16. Create multiple threads to access the contents of a stack. Synchronize thread to prevent simultaneous access to push and pop operations

```
package q29795;
import java.util.*;
import java.util.Scanner;
import java.util.Stack;
public class StackThreads{
public static void main(String[] args){
Scanner sc=new Scanner(System.in);
System.out.println("Enter the size of the stack");
int size=sc.nextInt();
int i;
Stack<Integer>stack=new Stack();
for(i=0;i<size;i++){
System.out.println(i+1);
}
}
}
```

17. Write a Java program on creating multiple threads.

```
package q29796;
class threaddemo extends Thread{
public void run(){
System.out.println("Main thread exiting.");
for(int i=4;i>0;i--){
System.out.println("Two:"+i);
System.out.println("One:"+i);
System.out.println("Three:"+i);
}
System.out.println("Two exiting");
System.out.println("One exiting");
System.out.println("Three exiting");
}
}
class firstthread extends Thread{
public void run(){
System.out.println("New thread: Thread[One,5,main]");
System.out.println("New thread: Thread[Two,5,main]");
System.out.println("New thread: Thread[Three,5,main]");
System.out.println("Two:5\nOne:5\nThree:5");
//System.out.println("Main thread exiting");
}
}
public class MultiThreadDemo{
public static void main(String[] args){
threaddemo t1=new threaddemo();
firstthread m=new firstthread();
```

```
try{
m.start();
m.join();
}
catch(Exception e){
//System.out.println(e);
}
t1.start();
}
}
```

18. Create a Java program that utilizes multi-threading to generate multiplication tables.

```
package q18198;
import java.util.Scanner;

class TablePrinter implements Runnable {
private int tableNumber;

public TablePrinter(int tableNumber) {
this.tableNumber = tableNumber;
}

public void run(){
for(int i=1;i<=10;i++){
for(int j=1;j<=tableNumber;j++){
System.out.println(j+" * "+i+" = "+i*j);
}
}
}
//write your code....
}

public class Main {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the number of tables:");
int numTables = scanner.nextInt();
Thread[] threads = new Thread[numTables];
TablePrinter table = new TablePrinter(numTables);
Thread t1=new Thread(table);
t1.start();
//write your code....
}
}
```

19. Define two threads such that one thread should print even numbers and another thread should print odd numbers.

```
package q17210;
import java.util.*;
class even extends Thread{
int e;
public even(int e){
this.e=e;
}
public void run(){
for(int i=2;i<=e;i=i+2){
System.out.println("Even:"+i);
}
}
}
class odd extends Thread{
int o;
public odd(int o){
this.o=o;
}
public void run(){
for(int i=1;i<=o;i=i+2){
System.out.println("Odd:"+i);
}
}
}
public class Main{
public static void main(String[] args){
Scanner sc=new Scanner(System.in);
System.out.println("Enter the maximum even number:");
int n=sc.nextInt();
even t2=new even(n);
System.out.println("Enter the maximum odd number:");
int o=sc.nextInt();
odd t1=new odd(o);
t1.start();
t2.start();
}
}
```

20. Write a Java program that correctly implements the producer – consumer problem using the concept of inter thread communication

```
package q36394;
try {
wait();
} catch (InterruptedException e) {
e.printStackTrace();
```

```
}
}
this.value = value;
System.out.println("PUT:" + value);
available = true;
notify();
}
// write your code..

}
class Producer implements Runnable
{
private Product product;

Producer(Product product) {
this.product = product;
Thread t1 = new Thread(this);
t1.start();
}

public void run() {
for (int i = 0; i < 6; i++) {
product.put(i);
try {
Thread.sleep(100); // Simulating some processing time
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
// write your code..


}
class Consumer implements Runnable
{
private Product product;

Consumer(Product product) {
this.product = product;
Thread t2 = new Thread(this);
t2.start();
}

public void run() {
for (int i = 0; i < 6; i++) {
product.get();
try {
Thread.sleep(100); // Simulating some processing time
```

```
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}



}
```