int[] b = new int[n];
                int[] res = new int[n];

                System.out.println("Enter elements of first array:");
                for (int i = 0; i < n; i++) {
                        a[i] = sc.nextInt();
                }

                System.out.println("Enter elements of second array:");
                for (int i = 0; i < n; i++) {
                        b[i] = sc.nextInt();
                }

                OddIndexProduct t1 = new OddIndexProduct(a, b, res);
                EvenIndexProduct t2 = new EvenIndexProduct(a, b, res);

                t1.start();
                t2.start();

                System.out.println("Result array:");
                for (int i : res) {
                        System.out.print(i + " ");
                }
        }
}
```

```
C:\24bcsa08\Assignment_8\q9>javac ArrayProduct.java

C:\24bcsa08\Assignment_8\q9>java ArrayProduct
Enter size: 10
Enter elements of first array:
8 5 9 4 9 4 2 6 4 1
Enter elements of second array:
4 9 3 5 7 5 2 8 1 4
Result array:
32 45 27 20 63 20 4 48 4 4
```

10. Write a simple Java thread program to compute the sum of n natural numbers. The program should read the number of threads m and value of n from the user. Each of the threads should add its share of assigned number to a global variable. When all the threads are done, the global variable should contain the result. The program should use a Synchronized block to make sure that only one thread is updating the global variable at a given time.

```
import java.util.Scanner;

class SumThread extends Thread {
   int start, end;
   static int total = 0;
   static final Object lock = new Object();
```

```
    SumThread(int start, int end) {
        this.start = start;
        this.end = end;
    }

    public void run() {
        int sum = 0;
        System.out.println(getName() + " adding numbers from " + start + " to " + end);
        for (int i = start; i <= end; i++) {
            sum += i;
        }

        synchronized (lock) {
            total += sum;
            System.out.println(getName() + " finished. Partial sum = " + sum + ", Total = " +
total);
        }
    }
}

public class SumOfNumbers {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt();
        System.out.print("Enter number of threads: ");
        int m = sc.nextInt();

        SumThread[] t = new SumThread[m];
        int part = n / m;
        int start = 1;

        for (int i = 0; i < m; i++) {
            int end = (i == m - 1) ? n : (start + part - 1);
            t[i] = new SumThread(start, end);
            t[i].setName("Thread-" + (i + 1));
            t[i].start();
            start = end + 1;
        }

        for (int i = 0; i < m; i++) {
            try {
                t[i].join();
            } catch (Exception e) {}
        }
        System.out.println("Final Sum of first " + n + " natural numbers = " + SumThread.total);
    }
}
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
C:\24bcsa08\Assignment_8\q10>javac SumOfNumbers.java

C:\24bcsa08\Assignment_8\q10>java SumOfNumbers
Enter n: 15
Enter number of threads: 4
Thread-3 adding numbers from 7 to 9
Thread-4 adding numbers from 10 to 15
Thread-2 adding numbers from 4 to 6
Thread-1 adding numbers from 1 to 3
Thread-3 finished. Partial sum = 24, Total = 24
Thread-1 finished. Partial sum = 6, Total = 30
Thread-2 finished. Partial sum = 15, Total = 45
Thread-4 finished. Partial sum = 75, Total = 120
Final Sum of first 15 natural numbers = 120
```

11. Write a Java thread program to search the minimum number in a given array. The program should read the number of elements in the array, number of threads to be created and the array elements from the user. Each thread should find minimum element in an assigned block of elements and compare to global minimum element. When all the threads are done, the global variable should contain the minimum element. It should use a Synchronized block to make sure that only one thread is updating the global minimum variable at any given time

```
import java.util.Scanner;

class MinFinder extends Thread {
    int[] arr;
    int start, end;
    static int globalMin = Integer.MAX_VALUE;
    static final Object lock = new Object();

