Name – Mahemud Borgave

PRN – 23620005

Batch – S3

**Assignment No – 07**

1. Can we call the run() method instead of start()?

Ans –

Yes, you can directly call the run() method instead of start() in a Java program with threads. However, doing so will execute the code in the current thread, without creating a new thread for concurrent execution. Here's an example illustrating this:

java

Copy code

public class MyThread implements Runnable {

public void run() {

System.out.println("Thread is running...");

}

public static void main(String[] args) {

MyThread myThread = new MyThread();

// Call run() directly instead of start()

myThread.run();

}

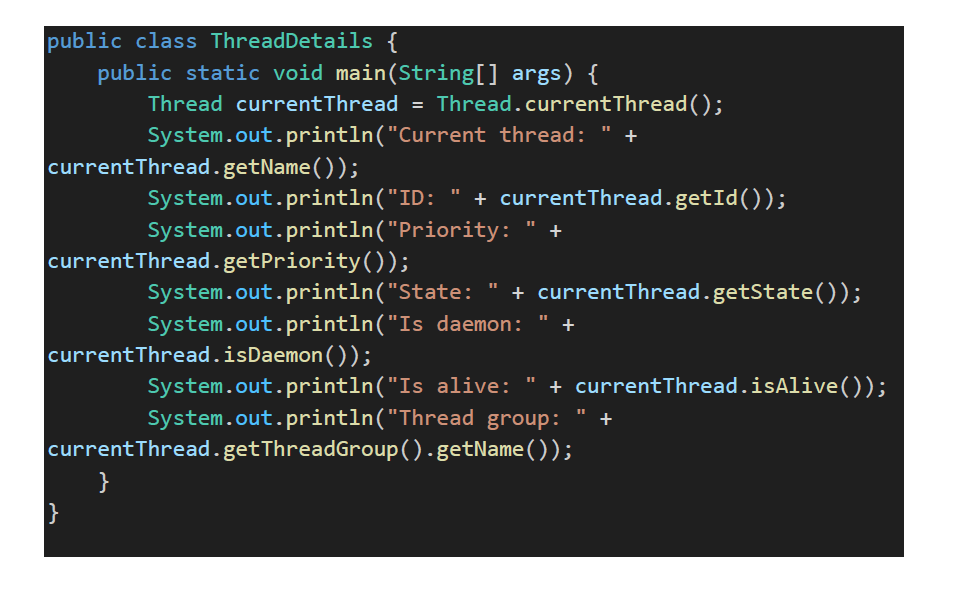
}

1. Explain the use of word Synchronized

Ans -

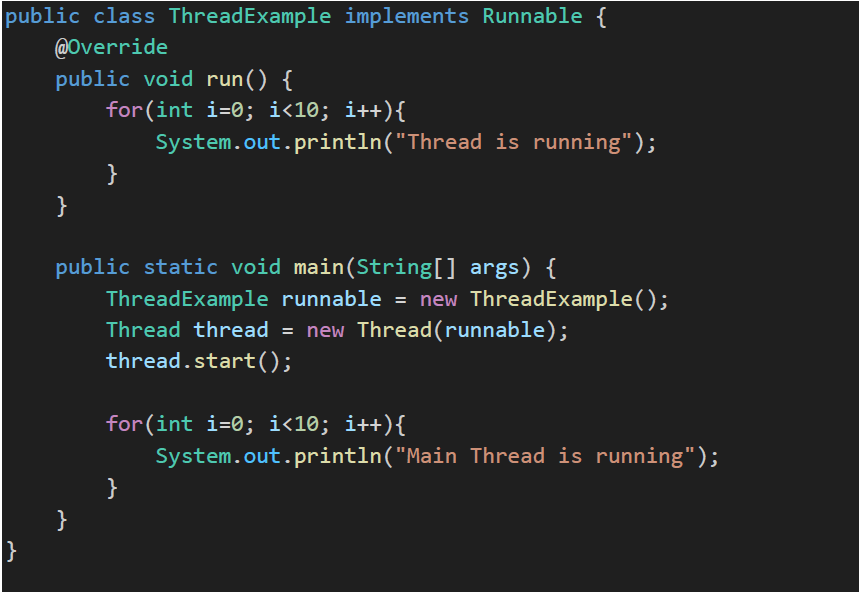
When a method or a block of code is marked as synchronized, only one thread can execute that code at any given time. Other threads must wait until the executing thread releases the lock on the object that the synchronized block is synchronized on.

1. Write a program to display thread information.





1. Create a thread using Thread class and Runnable class.





1. Write a program for thread communication and synchronization.

public class ThreadCommunicationExample {

    public static void main(String[] args) {

        Message message = new Message();

        Thread senderThread = new Thread(new Sender(message), "SenderThread");

        Thread receiverThread = new Thread(new Receiver(message), "ReceiverThread");

        senderThread.start();

        receiverThread.start();

    }

}

class Message {

    private String message;

    private boolean messageSent = false;

    // Method to send a message

    public synchronized void send(String message) {

        // If message is already sent, wait for receiver to receive it

        while (messageSent) {

            try {

                wait(); // Release the lock and wait for notify() call

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

        this.message = message;

        System.out.println("Message sent: " + message);

        messageSent = true;

        // Notify the receiver thread that message is sent

        notify();

    }

    // Method to receive a message

    public synchronized String receive() {

        // If message is not sent, wait for sender to send it

        while (!messageSent) {

            try {

                wait(); // Release the lock and wait for notify() call

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

        // Once message is sent, reset messageSent flag and notify sender

        messageSent = false;

        notify();

        return message;

    }

}

class Sender implements Runnable {

    private Message message;

    public Sender(Message message) {

        this.message = message;

    }

    public void run() {

        String[] messages = {"Hello", "How are you?", "Goodbye"};

        for (String msg : messages) {

            message.send(msg);

            try {

                Thread.sleep(1000); // Sleep for 1 second

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

}

class Receiver implements Runnable {

    private Message message;

    public Receiver(Message message) {

        this.message = message;

    }

    public void run() {

        for (int i = 0; i < 3; i++) {

            String receivedMessage = message.receive();

            System.out.println("Message received: " + receivedMessage);

        }

    }

}

