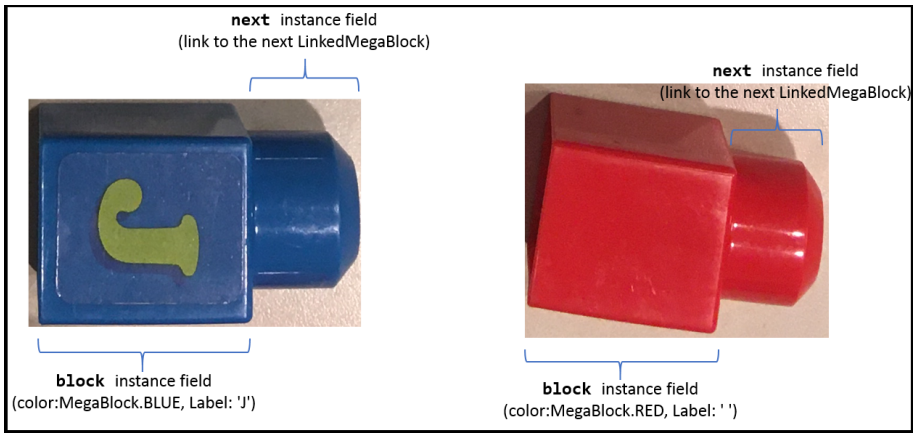
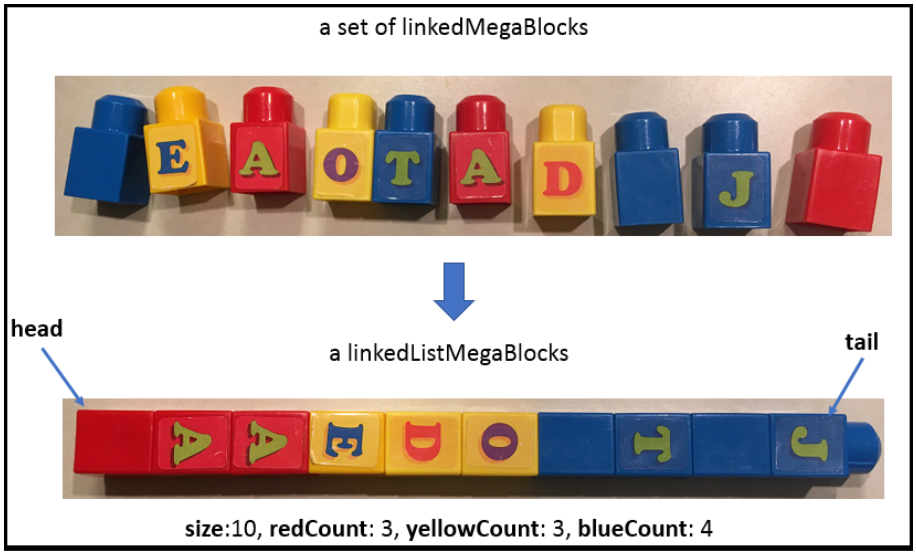
P06 Mega Blocks Builder

Overview

This assignment involves the implementation of a chain of mega blocks as a singly linked list from scratch. Figure 1 presents a graphic illustration of the objects that we are going to use in this program (a mega block, a linked mega block, and a linked list of mega blocks). Generally, little kiddos enjoy building chains and towers using those mega blocks. We note also that our `P06 Mega Blocks Builder Application` is not a graphic application.



(a) Linked Mega Blocks



(b) Linked List of Mega Blocks

Figure 1: Graphic Illustration of LinkedMegaBlock and a LinkedListMegaBlock objects

Learning Objectives

The goals of this assignment include:

- Implement a singly linked list of mega blocks from scratch.
- Further developing your experience working with object oriented design code and exception handling.
- Gain more experience with developing unit tests.

