Computer Science AP/X RIT — CSCI-140/242 Lab 6

10/3/2022

1 Requirements

1.1 Command Line

1.1.1 Uncompression

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

```
$ java RITUncompress input-file.rit output-file.raw
```

The run configurations for uncompressing use the `data/rit` directory for input file, and will generate the output file `generated/raw` directory.

Note that in the in-lab activity, writing the uncompressed file was not required, but it is for the full lab.

1.1.2 Compression

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

```
$ java RITCompress input-file.raw output-file.rit
```

The run configurations for compressing use the `data/raw` directory for input file, and will generate the output file `generated/rit` directory.

1.2 File Format

Recall that the expected raw and compressed files are stored in the `data/` directory.

1.2.1 Raw Image File (RITUncompress)

If the raw file is present it is guaranteed to be correctly formatted and contains the exact number of pixels that make up a power of 2. 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.

The only error that can occur when uncompressing is if the user attempt to `writeUncompressed()` to the uncompressed file before calling `uncompress()`. In this case the following should be displayed to standard error and the program should halt without creating the output file:

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

1.2.2 RIT File (RITCompress)

If the compressed file is present it is guaranteed to be correctly formatted and square in size. The first line is the total number of pixels, followed by the quadtree values (one per line). The only error that can occur is if the user attempts to `writeCompressed()` to the compressed file before calling `compress()`. 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 Standard Output

You can access expected standard outputs for both the compression and uncompression programs in your project `output/` directory.

1.3.1 RITUncompress

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.

1.3.2 RITCompress

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. `(1.0 - compressed-size / raw-size) * 100)`.

1.4 Error Handling

1.4.1 RITUncompress

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

```
java.io.FileNotFoundException: input-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.
```

1.4.2 RITCompress

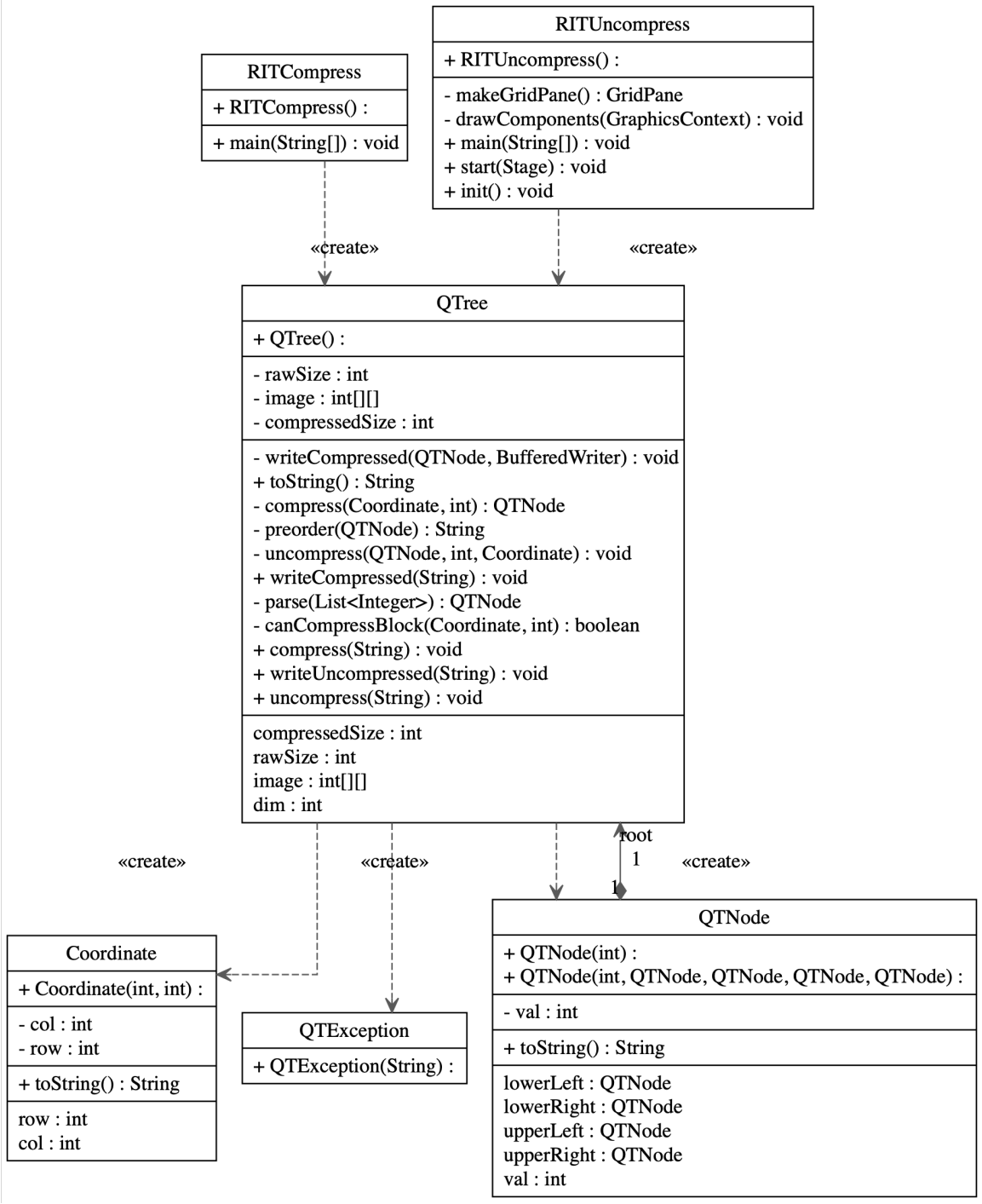
If the input file is not found, or can't be opened, it should print the following to standard error and exit (where {`input-file`} is the name of the file):

```
java.io.FileNotFoundException: input-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 {`output-file`} is the name of the file):

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

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. If not done already, 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 `writeUncompress()` routine. It should write the output file into the `generated/raw/` when using the run configurations.

3. 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.

4. Finally implement `QTree`'s public `writeCompressed()` routine. This should recursively traverse the tree that was build by `compress()` and write the values to standard output. When using the supplied run configurations, the output file will be stored in the `generated/rit/` folder.

3.1 Grading

The grade breakdown for this assignment is as follows:

- Problem Solving: 15%
- In-Lab Activity: 10%
- Functionality: 70%
  - Compression: 35%
  - Uncompression: 35%
- Code Style, Documentation and Version Control: 5%

3.2 Submission

You should zip up your `src` folder and name the file `lab6.zip`. Upload this zip file to the MyCourses Assignment dropbox by the due date.