**Assignment No. 5**

**Multinthreading**

**Question 1**

**Write a Java program to create and run a thread by extending the Thread class. The thread should print "Hello from Thread" five times.**

**Ans:-**

class MyThread extends Thread

{

public void run()

{

for (int i = 0;i<5;i++)

{

System.out.println("Hello from Thread");

try

{

Thread.sleep(500);

}

catch (InterruptedException e)

{

System.out.println("Thread interrupted");

}

}

}

}

public class ThreadExample1

{

public static void main(String[] args)

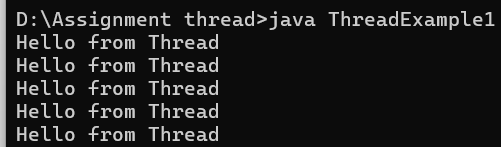
{

MyThread thread = new MyThread();

thread.start();

}

}



**Question 2**

**Write a Java program to create and run a thread by implementing the Runnable interface. The thread should print numbers from 1 to 5.**

**Ans:-**

class NumberPrinter implements Runnable

{

@Override

public void run()

{

for(int i = 1; i <= 5; i++)

{

System.out.println("Number:" +i);

try

{

Thread.sleep(500);

}

catch (InterruptedException e)

{

System.out.println("Thread interrupted");

}

}

}

}

public class ThreadExample2

{

public static void main(String[] args)

{

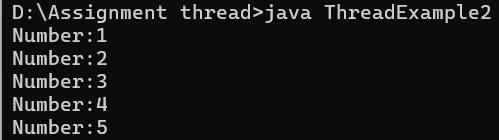
NumberPrinter printer = new NumberPrinter();

Thread thread = new Thread(printer);

thread.start();

}

}

****

**Question 3**

**Write a Java program where the main thread prints "Main Thread Running" and a child thread prints "Child Thread Running". Run them simultaneously.**

**Ans:-**

public class ThreadExample3

{

public static void main(String[] args)

{

Thread childThread = new Thread(() ->{

System.out.println("child Thread Running");

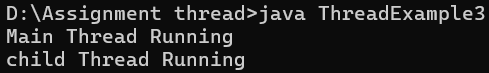
});

childThread.start();

System.out.println("Main Thread Running");

}

}

****

**Question 4**

**Write a Java program to demonstrate the use of setName() and getName() methods for threads.**

**Ans:-**

public class ThreadExample4

{

public static void main(String[] args)

{

Thread thread1 = new Thread(() ->

{

System.out.println("Running: "+ Thread.currentThread().getName());

});

thread1.setName("Custom-Thread-1");

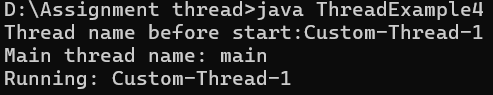
System.out.println("Thread name before start:"+ thread1.getName());

thread1.start();

System.out.println("Main thread name: "+ Thread.currentThread().getName());

}

}

****

**Question 5**

**Write a Java program to demonstrate the use of setPriority() and getPriority() methods by creating two threads with different priorities.**

**Ans:-**

public class ThreadExample5

{

public static void main(String[] args)

{

Thread highPriorityThread = new Thread(() ->

{

System.out.println("High Priority Thread running with priority:"+ Thread.currentThread().getPriority());

});

Thread lowPriorityThread = new Thread(() ->

{

System.out.println("Low Priority Thread running with priority:"+ Thread.currentThread().getPriority());

});

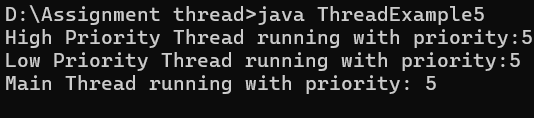
highPriorityThread.start();

lowPriorityThread.start();

System.out.println("Main Thread running with priority: " +Thread.currentThread().getPriority());

}

}



**Question 6**

**Write a Java program where one thread prints numbers from 1 to 10, and another thread prints numbers from 11 to 20.**

**Ans:-**

public class ThreadExample6

{

public static void main(String[] args)

{

Thread thread1 = new Thread(() ->

{

for (int i=1; i<=10; i++)

{

System.out.println("Thread 1: "+i);

try{

Thread.sleep(100);

}catch (InterruptedException e){

Thread.currentThread().interrupt();

}

}

});

Thread thread2 = new Thread(() ->

{

for (int i=11; i<=20; i++)

{

System.out.println("Thread 2: "+i);

try{

Thread.sleep(100);

}catch (InterruptedException e){

Thread.currentThread().interrupt();

