package Items;

import cpu.Simulation;

import javax.swing.table.AbstractTableModel;

/\*\*

\* This class is responsible for representing jobs data on the table in the GUI frame

\*/

public class MyTable extends AbstractTableModel{

private Queue tableQueue ; // job queue for the table

private String[] columnNames = {"#","Arrive","Burst","Priority","Start","Wait" , "Remain" ,"Finish" , "Turn" , "%"}; // table header

/\*\*

\* create new table

\* @param queue the queue of jobs to be represented in the table

\*/

public MyTable( Queue queue)

{

tableQueue = queue.getCopy();

this.fireTableRowsUpdated(1, 1);

}

/\*\*

\* @return number of rows in the table

\*/

@Override

public int getRowCount() {

return tableQueue.size(); // number of rows equals number of jobs in the queue

}

/\*\*

\* @return number of columns in the table

\*/

@Override

public int getColumnCount() {

return 10;

}

/\*\*

\* Calculate the average waiting time of all the jobs in the queue

\* @return average waiting time

\*/

public double getAverageWaiting()

{

double average = 0 ;

for(int i =0 ; i< tableQueue.size() ; i++)

{

average += (Integer) getValueAt(i, 5); // get 5th value of the table for every job

}

return (average/tableQueue.size());

}

/\*\*

\* Calculate the average turnaround time for all the jobs in the queue

\* @return

\*/

public double getAverageTurn()

{

double aveg = 0 ;

for(int i =0 ; i< tableQueue.size() ; i++)

{

aveg += (Integer) getValueAt(i, 8); // get the 8th value of the table for every job

}

return (aveg/tableQueue.size());

}

/\*\*

\* return the value of a specific place in the table

\* @param rowIndex row index of the wanted value

\* @param columnIndex column index of the wanted value

\* @return the wanted value at a specific row and column

\*/

@Override

public Object getValueAt(int rowIndex, int columnIndex) {

Job job = tableQueue.getJob(rowIndex);

switch(columnIndex)

{

case 0 : return job.jobNumber;

case 1 : return job.arrivalTime;

case 2 : return job.burst;

case 3 : return job.priority;

case 4 : return job.getStart();

case 5 : return job.getWaitTime(Simulation.Time);

case 6 : return job.getRemainTime();

case 7 : return job.getFinish();

case 8 : return job.getTurnaround(Simulation.Time);

case 9 : return job.getPercent();

default: return 0;

}

}

/\*\*

\* return the column's header

\* @param column column index

\* @return name of the header of the wanted column

\*/

@Override

public String getColumnName(int column)

{

return columnNames[column];

}

/\*\*

\* replace a specific job in the queue with another job

\* @param other the new job to replace with in the queue of the table

\*/

public void setValueAT(Job other)

{

int n = other.jobNumber;

for(int i=0 ; i<tableQueue.size() ; i++)

{

if(tableQueue.getJob(i).jobNumber == n)

{

tableQueue.set(i, other);

return;

}

}

}

}