

Concurrent Programming (IV)

Threads in Java

- There are two ways to create threads in Java: extends the Thread class or implement the Runnable interface.
- In both ways, we need to implement run method, which is equivalent to the worker function in C.
- Then, we need to create a thread object and call the start method.
- Show HelloRunnable.java and HelloThread.java

```
/**
 * HelloRunnable.java
 */
public class HelloRunnable implements Runnable {
    public void run () {
        System.out.println("Hello Runnable!");
    }

    public static void main (String[] args) {
        Thread t = new Thread(new HelloRunnable());
        t.start();
    }
}
```

```
/**
 * HelloThread.java
 */
public class HelloThread extends Thread {
    public void run () {
        System.out.println("Hello Thread!");
    }

    public static void main (String[] args) {
        Thread t = new HelloThread();
        t.start();
    }
}
```

- The difference in creating a thread object is due to that the Thread class implements runnable interface.
- Show MyJoin.java
 - Run the code without the join part (for (JoinThread s : list)), and ask why we get the output.
 - Output: 13298
 - Reason: thread for main method ends before other threads finish.
 - Solution: using join()
 - t.join() causes the current thread to pause execution until t's thread terminates.
 - Run the code with join part, but without synchronization part (MyJoin.addCount()), ask the reason.
 - Output: 14938
 - Reason: race conditions
 - Solution: synchronization
 - Run the code with join part and the synchronization part.
 - Output: 50000
 - synchronized: synchronized methods provide a simply strategy to avoid race conditions.

```
/**
 * MyJoin.java
 */
import java.util.ArrayList;

class JoinThread extends Thread {
    public void run () {

        for (int i = 0; i < 10000; i++)
            MyJoin.count++;

        // for (int i = 0; i < 10000; i++)
        //     MyJoin.addCount();
    }
}

public class MyJoin {

    public static int count = 0;

    public static synchronized void addCount () {
        count++;
    }

    public static void main (String args[]) {
        ArrayList<JoinThread> list = new ArrayList<JoinThread>();

        for (int i = 0; i < 5; i++) {
            JoinThread s = new JoinThread();
            list.add(s);
            s.start();
        }

        // join part
        for (JoinThread s : list) {
            try {
                s.join();
            } catch (Exception e) {
                System.err.println(e);
            }
        }

        System.out.println(count);
    }
}
```

- Multithreading Prime Counter in Java

```

public class PrimeCounter extends Thread {
    private int start;
    private int extent;
    private int count;

    public PrimeCounter (int s, int e) {
        start = s;
        extent = e;
        count = 0;
    }

    public boolean is_prime (int n) {
        if (n < 2) return false;
        if (n == 2) return true;
        if (n%2 == 0) return false;
        for (int i = 3; i < n; i += 2) {
            if (n % i == 0) return false;
        }
        return true;
    }

    public void run () {
        for (int i = start; i < start + extent; i++) {
            if (is_prime(i)) {
                count += 1;
            }
        }
    }

    public int count () {
        return count;
    }

    public static void main (String args[]) {
        int N = 1000000;

        PrimeCounter t1 = new PrimeCounter(0, N/2);
        PrimeCounter t2 = new PrimeCounter(N/2, N/2);
        PrimeCounter t3 = new PrimeCounter(0, N);

        long start = System.currentTimeMillis();

        t1.start();
        t2.start();
        try {
            t1.join();
            t2.join();
        } catch (Exception e) {
            System.err.println(e);
        }

        int count = t1.count() + t2.count();

        long end = System.currentTimeMillis();
        System.out.printf("Two threads count %d primes in %d ms\n", count, end-
start);
        System.out.printf("Two threads count %d primes in %d ms\n", count, end-start);
    }
}

```

```

        start = System.currentTimeMillis();

        t3.start();
        try {
            t3.join();
        } catch (Exception e) {
            System.err.println(e);
        }

        count = t3.count();

        end = System.currentTimeMillis();
        System.out.printf("One threads count %d primes in %d ms\n", count, end-
start);
    }
}

```

- Multithreading Prime Counter in Python

```

# PrimeCounter.py

import threading
import time

class PrimeCounter (threading.Thread):

    def __init__ (self, s, e):
        threading.Thread.__init__(self)
        self.begin = s
        self.extent = e
        self.count = 0

    def is_prime (self, n):
        if (n < 2):
            return False
        if (n == 2):
            return True
        if (n % 2 == 0):
            return False
        for i in range (3, n, 2):
            if (n % i == 0):
                return False;
        return True

    def run (self):
        for i in range (self.begin, self.begin + self.extent):
            if self.is_prime(i):
                self.count += 1

    def getCount (self):
        return self.count

```

```
def main():
    N = 100000

    pc1 = PrimeCounter(0, N//2)
    pc2 = PrimeCounter(N//2, N//2)
    pc3 = PrimeCounter(0, N)

    t1 = time.perf_counter()

    pc1.start()
    pc2.start()

    pc1.join()
    pc2.join()

    t2 = time.perf_counter()

    print("2 thread count ", pc1.getCount() + pc2.getCount(), " primes")
    print("time is ", t2 - t1)

    t1 = time.perf_counter()

    pc3.start()
    pc3.join()

    t2 = time.perf_counter()

    print("1 thread count ", pc3.getCount(), " primes")
    print("time is ", t2 - t1)

if __name__ == "__main__":
    main()
```