**NAME: TAKANKHAR SHUBHAM**

**CLASS: SY MCA**

**ROLL NO: 54**

**LAB ASSIGNMENT 2 : THREAD CREATION ,CLASS METHODS AND INTER-THREAD COMMUNICATION**

**SOURCE CODE**

//creation of thread using Thread class

class threads extends Thread {

    public void run() {

        System.out.println("Running thread: " + this.getName());

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

            try {

                Thread.sleep(200);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

            System.out.println(i);

        }

    }

    public static void main(String args[]) {

        threads t0 = new threads();

        threads t1 = new threads();

        threads t2 = new threads();

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

        System.out.println("ID:" + t0.getId() + "\tName:" + t0.getName() + "\tState:" + t0.getState() + "\tPriority:"

                + t0.getPriority());

        System.out.println("ID:" + t1.getId() + "\tName:" + t1.getName() + "\tState:" + t1.getState() + "\tPriority:"

                + t1.getPriority());

        System.out.println("ID:" + t2.getId() + "\tName:" + t2.getName() + "\tState:" + t2.getState() + "\tPriority:"

                + t2.getPriority() + "\n");

        // set name of thread

        System.out.println("Changing Name/Priority");

        t2.setName("ThreadTwo");

        t2.setPriority(10);

        System.out.println("ID:" + t2.getId() + "\tName:" + t2.getName() + "\tState:" + t2.getState() + "\tPriority:"

                + t2.getPriority() + "\n");

        // DAEMON THREAD

        System.out.println("Changing thread 2 to Daemon thread");

        System.out.println("Before >> Alive: " + t2.isAlive() + "\tDaemon: " + t2.isDaemon());

        t2.setDaemon(true);

        System.out.println("After  >> Alive: " + t2.isAlive() + "\tDaemon: " + t2.isDaemon());

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

        System.out.println("\nOUTPUT:");

        t0.start();

        try {

            t0.join();

            System.out.println("<<join method refuses other threads (1,2) to run until thread 0 is terminated>>\n");

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

        Thread.yield();

        t1.start();

        t2.start();

        try {

            sleep(1201); // to halt processing for 1201 millisecs

            System.out.println("<<we get output from thread 2 then thread 1 as Priority of thread 2 is higher>>\n");

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

        // Inter-thread communication

        System.out.println("Inter-Thread Communication: ");

        Data data = new Data();

        Thread sender = new Thread(new Sender(data));

        sender.start();

        Thread receiver = new Thread(new Receiver(data));

        receiver.start();

    }

}

class Sender implements Runnable {

    Data data;

    Sender(Data data) {

        this.data = data;

    }

    public void run() {

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

        String msg = "\tHello From," + Thread.currentThread().getName();

        data.send(msg);

    }

};

class Receiver implements Runnable {

    static String message;

    Receiver(Data data) {

        message = data.receive();

    }

    public void run() {

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

        System.out.println(message);

    }

}

class Data {

    boolean transfer = true;

    static String msg;

    synchronized void send(String msg) {

        while (!transfer) {

            try {

                wait();

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

                System.out.println("Thread interrupted:" + e);

            }

        }

        transfer = false;

        this.msg = msg;

        notifyAll();

    }

    synchronized String receive() {

        while (transfer) {

            try {

                wait();

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

                System.out.println("Thread interrupted:" + e);

            }

        }

        transfer = true;

        notifyAll();

        return msg;

    }

}

**OUTPUT**

