Computer Science AP/X Quadtree

CSCI-140/242 Lab 8

10/21/2016

1 Requirements

1.1 Command Line

1.1.1 Compression

The compression main program is `RITCompress.java` and is run on the command line as:

```
$ java RITCompress.java in-file out-file
```

1.1.2 Uncompression

The uncompression main program is `RITUncompress.java` and is run on the command line as:

```
$ java RITUncompress.java data-file
```

1.2 File Format

Recall that the raw and compressed files come from the zip file you downloaded during the in-lab activity:

`https://www.cs.rit.edu/~csapx/Labs/Quadtree/code.zip`

They are stored in the `images` directory.

1.2.1 Raw Image File

If the raw file is present it is guaranteed to be correctly formatted but may not contain enough data to make the image square. The format of the file is the pixel values of the image going from left to right and top to bottom. Each grayscale value, `0-255`, is contained on its own line.

1.2.2 Compressed File

If the compressed file is present it is guaranteed to be correctly formatted and square in size. The only error that can occur is if the user attempts to `write` out a compressed file before calling `compress` on it. In this case the following should be displayed to standard error and the program should halt without creating the output file:

```
QTException: Error writing compressed file.  File has not been compressed.
```

1.3 Output

You can access sample outputs for both the compression and uncompression programs at:

`https://www.cs.rit.edu/~csapx/Labs/Quadtree/output`

1.3.1 Compress

When compressing the following information the following is displayed to standard output upon successful completion:

```
QTree: preorder-node-values
Raw image size: raw-size
Compressed image size: compressed-size
Compression %: ##.##
```

1. `preorder-node-values` are the values of the tree with a space between each.
2. `raw-size` is the size of the raw image in terms of total number of pixels.
3. `compressed-size` is the size of the compressed image.
4. `##.##` is the compression ratio, e.g. `compressed-size / (raw-size*100)`.

1.3.2 Uncompress

When the image is uncompressed the tree should be displayed to standard output:

```
QTree: preorder-node-values
```

Here `preorder-node-values` are the values of the tree with a space between each.

1.4 Error Handling

1.4.1 Compress

If the input file is not found, or can't be opened, it should print the following to standard error and exit (where {`in-file`} is the name of the file):

```
java.io.FileNotFoundException: in-file (No such file or directory)
```

If the output file can't be opened, it should print the following to standard error and exit (where {`out-file`} is the name of the file):

```
java.io.FileNotFoundException: out-file (Permission denied)
```