Grading Rubric

5 points	**Pre-Assignment Quiz:** The P6 pre-assignment quiz is accessible through Canvas before having access to this specification by 9:59PM on Sunday 10/20/2019. Access to the pre-assignment quiz will be unavailable passing its deadline.
20 points	**Immediate Automated Tests:** Upon submission of your assignment to Gradescope, you will receive feedback from automated grading tests about whether specific parts of your submission conform to this write-up specification. If these tests detect problems in your code, they will attempt to give you some feedback about the kind of defect that they noticed. Note that passing all of these tests does NOT mean your program is otherwise correct. To become more confident of this, you should run additional tests of your own.
15 points	**Additional Automated Tests:** When your manual grading feedback appears on Gradescope, you will also see the feedback from these additional automated grading tests. These tests are similar to the Immediate Automated Tests, but may test different parts of your submission in different ways.
10 points	**Manual Grading Feedback:** After the deadline for an assignment has passed, the course staff will begin manually grading your submission. We will focus on looking at your algorithms, use of programming constructs, and the style and readability of your code. This grading usually takes about a week from the hard deadline, after which you will find feedback on Gradescope.

Additional Assignment Requirements and Notes

- You MUST NOT add any fields either instance or static, and any public methods either static or instance to your `MegaBlock`, `LinkedMegaBlock`, and `LinkedListMegaBlock` classes, other than those defined in this write-up and these javadocs.
- DO NOT submit the provided `MegaBlock.java` and `Color.java` files on gradescope.
- You ARE NOT allowed to use a dummy node at the head of the LinkedListMegaBlock linked list.
- You CAN define local variables that you may need to implement the methods defined in this program.
- You CAN define private methods to help implement the different public methods defined in these javadocs, if needed.
- In addition to the required test methods, we HIGHLY recommend (not require) that you develop your own unit tests (public static methods that return a boolean) to convince yourself of the correctness of every behavior implemented in your `LinkedMegaBlock` and `LinkedListMegaBlock` classes with respect to these javadocs and the additional details provided in this write-up. Make sure to design the test scenarios for every method before starting its implementation. Make sure also to test all the special cases.
- Appendix A presents the set of exception classes that you may use while developing this assignment.
- Feel free to reuse the javadoc method headers provided in these javadocs in your class and method javadoc headers.

1 Getting Started

Start by creating a new Java Project in eclipse called P06 Mega Blocks Builder, for instance. You have to ensure that your new project uses Java 8, by setting the "Use an execution environment JRE:" drop down setting to "JavaSE-1.8" within the new Java Project dialog box. Then, add these provided Color.java, and MegaBlock.java source files to your project. The provided `MegaBlock` class represents the data type of the mega blocks that will be used in our program. Read carefully through the provided code and its javadoc method headers and try to familiarize yourself with it. You can notice that every megaBlock object has a color whose value cannot be changed once set (final field) and a label. The label of a megaBlock object is of type char. It can be any char including a space ' '. Three constants are defined in the `enum Color` to serve as possible colors for the megaBlock objects. These colors are Color.RED, Color.YELLOW, and Color.BLUE. In the next steps, you are going to create 3 classes and add them to this project.

2 Create the MegaBlockBuilderTester class

Now, create a new class called MegaBlockBuilderTester and add it to your project. You HAVE TO implement all the P06's test methods (the required ones and the ones that you may define in your own) in this class. The first test methods that you have to implement must have exactly the following signatures. They should check for the correctness of `MegaBlock.equals()` and `MegaBlock.toString()` methods with respect to their description provided in their javadoc method header.

```
public static boolean testMegaBlockEquals()
public static boolean testMegaBlockToString()
```

3 Create the LinkedMegaBlock class

Create a new class called `LinkedMegaBlock`. Each instance of this class represents a linked mega block (as graphically illustrated in 1.(a). Every LinkedMegaBlock object should have only the TWO following instance fields. You are NOT allowed to add any additional instance or static field to the `LinkedMegaBlock` class.

```
private MegaBlock block; // data field of this linkedMegaBlock
private LinkedMegaBlock next; // link to the next linkedMegaBlock
```

Now, implement the constructor and the public methods defined for the `LinkedMegaBlock` class according to their detailed javadocs description provided within these javadocs. We provide in the following additional details about the `LinkedMegaBlock.toString()` method. A LinkedMegaBlock object should be represented by the following String:

