**//Example 1: Demonstration\_1.java**

/\* Creating three threads using the class Thread and then running them concurrently. \*/

class ThreadA extends Thread{

public void run( ) {

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

System.out.println("From Thread A with i = "+ -1\*i);

}

System.out.println("Exiting from Thread A ...");

}

}

class ThreadB extends Thread {

public void run( ) {

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B with j= "+2\* j);

}

System.out.println("Exiting from Thread B ...");

}

}

class ThreadC extends Thread{

public void run( ) {

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread C with k = "+ (2\*k-1));

}

System.out.println("Exiting from Thread C ...");

}

}

public class Demonstration\_1 {

public static void main(String args[]) {

ThreadA a = new ThreadA();

ThreadB b = new ThreadB();

ThreadC c = new ThreadC();

a.start();

b.start();

c.start();

System.out.println("... Multithreading is over ");

}

}

**//Example 2: Demonstration\_2.java**

/\* Creating three threads using the Runnable interface and then running them concurrently. \*/

class ThreadX implements Runnable{

public void run( ) {

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

System.out.println("Thread X with i = "+ -1\*i);

}

System.out.println("Exiting Thread X ...");

}

}

class ThreadY implements Runnable {

public void run( ) {

for(int j = 1; j <= 5; j++) {

System.out.println("Thread Y with j = "+ 2\*j);

}

System.out.println("Exiting Thread Y ...");

}

}

class ThreadZ implements Runnable{

public void run( ) {

for(int k = 1; k <= 5; k++) {

System.out.println("Thread Z with k = "+ (2\*k-1));

}

System.out.println("Exiting Thread Z ...");

}

}

public class Demonstration\_2 {

public static void main(String args[]) {

ThreadX x = new ThreadX();

Thread t1 = new Thread(x);

ThreadY y = new ThreadY();

Thread t2 = new Thread(y);

//ThreadZ z = new ThreadZ();

//Thread t3 = new Thread(z);

Thread t3 = new Thread(new ThreadZ());

t1.start();

t2.start();

t3.start();

System.out.println("... Multithreading is over ");

}

}

**//Example 3: Demonstration\_3.java**

/\* Demonstration of thread class methods : getID() \*/

/\* Java code for thread creation by extending the Thread class \*/

class ThreadId extends Thread {

public void run() {

try {

// Displaying the thread that is running

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

}

catch (Exception e) {

// Throwing an exception

System.out.println ("Exception is caught");

}

}

}

public class Demonstration\_3{

public static void main(String[] args)

{

int n = 8; // Number of threads

for (int i=0; i<8; i++)

{

ThreadId object = new ThreadId();

object.start();

}

}

}

**//Example 4: Demonstration\_4.java**

/\* Use of yield(), stop() and sleep() methods \*/

class ClassA extends Thread{

public void run() {

System.out.println("Start Thread A ....");

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

if (i==1) yield();

System.out.println("From Thread A: i = "+ i);

}

System.out.println("... Exit Thread A");

}

}

class ClassB extends Thread{

public void run() {

System.out.println("Start Thread B ....");

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B: j = "+ j);

if (j==2) stop();

}

System.out.println("... Exit Thread B");

}

}

class ClassC extends Thread{

public void run() {

System.out.println("Start Thread C ....");

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread B: j = "+ k);

if (k==3){

try{

sleep(1000);

}catch(Exception e){}

}

}

System.out.println("... Exit Thread C");

}

}

public class Demonstration\_4{

public static void main (String args[]) {

ClassA t1 = new ClassA();

ClassB t2 = new ClassB();

ClassC t3 = new ClassC();

t1.start(); t2.start(); t3.start();

System.out.println("... End of executuion ");

**}**

**}**

**//Example 5: Demonstration\_5.java**

/\* Use of suspend() and resume() methods \*/

class Thread1 extends Thread {

public void run( ) {

try{

System.out.println (" First thread starts running" );

sleep(10000);

System.out.println (" First thread finishes running" );

}

catch(Exception e){ }

}

}

class Thread2 extends Thread {

public void run( ) {

try{

System.out.println ( "Second thread starts running");

System.out.println ( "Second thread is suspended itself ");

suspend( );

System.out.println (" Second thread runs again" );