1.4.2 Uncompress

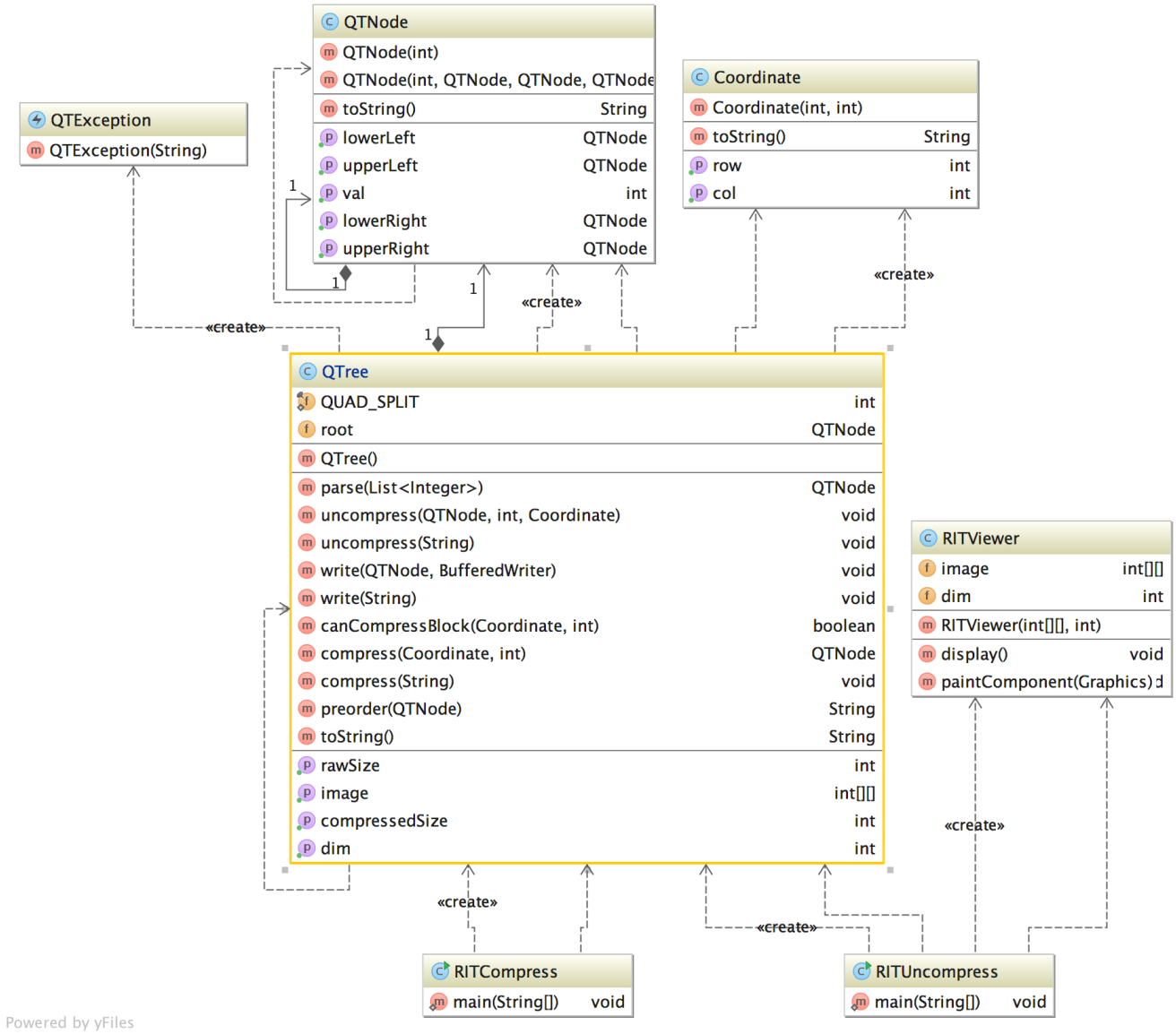
If the file is not found it should print the following to standard error and exit (where {`data-file`} is the name of the file):

```
java.io.FileNotFoundException: data-file (No such file or directory)
```

If there is not enough data when uncompressing the raw image file, the program should halt with the standard error message:

```
QTException: Error uncompressing.  Not enough data.
```

2 Design



For this lab we are giving you our suggested private state and methods for `QTree`. You are free to change this however you wish. Just make sure the signatures of the public methods are not altered.

3 Implementation

The suggested approach for implementing this project is

1. Complete implementing `QTree`'s public `uncompress` routine from the in-lab activity. The standard output should match the solution, and the image viewer should display the raw image.

2. Now implement `QTree`'s public `compress` routine. It should first read the raw data into the internal 2-D image array. Then you can write the recursive private `compress` routine. It uses a private helper method, `canCompressBlock` which checks that all the values in a provided region of the 2-D image are the same value or not. In the end, the main program will display the tree to standard output, so make sure it is correct.

3. Finally implement `QTree`'s public `write` routine. This should recursively traverse the tree that was build by `compress` and write the values to standard output. Do not forget to close the output file or else it could be incomplete or invalid!

3.1 Grading

The grade breakdown for this assignment is as follows:

- Problem Solving: 15%
- In-Lab Activity: 15%
- Functionality: 60%
  - Compression: 30%
  - Uncompression: 30%
- Code Style, Documentation and Version Control: 10%

3.2 Submission

3.3 `try` Submission

The `try` command for this lab is:

```
try csapx-grd lab8-1 QTree.java log.txt
```

These default files are automatically combined with your code when you submit through try.

Recall that to verify your latest submission has been saved successfully, you can run `try` with the query option, `-q`:

```
try -q csapx-grd lab8-1
```