LAB17_ANP-c6339_THREADING1

Due date: Wednesday, 29 November 2023, 5:30 AM
Maximum number of files: 1
Type of work: Individual work

Lingabathula Thapaswi[AF0339471]

Assignment-0.

create a multithreading program to display the given name with welcome message display greeting for availlable users.

Solution:-

```
import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;


public class MultithreadedWelcome {


   public static void main(String[] args) {

      List<String> users = new ArrayList<>();

      users.add("Alice");

      users.add("Bob");

      users.add("Charlie");


      ExecutorService executorService = Executors.newFixedThreadPool(users.size());


      for (String user : users) {

         executorService.submit(() -> displayWelcomeMessage(user));

      }


      executorService.shutdown();

   }


   private static void displayWelcomeMessage(String user) {

      System.out.println("Welcome, " + user + "!");

   }
```

}

Output:

```
C:\Users\thila\OneDrive\Desktop\Anudip_Labs>javac MultithreadedWelcome.java

C:\Users\thila\OneDrive\Desktop\Anudip_Labs>java MultithreadedWelcome
Welcome, Charlie!
Welcome, Bob!
Welcome, Alice!
```

Assignment-1.

create a multithreading program to display the given name with welcome message display greeting for available users.

store 5 names using array of string, pass the string to the methods to display greeting message.

create 2 threads to perform the above task.

Solution:-

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class DisplayWelcomeMessage implements Runnable {

    private String name;

    public DisplayWelcomeMessage(String name) {
        this.name = name;
    }

    @Override
    public void run() {
        System.out.println("Welcome, " + name + "!");
    }
}

public class MultithreadedWelcomeMsg {

    public static void main(String[] args) {
        String[] names = {"Alice", "Bob", "Charlie", "David", "Emily"};
```

```
        ExecutorService executorService = Executors.newFixedThreadPool(2);

        for (String name : names) {
            executorService.submit(new DisplayWelcomeMessage(name));
        }

        executorService.shutdown();
    }
}
```

Output:

```
C:\Users\thila\OneDrive\Desktop\Anudip_Labs>javac MultithreadedWelcomeMsg.java

C:\Users\thila\OneDrive\Desktop\Anudip_Labs>java MultithreadedWelcomeMsg
Welcome, Bob!
Welcome, Alice!
Welcome, Charlie!
Welcome, David!
Welcome, Emily!
```

Assignment-1.1

With the program 1(1.create a multithreading program to display the given name with welcome message display greeting for available users.

store 5 names using array of string, pass the string to the methods to display greeting message.

create 2 threads to perform the above task.), assing highest and lowest priority and stop highest priority thread in middle of the process.

Solution:-

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.PriorityBlockingQueue;

class DisplayWelcomeMessage implements Runnable, Comparable<DisplayWelcomeMessage> {

    private String name;
    private int priority;

    public DisplayWelcomeMessage(String name, int priority) {
        this.name = name;
        this.priority = priority;
    }
```

```
    @Override
    public void run() {
        System.out.println("Welcome, " + name + "!");

        try {
            Thread.sleep(2000); // Simulate some work
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + name);
        }
    }

    @Override
    public int compareTo(DisplayWelcomeMessage other) {
        return Integer.compare(this.priority, other.priority);
    }
}

public class MultithreadedWelcomeWithPriority {

    public static void main(String[] args) {
        PriorityBlockingQueue<DisplayWelcomeMessage> tasks =
                new PriorityBlockingQueue<>();

        tasks.add(new DisplayWelcomeMessage("Alice", 10));
        tasks.add(new DisplayWelcomeMessage("Bob", 5));
        tasks.add(new DisplayWelcomeMessage("Charlie", 1));

        ExecutorService executorService = Executors.newFixedThreadPool(2);

        for (int i = 0; i < 3; i++) {
            executorService.submit(() -> {
                while (true) {
                    try {
```

```
                DisplayWelcomeMessage task = tasks.take();
                task.run();
            } catch (InterruptedException e) {
                break;
            }
        }
    });
}

// Interrupt the thread with the highest priority
tasks.remove(tasks.peek());

executorService.shutdown();
}
}
```

Output:

```
C:\Users\thila\OneDrive\Desktop\Anudip_Labs>javac MultithreadedWelcomeWithPriority.java

C:\Users\thila\OneDrive\Desktop\Anudip_Labs>java MultithreadedWelcomeWithPriority
Welcome, Alice!
Welcome, Bob!
^C
```

Assignment-2.

create two thread one thread is finding the average of the first 10 numbers and another thread is printing the square of the number stored in array arr={1,20,50,15,30} and make sure both threads can execute one by one.

Solution:-

```
import java.util.Scanner;


public class MultithreadedAverageAndSquare {


    public static void main(String[] args) {
        // Create an array of numbers
        int[] arr = {1, 20, 50, 15, 30};


        // Create two threads
        Thread averageThread = new Thread(() -> {
```

```
    // Calculate the average of the first 10 numbers
    int sum = 0;
    for (int i = 0; i < arr.length; i++) {
        sum += arr[i];
    }
    double average = sum / arr.length;

    // Print the average
    System.out.println("\nAverage of the numbers: " + average);
});

Thread squareThread = new Thread(() -> {
    // Calculate the square of each number in the array
    for (int i = 0; i < arr.length; i++) {
        arr[i] *= arr[i];
    }

    // Print the squared numbers
    System.out.println("Square of the numbers:");
    for (int num : arr) {
        System.out.print(num + " ");
    }
});

// Start the threads
averageThread.start();
squareThread.start();

// Wait for the threads to finish
try {
    averageThread.join();
    squareThread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
```

```
    }
  }
}
```

Output:

```
C:\Users\thila\OneDrive\Desktop\Anudip_Labs>javac MultithreadedAverageAndSquare.java

C:\Users\thila\OneDrive\Desktop\Anudip_Labs>java MultithreadedAverageAndSquare
Square of the numbers:
1 400 2500 225 900
Average of the numbers: 23.0
```