}

}

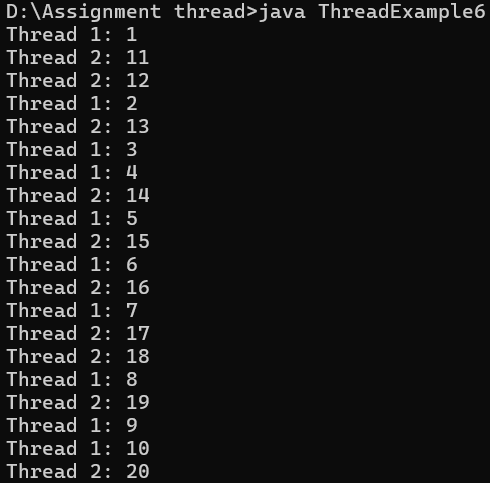
});

thread1.start();

thread2.start();

}

}



**Question 7**

**Write a Java program to demonstrate the use of the sleep() method by pausing a thread for 1 second after printing each number.**

**Ans:-**

public class SleepDemo

{

public static void main(String[] args)

{

for (int i=1; i<=5; i++)

{

System.out.println("Number: "+i);

try

{

Thread.sleep(1000);

}

catch (InterruptedException e)

{

System.out.println("Thread was interrupted.");

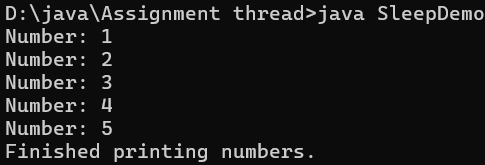
}

}

System.out.println("Finished printing numbers.");

}

}



**Question 8**

**Write a Java program where the main thread waits for a child thread to finish using the join() method.**

**Ans:-**

public class JoinExample

{

public static void main(String[] args)

{

System.out.println("Main thread started");

Thread childThread = new Thread(() ->

{

System.out.println("Child thread started.");

try

{

Thread.sleep(2000);

}

catch (InterruptedException e)

{

System.out.println("child thread interrupted.");

}

System.out.println("child thread finished");

});

childThread.start();

try

{

childThread.join();

}

catch (InterruptedException e)

{

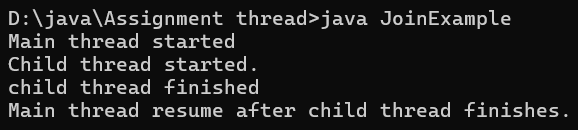
System.out.println("Main thread interrupted while waiting.");

}

System.out.println("Main thread resume after child thread finishes.");

}

}

****

**Question 9**

**Write a Java program to check whether a thread is alive or not using the isAlive() method.**

**Ans:-**

public class IsAliveExample

{

public static void main(String[] args)

{

Thread workerThread = new Thread(() ->

{

System.out.println("Worker thread started.");

try

{

Thread.sleep(3000);

}

catch(InterruptedException e)

{

System.out.println("Worker thread interrupted");

}

System.out.println("Worker thread finished");

});

workerThread.start();

System.out.println("Is worker thread alive?"+ workerThread.isAlive());

try{

Thread.sleep(1000);

System.out.println("Checking after 1 second: Is worker thread alive?"+workerThread.isAlive());

workerThread.join();

}

catch (InterruptedException e)

{

System.out.println("Main thread innterrupted.");

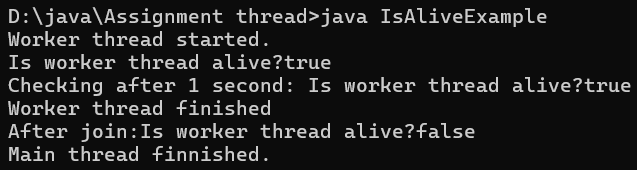
}

System.out.println("After join:Is worker thread alive?"+workerThread.isAlive());

System.out.println("Main thread finnished.");

}

}



**Question 10**

**Write a Java program to create two threads: ● Thread 1 prints "Good Morning" 5 times.**

**● Thread 2 prints "Welcome" 5 times.**

**Run both threads simultaneously.**

**Ans:-**

public class TwoThreadsExample

{

public static void main(String[] args)

{

Thread thread1 = new Thread(() ->

{

for (int i=0; i<5; i++)

{

System.out.println("Good Morning");

try

{

Thread.sleep(500);

}

catch (InterruptedException e)

{

System.out.println("Thread1 interrupted");

}

}

});

Thread thread2 = new Thread(() ->

{

for (int i=0; i<5; i++)

{

System.out.println("Welcome");

try

{

Thread.sleep(500);

}

catch (InterruptedException e)

{

System.out.println("Thread2 interrupted");