}

catch(Exception e){ }

}

}

class Demonstration\_5{

public static void main (String args[ ] ){

try{

Thread1 first = new Thread1( ); // It is a newborn thread i.e. in Newborn state

Thread2 second= new Thread2( ); // another new born thread

first.start( ); // first is scheduled for running

second.start( ); // second is scheduled for running

System.out.println("Revive the second thread" ); // If it is suspended

second.resume( );

System.out.println ("Second thread went for 10 seconds sleep " );

second.sleep (10000);

System.out.println ("Wake up second thread and finishes running" );

System.out.println ( " Demonstration is finished ");

}

catch(Exception e){ }

}

}

**//Example 6: Demonstration\_6.java**

/\* Setting priority to threads \*/

class ClassA extends Thread{

public void run() {

System.out.println("Start Thread A ....");

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

System.out.println("From Thread A: i = "+ i);

}

System.out.println("... Exit Thread A");

}

}

class ClassB extends Thread{

public void run() {

System.out.println("Start Thread B ....");

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B: j = "+ j);

}

System.out.println("... Exit Thread B");

}

}

class ClassC extends Thread{

public void run() {

System.out.println("Start Thread C ....");

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread B: j = "+ k);

}

System.out.println("... Exit Thread C");

}

}

class Demonstration\_6{

public static void main (String args[]) {

ThreadA t1 = new ThreadA();

ThreadB t2 = new ThreadB();

ThreadC t3 = new ThreadC();

t3.setPriority(Thread.MAX\_PRIORITY);

t2.setPriority(t2.getPriority() + 1);

t1.setPriority(Thread.MIN\_PRIORITY);

t1.start(); t2.start(); t3.start();

System.out.println("... End of executuion ");

}

}

**//Example 7: ThreadPriorityTest.java**

/\* Setting priority to threads \*/

class ClassA extends Thread{

public void run() {

System.out.println("Start Thread A ....");

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

System.out.println("From Thread A: i = "+ i);

}

System.out.println("... Exit Thread A");

}

}

class ClassB extends Thread{

public void run() {

System.out.println("Start Thread B ....");

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B: j = "+ j);

}

System.out.println("... Exit Thread B");

}

}

class ClassC extends Thread{

public void run() {

System.out.println("Start Thread C ....");

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread B: j = "+ j);

}

System.out.println("... Exit Thread C");

}

}

class ThreadPriorityTest{

public static void main (String args[]) {

TheadA t1 = new ThreadA();

TheadB t2 = new ThreadB();

TheadC t3 = new Thread3();

t3.setPriority(Thread.MAX\_PRIORITY);

t2.setPriority(Thread.getPriority() + 1);

t1.setPriority(Thread.MIN\_PRIORITY);

t1.start(); t2.start(); t3.start();

System.out.println("... End of executuion ");

}

}

**//Example 8: ArrangingNos.java**

/\* Following Java application create a list of numbers and then sort in ascending order as well as in descending order simultaneously. \*/

import java.util.\*;

class Numbers {

public int result[] = new int[10];

void displayListOfNos()

{

System.out.println("Numbers stored in the array:");

for( int idx=0; idx<10; ++idx) {

System.out.println(result[idx]);

}

}

void fillTheArray(int aUpperLimit, int aArraySize)

{

if (aUpperLimit <=0) {

throw new IllegalArgumentException("UpperLimit must be positive: " + aUpperLimit);

}

if (aArraySize <=0) {

throw new IllegalArgumentException("Size of returned List must be greater than 0.");

}

Random generator = new Random();

for( int idx=0; idx result[j]) {

int temp = result[i];

result[i] = result[j];

result[j] = temp;

}

}

}

displayListOfNos();

}

}

class ArrangementAscending implements Runnable {

Numbers n1 ;

ArrangementAscending(Numbers n) {

n1 = n;

new Thread(this).start();

}

public void run() {

n1.sortAscending();

}

}

class ArrangementDescending implements Runnable {

Numbers n2;

ArrangementDescending(Numbers n) {

n2 = n;

new Thread(this).start();

}

public void run() {

n2.sortDescending();

}

}

class ArrangingNos {

public static void main(String args[]) {

Numbers n = new Numbers();

n.fillTheArray(20,10);

ArrangementAscending a1 = new ArrangementAscending(n);

ArrangementDescending d1 = new ArrangementDescending(n);

}

}