Name: Thota GuruTheja Reddy

Regno: 19BCD7034

Lab: 10

1. Consider two friends Herbert and Schildt are chatting and discussing about next day exam preparation. Create a class chat with two methods Ask and Reply to perform this. Create two threads Herbert and Schidlt using Runnable Interface. Implement inter thread communication for the scenario and exchange the following messages

Herbert : Hi

Schildt : Hello

Herbert : Tomorrow do you have exam

Schildt : yes

Herbert : have you prepared?

Schildt : ya i am preparing

Herbert : All the best!

Schildt : Thank you

Ans:

class Chat {

boolean flag = false;

public synchronized void Ask(String msg) {

if (flag) {

try {

wait();

} catch (InterruptedException e) {

System.out.println(e);

}

}

System.out.println(msg);

flag = true;

notify();

}

public synchronized void Reply(String msg) {

if (!flag) {

try {

wait();

} catch (InterruptedException e) {

System.out.println(e);

}

}

System.out.println(msg);

flag = false;

notify();

}

}

class Herbert implements Runnable {

Chat m;

String[] s1 = { "Hi", "Tomorrow do you have exam", "Have you prepared?","All the best!" };

public Herbert(Chat m1) {

this.m = m1;

new Thread(this, "Ask").start();

}

public void run() {

for (int i = 0; i < s1.length; i++) {

m.Ask(s1[i]);

}

}

}

class Schildt implements Runnable {

Chat m;

String[] s2 = { "Hello", "Yes", "ya I am preparing", "Thank you" };

public Schildt(Chat m2) {

this.m = m2;

new Thread(this, "Reply").start();

}

public void run() {

for (int i = 0; i < s2.length; i++) {

m.Reply(s2[i]);

}

}

}

public class Main {

public static void main(String[] args) {

Chat m = new Chat();

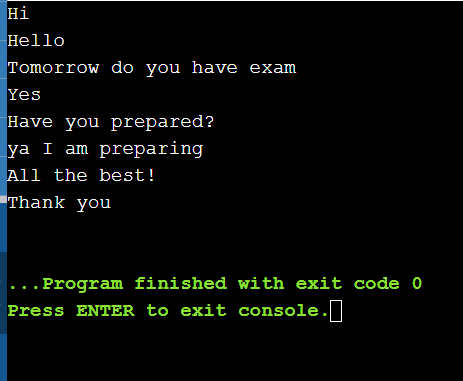
new Herbert(m);

new Schildt(m);

}

}

Output:



2. Apply Inter thread communication to solve the Producer-Consumer problem with a common or shared bounded buffer(Queue) holding up to 5 elements.

The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enters them into the buffer. The consumer removes the items from the buffer and consumes them.

A producer should not produce items into the buffer when the consumer is consuming an item from the buffer and vice versa. So the buffer should only be accessed by the producer or consumer at a time.

Whenever buffer is filled up and no more space to add the element into the queue(buffer) producer has to wait until the buffer is emptied by consumer. Whenever the buffer is empty and no more items are available for consumption the consumer should wait for producer to produce elements. Write a solution for N elements, where N is multiple of 5 other than 0.

Ans:

import java.util.concurrent.\*;

class Producer extends Thread {

private BlockingQueue<Integer> sharedQueue;

public Producer(BlockingQueue<Integer> aQueue) {

super("PRODUCER");

this.sharedQueue = aQueue;

}

public void run() {

for (int i = 1; i < 6; i++) {

try {

System.out.println(getName() + " produced " + i);

sharedQueue.put(i);

Thread.sleep(200);

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

class Consumer extends Thread {

private BlockingQueue<Integer> sharedQueue;

public Consumer(BlockingQueue<Integer> aQueue) {

super("CONSUMER");

this.sharedQueue = aQueue;

}

public void run() {

try {

while (true) {

Integer item = sharedQueue.take();

System.out.println(getName() + " consumed " + item);

}

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

public class Main {

public static void main(String[] args) {

BlockingQueue<Integer> sharedQ = new LinkedBlockingQueue<Integer>();

Producer p = new Producer(sharedQ);

Consumer c = new Consumer(sharedQ);

p.start();

c.start();

}

}

Output:

