1.Write a Java program for the tasks printA, printB, and print100 are executed simultaneously to display the letter a 100 times, the letter b 100 times, and the numbers from 1 to 100.

// Define a class that implements Runnable interface

class Task implements Runnable {

// Declare a private variable to store the task name

private String name;

// Define a constructor that takes a task name as an argument

public Task(String name) {

this.name = name;

}

// Override the run method to perform the task

public void run() {

// Use a switch statement to handle different tasks

switch (name) {

case "printA":

// Print the letter a 100 times

for (int i = 0; i < 100; i++) {

System.out.print("a");

}

break;

case "printB":

// Print the letter b 100 times

for (int i = 0; i < 100; i++) {

System.out.print("b");

}

break;

case "print100":

// Print the numbers from 1 to 100

for (int i = 1; i <= 100; i++) {

System.out.print(i);

}

break;

default:

// Print an error message if the task name is invalid

System.out.println("Invalid task name: " + name);

}

}

}

// Define the main class

public class Main {

// Define the main method

public static void main(String[] args) {

// Create three objects of Task class with different names

Task taskA = new Task("printA");

Task taskB = new Task("printB");

Task task100 = new Task("print100");

// Create three threads with the task objects as arguments

Thread threadA = new Thread(taskA);

Thread threadB = new Thread(taskB);

Thread thread100 = new Thread(task100);

// Start the threads

threadA.start();

threadB.start();

thread100.start();

}

}

2.Demonstrate the concept of Threads to achieve multitasking program using Thread class and Runnable Interface (separately) for the below scenario:

Create an Array of 9 numbers. And create three Threads to split the task evenly among the three threads. And each thread has to add up and report the answer to the main thread where the main thread waits for the 3 threads and computes the summation of all the three threads. Note: Assign names to the threads as well.

/ Define a class that extends Thread class

class SumThread extends Thread {

// Declare a private array to store the numbers

private int[] array;

// Declare a private variable to store the sum

private int sum;

// Define a constructor that takes an array as an argument

public SumThread(int[] array) {

this.array = array;

this.sum = 0;

}

// Override the run method to perform the sum

public void run() {

// Loop through the array and add each element to the sum

for (int num : array) {

sum += num;

}

// Print the sum and the thread name

System.out.println("Sum by " + this.getName() + " is " + sum);

}

// Define a getter method to return the sum

public int getSum() {

return sum;

}

}

// Define the main class

public class Main {

// Define the main method

public static void main(String[] args) throws InterruptedException {

// Create an array of 9 numbers

int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9};

// Create three subarrays of size 3 each

int[] subarray1 = new int[3];

int[] subarray2 = new int[3];

int[] subarray3 = new int[3];

// Copy the elements from the original array to the subarrays

System.arraycopy(array, 0, subarray1, 0, 3);

System.arraycopy(array, 3, subarray2, 0, 3);

System.arraycopy(array, 6, subarray3, 0, 3);

// Create three objects of SumThread class with different subarrays as arguments

SumThread thread1 = new SumThread(subarray1);

SumThread thread2 = new SumThread(subarray2);

SumThread thread3 = new SumThread(subarray3);

// Assign names to the threads

thread1.setName("Thread1");

thread2.setName("Thread2");

thread3.setName("Thread3");

// Start the threads

thread1.start();

thread2.start();

thread3.start();

// Wait for the threads to finish using join method

thread1.join();

thread2.join();

thread3.join();

// Compute the summation of all the three threads using getSum method

int totalSum = thread1.getSum() + thread2.getSum() + thread3.getSum();

// Print the total sum and the main thread name

System.out.println("Total sum by " + Thread.currentThread().getName() + " is " + totalSum);

}

}