Solutions for Assignment Unit 3

Tape.java

```
/*
A Turing machine works on a "tape" that is used for both input and output. The tape is made up of little squares called cells lined up in a horizontal row that stretches, conceptually, off to infinity in both directions. Each cell can hold one character. Initially, the content of a cell is a blank space. One cell on the tape is considered to be the current cell. This is the cell where the machine is located. As a Turing machine computes, it moves back and forth along the tape and the current cell changes.
A Turing machine tape can be represented by a doubly-linked list where each cell has a pointer to the previous cell (to its left) and to the next cell (to its right). The pointers will allow the machine to advance from one cell to the next cell on the left or to the next cell on the right. Each cell can be represented as an object of type Cell as defined by the class:
public class Cell {
public char content; // The character in this cell.
public Cell next; // Pointer to the cell to the right of this one.
public Cell prev; // Pointer to the cell to the left of this one.
}
This class is already defined in the file Cell.java, so you don't have to write it yourself.
_____
Your task is to write a class named Tape to represent Turing machine tapes. The class should have an instance variable of type Cell that points to the current cell. To be compatible with the classes that will use the Tape class, your class must include the following methods:
ɔ€¢ public Cell getCurrentCell() -- returns the pointer that points to the current cell.
ɔ€¢ public char getContent() -- returns the char from the current cell.
ɔ€¢ public void setContent(char ch) -- changes the char in the current cell to the specified value.
ɔ€¢ public void moveLeft() -- moves the current cell one position to the left along the tape. Note that if the current cell is the leftmost cell that exists, then a new cell must be created and added to the tape at the left of the current cell, and then the current cell pointer can be moved to point to the new cell. The content of the new cell should be a blank space. (Remember that the Turing machine's tape is conceptually infinite, so your linked list must must be prepared to expand on demand, when the machine wants to move past the current end of the list.)
ɔ€¢ public void moveRight() -- moves the current cell one position to the right along the tape. Note that if the current cell is the rightmost cell that exists, then a new cell must be created and added to the tape at the right of the current cell, and then the current cell pointer can be moved to point to the new cell. The content of the new cell should be a blank space.
ɔ€¢ public String getTapeContents() -- returns a String consisting of the chars from all the cells on the tape, read from left to right, except that leading or trailing blank characters should be discarded. The current cell pointer should not be moved by this method; it should point to the same cell after the method is called as it did before. You can create a different pointer to move along the tape and get the full contents. (This method is the hardest one to implement.)
It is also useful to have a constructor that creates a tape that initially consists of a single cell. The cell should contain a blank space, and the current cell pointer should point to it. (The alternative -- letting the current cell pointer be null to represent a completely blank tape -- makes all the methods in the class more difficult to implement.)
To test your Tape class, you can run the programs that are defined by the files TestTape.java, TestTapeGUI.java, and TestTuringMachine.java. The first two programs just do things with a tape, to test whether it is functioning properly. TestTuringMachine actually creates and runs several Turing machines, using your Tape class to represent the machines' tapes.
*/

package turing;

public class Tape {

private Cell currentCell; // Current cell pointer

public Tape() { //Constructor to create a blank tape with a single cell, which contains a blank space.
Cell newCell = new Cell();
newCell.content = ' ';
newCell.prev = null;
newCell.next = null;
currentCell = newCell;
}
```

```
public Cell getCurrentCell() { //The pointer to current cell.
return currentCell;
}

public char getContent() { //The content of current cell.
return currentCell.content;
}

public void setContent(char ch) { //ch The character to be set into the current cell.
currentCell.content = ch;
}

public void moveLeft() { //Moves the current cell one position to the left along the tape.
if (currentCell.prev == null) {
Cell newCell = new Cell();
newCell.content = ' ';
newCell.prev = null;
newCell.next = currentCell;
currentCell.prev = newCell;
}
currentCell = currentCell.prev;
}

public void moveRight() { //Moves the current cell one position to the right along the tape.
if (currentCell.next == null) {
Cell newCell = new Cell();
newCell.content = ' ';
newCell.next = null;
newCell.prev = currentCell;
currentCell.next = newCell;
}
currentCell = currentCell.next;
}

public String getTapeContents() { //Returns a String consisting of the chars from all the cells on the tape.
Cell pointer = currentCell;
while (pointer.prev != null)
pointer = pointer.prev;
String strContent = "";
while (pointer != null) {
strContent += pointer.content;
pointer = pointer.next;
}
strContent = strContent.trim(); //Returns a copy of the string, with leading and trailing whitespace omitted.
return strContent;
}
}
```

Last modified: Wednesday, 13 May 2020, 1:15 PM