```
package Algorithm;

import Items.Job;
import Items.Queue;

/**
 *
 * "Shortest time remaining first" is same as "shortest job first"
 * except it orders the jobs in the ready queue by the shortest remaining time
 * including the job that is getting processed by the CPU.
 *
 * A job in the ready queue will replace the job which currently processed
 * by the CPU if it has shorter remaining time.
 *
 * shortest time remaining first is a preemptive algorithm.
 */
public class STRF extends MyAlgorithm{

   /**
    * pass the work queue to super class to initialize lists
    * @param workQueue queue of lists to be worked on
    */
   public STRF(Queue workQueue)
   {
      super(workQueue);
   }

   /**
    * shows what happen in a single step when using this algorithm
    * @param simulationTime current time of this simulation
    * @return job the CPU was working on
    */
   @Override
   public Job nextStep (int simulationTime)
   {
      updateReadyQueue(simulationTime); // add newly arrived jobs to the ready queue
      /* add the current job back to the ready queue to compare remaining time
         with the newly added jobs to the ready queue */
      if(simulationTime!=0 && currentJob.getRemainTime() !=0)
       {readyQueue.addJob(currentJob); }
      if(readyQueue.size() > 1) {readyQueue.OrderedByShortRemain();} // order jobs in ready queue by
remaining time
      if(readyQueue.isEmpty()) {return null;}
      setCurrentJob();  // move the first job in the ready queue to be the current working job
      return workInCPU(simulationTime); //process the current job by the CPU
   }
}
```