Java Concurrency

Cs370 Help Session
By Sean Thunquest
Synchronization in Java

- Java has a built-in monitor
  - Allows threads to have mutual exclusion
  - Allows threads the ability to wait (block) for a condition to become true

- Extendable Thread class
  - Instantiate and myThread.start()
  - @Override run()

- Signaling support
  - wait()
  - notify()
public class PhilosopherThread extends Thread {
    @Override
    public void run() {
        //The entry point for each thread
        ...
    }
    ...
}
Creating and Starting Threads

PhilosopherThread Socrates = new PhilosopherThread(table, seat);
Socrates.start();

//begins our Socrates thread and invokes the run() method
Synchronized Methods

- A piece of logic marked with `synchronized` becomes a synchronized block, allowing only one thread to execute at any given time.

```java
public synchronized void pickup(int i) throws InterruptedException {
    ...
}
```
wait(), notify(), and notifyAll()

- **wait()**
  - Causes current thread to wait until another thread invokes the notify() or notifyAll() method.

- **notify()**
  - notify() wakes up one thread waiting for the lock.

- **notifyAll()**
  - The notifyAll() method wakes up all the threads waiting for the lock; the JVM selects one of the threads from the list of threads waiting for the lock and wakes that thread up.
COMPILER= javac
JRE= java
FILES= DiningPhilosophers.java PhilosopherTable.java PhilosopherThread.java
EXE= DiningPhilosophers
TAR= HW6.tar

all:  
   $(COMPILER) $(FILES)
run:  
   $(JRE) $(EXE)
clean:  
   rm *.class
package:  
   tar -cvf $(TAR) $(FILES) Makefile
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

public static String getTime(){
    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSSSSS");
    Date date = new Date();
    return dateFormat.format(date);
}

- If you use a package, verify that your makefile builds your program
  - Or don’t use a package.
Flow Chart of Round Robin

Ready Queue → CPU

Time Quantum over but not yet complete

Completed

IO Bound Wait
Round Robin Pseudocode

- Have a wait time and end time arrays where each element is set to -1
- Have an array for Gantt Chart where each element is a time unit and set each element to -1
- Execute while loop till all processes are complete
- For \(i\) = 0 to number of processes -1
  - For \(x\) = 0 to \(i\) - 1
    - if end time of \(x\) is less than arrival time of \(i\) and burst time of \(x\) is not zero set flag1 to false
  - For \(x\) = \(i\) + 1 to number of processes -1
    - if end time of \(i\) is greater than arrival time of \(x\) and end time of \(x\) is -1 set flag1 and flag2 to false
  - if arrival time of \(i\) is \(\leq \) current time and burst time of \(i\) is not 0 and flag1 is true
    - if wait time of \(i\) == -1
      - wait time of \(i\) = current time – arrival time of \(i\)
    - else
      - wait time of \(i\) = current time – end time of \(i\)
    - if burst time of \(i\) \(\leq \) q
      - set the Gantt Chart array values from current time to current time + burst time with the process id
      - change current time to current time + burst time of \(i\)
      - turnaround time of \(i\) = current time – arrival time of \(i\)
**Round Robin Pseudocode**

- set burst time of i to 0
- increment the number of processes to be completed by 1
- set end time of I to current time

**set flag2 to false**

- else
  - Reduce burst time of i by time quantum
  - set the Gantt Chart array values from current time to current time + time quantum with the process id
  - change current time to current time + time quantum
  - set end time of i to current time if it is less than that
  - set flag2 to false

- if flag1 is false make it true

- if flag2 is true
  - set Gantt Chart element at current time to -1 to indicate IDLE
  - increment current time by false

- else set flag2 to true