    MinFinder(int[] arr, int start, int end) {
        this.arr = arr;
        this.start = start;
        this.end = end;
    }

    public void run() {
        int localMin = Integer.MAX_VALUE;
        System.out.println(getName() + " checking from index " + start + " to " + end);

        for (int i = start; i <= end; i++) {
            if (arr[i] < localMin) {
                localMin = arr[i];
            }
        }

        synchronized (lock) {
            if (localMin < globalMin) {
                globalMin = localMin;
            }
            System.out.println(getName() + " finished. Local Min = " + localMin + ", Global Min
= " + globalMin);
        }
    }
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
}

public class MinArray {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter number of elements: ");
      int n = sc.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter " + n + " elements:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }

      System.out.print("Enter number of threads: ");
      int m = sc.nextInt();
      MinFinder[] threads = new MinFinder[m];
      int part = n / m;
      int start = 0;

      for (int i = 0; i < m; i++) {
         int end = (i == m - 1) ? (n - 1) : (start + part - 1);
         threads[i] = new MinFinder(arr, start, end);
         threads[i].setName("Thread-" + (i + 1));
         threads[i].start();
         start = end + 1;
      }

      for (int i = 0; i < m; i++) {
         try {
            threads[i].join();
         } catch (Exception e) {}
      }
      System.out.println("Minimum element in the array = " + MinFinder.globalMin);
   }
}
```

```
C:\24bcsa08\Assignment_8\q11>javac MinArray.java

C:\24bcsa08\Assignment_8\q11>java MinArray
Enter number of elements: 10
Enter 10 elements:
9 5 2 8 1 6 7 2 9 3
Enter number of threads: 2
Thread-1 checking from index 0 to 4
Thread-2 checking from index 5 to 9
Thread-1 finished. Local Min = 1, Global Min = 1
Thread-2 finished. Local Min = 2, Global Min = 1
Minimum element in the array = 1
```

12. Write a java program in which main thread should create two child threads (Producer and Consumer). First child thread (Producer) should produce ten random integers between 1 to 100 and the second child thread (Consumer) should check whether the generated number is even or odd. At the end the second child thread (Consumer)

should print total number of even numbers received. Both the threads should wait and notify each other wherever necessary.

```
import java.util.Random;

class Shared {
   int num;
   boolean available = false;
}

class Producer extends Thread {
   Shared s;
   Random r = new Random();

   Producer(Shared s) {
      this.s = s;
   }

   public void run() {
      for (int i = 1; i <= 10; i++) {
         synchronized (s) {
            while (s.available) {
               try {
                  s.wait();
               } catch (InterruptedException e) {}
            }
            s.num = r.nextInt(100) + 1;
            System.out.println("Number generated by Producer: " + s.num);
            s.available = true;
            s.notify();
         }
      }
   }
}

class Consumer extends Thread {
   Shared s;
   int evenCount = 0;

   Consumer(Shared s) {
      this.s = s;
   }

   public void run() {
      for (int i = 1; i <= 10; i++) {
         synchronized (s) {
            while (!s.available) {
               try {
                  s.wait();
               } catch (InterruptedException e) {}
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025

```
            }
            if (s.num % 2 == 0) {
                evenCount++;
            }
            s.available = false;
            s.notify();
        }
    }
    System.out.println("Number even numbers received by Consumer: " + evenCount);
  }
}

public class WaitDemo {
  public static void main(String[] args) {
    Shared s = new Shared();
    Producer p = new Producer(s);
    Consumer c = new Consumer(s);

    p.start();
    c.start();

    try {
      p.join();
      c.join();
    } catch (InterruptedException e) {}
  }
}
```

```
C:\24bcsa08\Assignment_8\q12>javac WaitDemo.java

C:\24bcsa08\Assignment_8\q12>java WaitDemo
Number generated by Producer: 84
Number generated by Producer: 22
Number generated by Producer: 78
Number generated by Producer: 41
Number generated by Producer: 56
Number generated by Producer: 84
Number generated by Producer: 86
Number generated by Producer: 94
Number generated by Producer: 10
Number generated by Producer: 30
Number even numbers received by Consumer: 9
```

Name – Manojnya Harichandan
SIC – 24BCSA08
Roll No - 10
Date of Experiment - 31 / 10 / 2025