}

}

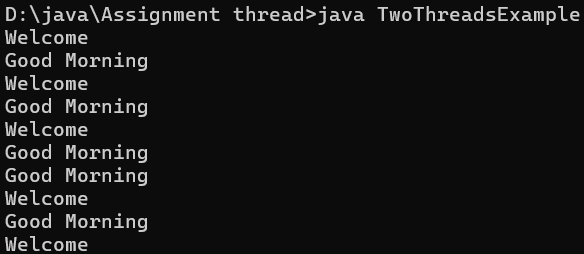
});

thread1.start();

thread2.start();

}

}



**Question 11**

**Write a Java program where one thread prints even numbers from 2 to 20, and another thread prints odd numbers from 1 to 19.**

**Ans:-**

public class EvenOddPrinter

{

public static void main(String[] args)

{

Thread threadeven = new Thread(() ->

{

for (int i=2; i<=20; i+=2)

{

System.out.println("Even: "+i);

try

{

Thread.sleep(100);

}

catch(InterruptedException e)

{

Thread.currentThread().interrupt();

}

}

});

Thread threadodd = new Thread(() ->

{

for (int i=1; i<20; i+=2)

{

System.out.println("odd: "+i);

try

{

Thread.sleep(100);

}

catch(InterruptedException e)

{

Thread.currentThread().interrupt();

}

}

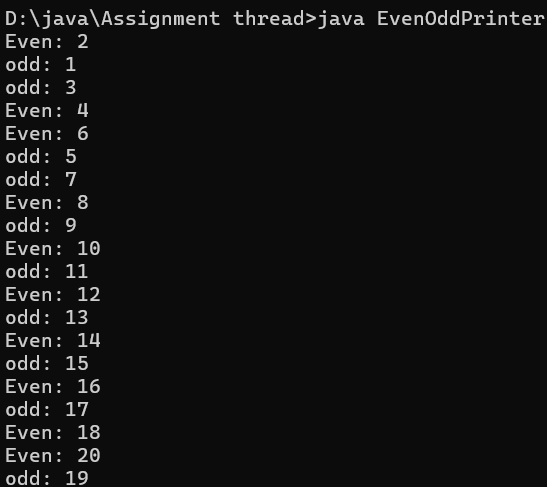
});

threadeven.start();

threadodd.start();

}

}



**Question 12**

**Write a Java program to create three threads. Each thread should print its own message 3 times.**

**Ans:-**

public class MultiThreadMessage

{

public static void main(String[] args)

{

Runnable task1 = () ->

{

for (int i=0; i<3; i++)

{

System.out.println("Thread 1:Hello from Thread one!");

try

{

Thread.sleep(100);

}

catch (InterruptedException e)

{

Thread.currentThread().interrupt();

}

}

};

Runnable task2 = () ->

{

for (int i=0; i<3; i++)

{

System.out.println("Thread 2:Hello from Thread one!");

try

{

Thread.sleep(100);

}

catch (InterruptedException e)

{

Thread.currentThread().interrupt();

}

}

};

Runnable task3 = () ->

{

for (int i=0; i<3; i++)

{

System.out.println("Thread 3:Hello from Thread one!");

try

{

Thread.sleep(100);

}

catch (InterruptedException e)

{

Thread.currentThread().interrupt();

}

}

};

Thread thread1 = new Thread(task1);

Thread thread2 = new Thread(task2);

Thread thread3 = new Thread(task3);

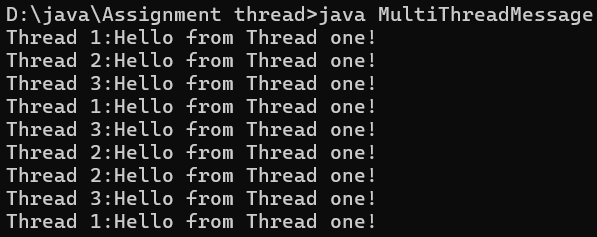
thread1.start();

thread2.start();

thread3.start();

}

}

****

**Question 13**

**Write a Java program to demonstrate the difference between calling run() directly and calling start() on a thread.**

**Ans:-**

public class Threaddiff

{

public static void main(String[] args)

{

Runnable task = () ->

{

System.out.println("Thread is running:"+Thread.currentThread().getName());

};

Thread thread = new Thread(task);

System.out.println("Calling run() directly:");

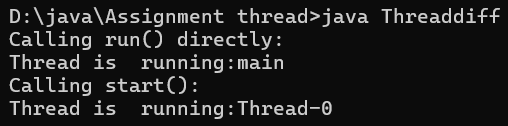
thread.run();

System.out.println("Calling start():");

thread.start();

