NWEN303 Concurrent Programming

Lab1: lets test concepts up to now!

Marco Servetto

VUW

Import code and complete it!

- Part 1:
  - Read the code in the next three slides, import it in a project in Eclipse and complete it.

```java
package code;
import java.util.List;import java.util.Random;import java.util.function.Supplier;
import java.util.stream.Collectors;import java.util.stream.Stream;
public class ForkJoinExercise {
  public static String id(String a,int iterations) {
    Random r=new Random();
    int x=r.nextInt(5);
    for(int i=0;i<=iterations*10000000;i++) {x+=r.nextInt(5);}
    if(x>100) {return a;}
    //TODO: try instead to comment all the above code and use the following
    //try {Thread.sleep(iterations*10);}
    //catch (InterruptedException e) {
    //  Thread.currentThread().interrupt();
    //  throw new Error(e);
    //  }
    return a;
    }
  public static String doA() { return id("A",1);}
  public static String doB() { return id("B",10);}
  public static String doC() { return id("C",2);}
  public static String doD() { return id("D",5);}
  public static String doAB(String a,String b) { return id(a+b,1);}
  public static String doCD(String c,String d) { return id(c+d,10);}
  public static String doAll(String ab,String cd) { return id(ab+cd,1);}
  @SafeVarargs public static <T> List<T> runInParallel(Supplier<T> ...ts){
    return Stream.of(ts).parallel()
      .map(f->f.get())
      .collect(Collectors.toList());
    }
  }
```

```java
package code;
import java.util.List;import static code.ForkJoinExercise.*;
public class Versions {
  public static String parallel1() {
    List<String>abcd=runInParallel(()->doA(),()->doB(),()->doC(),()->doD());
    String a=abcd.get(0);   String b=abcd.get(1);
    String c=abcd.get(2);   String d=abcd.get(3);
    List<String>res=runInParallel(()->doAB(a,b),()->doCD(c,d));
    return doAll(res.get(0),res.get(1));
    }
  public static String parallel2() {
    List<String>res=runInParallel(
      ()->{
        List<String>ab=runInParallel(()->doA(),()->doB());
        return doAB(ab.get(0),ab.get(1));
        },
      ()->{
        List<String>cd=runInParallel(()->doC(),()->doD());
        return doCD(cd.get(0),cd.get(1));
        });
    return doAll(res.get(0),res.get(1));
    }
  public static String sequential() {
    String a=doA();   String b=doB();
    String c=doC();   String d=doD();
    String ab=doAB(a,b);
    String cd=doCD(c,d);
    return doAll(ab,cd);
    }
  }
```

```java
@Test
void test() {
  //First, run the code just to trigger the JIT. How to run it?
  long t0 = System.currentTimeMillis();//take the time at start
  //run one version
  long t1 = System.currentTimeMillis();//take the time again
  //run another version... take the time again..
  //is there a better way?
  //run also the third version
  //print out results (only to give us a sense of accomplishment)
  System.out.println("Time Seq="+timeSeq);
  System.out.println("Time Par1="+timePar1);
  System.out.println("Time Par2="+timePar2);
  //assert the performance relationship
  assertTrue(timeSeq>timePar1);
  assertTrue(timeSeq>timePar2);
  assertTrue(timePar1>timePar2);
  }
```

Discussion

- Part 1:
  - Does it behave as we expected?

- Part 2:
  - From the slides, import the code of the reverseIndex and try to run such example. Try to use a very large text file.

    How much is the time for the index creation with respect of the time for the query?