}

}

****

**Question 14**

**Write a Java program to create a thread that calculates the sum of numbers from 1 to 100.**

**Ans:-**

public class SumThread

{

public static void main(String[] args)

{

Thread sumThread = new Thread(() ->

{

int sum = 0;

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

{

sum +=i;

}

System.out.println("Sum of numbers from 1 to 100 is:"+sum);

});

sumThread.start();

}

}



**Question 15**

**Write a Java program to demonstrate how to stop a thread gracefully using a boolean flag instead of the deprecated stop() method.**

**Ans:-**

public class GracefulStopDemo

{

private static volatile boolean running = true;

public static void main(String[] args)

{

Thread worker = new Thread(() ->

{

while (running)

{

System.out.println("Thred is working");

try

{

Thread.sleep(500);

}

catch(InterruptedException e)

{

Thread.currentThread().interrupt();

System.out.println("Thread interrupted");

}

}

System.out.println("Thread stopped gracefully");

});

worker.start();

try

{

Thread.sleep(3000);

}

catch (InterruptedException e)

{

Thread.currentThread().interrupt();

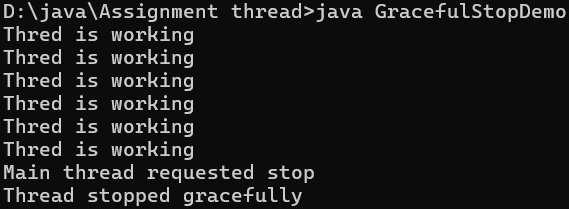
}

running = false;

System.out.println("Main thread requested stop");

}

}

****

**Question 16**

**Write a Java program where one thread prints the lowercase alphabet (a to z), and another thread prints the uppercase alphabet (A to Z).**

**Ans:-**

public class AlphabetPrinter {

public static void main(String[] args) {

Thread lowercaseThread = new Thread(() -> {

for (char c = 'a'; c <= 'z'; c++) {

System.out.print(c + " ");

try {

Thread.sleep(50);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

});

Thread uppercaseThread = new Thread(() -> {

for (char c = 'A'; c <= 'Z'; c++) {

System.out.print(c + " ");

try {

Thread.sleep(50);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

});

lowercaseThread.start();

uppercaseThread.start();

}

}

****

**Question 17**

**Write a Java program to demonstrate how multiple threads can access a shared counter variable. Show the problem of race condition (without synchronization).**

**Ans:-**

public class RaceConditionDemo {

static int counter = 0;

public static void main(String[] args) {

Thread t1 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

counter++;

}

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

counter++;

}

});

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final counter value: " + counter);

}

}

****

**Question 18**

**Write a Java program to demonstrate synchronization by using the synchronized keyword on a method that increments a counter.**

**Ans:-**

class Counter {

private int count = 0;

public synchronized void increment() {

count++;

}

public int getCount() {

return count;

}

}

public class SynchronizedDemo {

public static void main(String[] args) {

Counter counter = new Counter();

Thread t1 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

counter.increment();

}

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 1000; i++) {

counter.increment();

}

});

t1.start();

t2.start();

try {

t1.join();

t2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Final counter value: " + counter.getCount());

}

}

****

**Question 19**

**Write a Java program to create a thread that prints the current time every 2 seconds, five times.**

**Ans:-**

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

public class TimePrinter implements Runnable {

@Override

public void run() {

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");

for (int i = 0; i < 5; i++) {

LocalTime now = LocalTime.now();

System.out.println("Current time: " + now.format(formatter));

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

System.out.println("Thread was interrupted.");

}

}

}

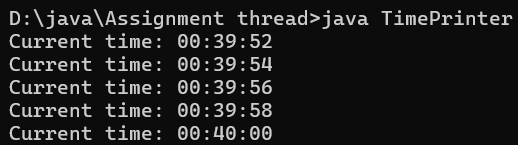
public static void main(String[] args) {

Thread timeThread = new Thread(new TimePrinter());

timeThread.start();

}

}

****

**Question 20**

**Write a Java program where two threads run in parallel: ● The first thread prints "Learning Java" 5 times.**

**● The second thread prints "Multithreading in action" 5 times.**

**Ans:-**

public class ParallelThreadsDemo

{

public static void main(String[] args) {

Thread thread1 = new Thread(() -> {

for (int i = 0; i < 5; i++) {

System.out.println("Learning Java");

try {

Thread.sleep(500);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

});

Thread thread2 = new Thread(() -> {

for (int i = 0; i < 5; i++) {

System.out.println("Multithreading in action");

try {

Thread.sleep(500);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

});

thread1.start();

thread2.start();

}

}