- block.toString() + " -> " // if next field is not null
- block.toString() + " -> END" // if next field is null

Next, create a test method inside your `MegaBlockBuilderTester` class with exactly the following signature. This method should test and make use of at least one LinkedMegaBlock constructor, an accessor (getter) method, and a mutator (setter) method.

```
public static boolean testLinkedMegaBlock()
```

4 Create the LinkedListMegaBlock class

Now, create a new class called `LinkedListMegaBlock` and add it to your P06 project source folder. This class models the singly linked list data structure that stores elements of type `MegaBlock`. This class MUST define the following instance fields.

```
private LinkedMegaBlock head; // head of this list
private LinkedMegaBlock tail; // tail of this list
private int size; // size of this list (total number of megablocks stored in this list)
private int redCount; // number of RED megablocks stored in this list
private int yellowCount; // number of YELLOW megablocks stored in this list
private int blueCount; // number of BLUE megablocks stored in this list
```

You are not allowed to add any additional instance or static field to this class. Recall also that you ARE NOT allowed to use a dummy node at the head of the LinkedListMegaBlock linked list.

Now, make sure to implement the constructor and all the methods defined in this class with respect to the specification provided in these javadocs. We note that our list of mega blocks will be implemented using linked mega blocks. Our linked list can store only mega blocks having one of three colors defined in the `Color` enum only (Color.RED, Color.YELLOW, and Color.BLUE). Adding, replacing, or removing mega blocks to/from the list must obey to the following rules. Further details are provided in these javadocs.

- RED mega blocks can be added only at the head of the list.
- RED mega blocks can be removed only from the head of the list.
- BLUE mega blocks can be added only at the end (tail) of the list.
- BLUE mega blocks can be removed only from the end (tail) of the list.
- YELLOW mega blocks can be added at any position of the list, but AFTER all the already added RED mega blocks and BEFORE all the already added BLUE mega blocks, as illustrated in Fig. 1.
- YELLOW mega blocks can be removed given their position in the list.
- A mega block within the list can be replaced with ONLY another mega block with the same color, independently of the value of its label.

We note also the `LinkedListMegaBlock.toString()` method should return the following String.

- An empty String “”; if the list is empty.
- A string with each mega block String representation in the list separated by “ -> ”s, and ending with “ -> END”; if the list is not empty.

You are HIGHLY encouraged to add additional test methods to your `MegaBlockBuilderTester` class to gain confidence that are the methods defined in your `LinkedListMegaBlock` are working correctly with respect to these javadocs the details provided above. All your test methods MUST return a boolean (true if the expected behavior is satisfied and false otherwise). In particular, create and add the following two test methods with exactly the following signatures to your `MegaBlockBuilderTester` class.

```
// checks for the correctness of the LinkedListMegaBlock.addRed() method
public static boolean testLinkedMegaBlockListAddRed()

// checks for the correctness of the LinkedListMegaBlock.removeBlue() method
public static boolean testLinkedListMegaBlockRemoveBlue()
```

Illustrative Example

In order to provide you with a better understanding on how to use the implemented classes, we provide in the following an example of source code and its expected output, when the methods are called with correct input arguments.

```
/**
 * Helper method to display the contents of a list of mega blocks
 * @param list a reference to a LinkedListMegaBlock object
 * @throws NullPointerException if list is null
 */
private static void display(LinkedListMegaBlock list) {
  // display list content
  System.out.println("list content: " + list);
  // display information about the size of this list and the its blocks' color counts
  System.out.println("Size: " + list.size() +
      ", redCount: " + list.getRedCount() +
      ", yellowCount: " + list.getYellowCount() +
      ", blueCount: " + list.getBlueCount());
  System.out.println();
}

/**
 * Driver method to show how to use the implemented classes in P06 Mega Blocks Builder
 * @param args input arguments
 */
public static void main(String[] args) {
  // Create a new empty LinkedListMegaBlocks list
  LinkedListMegaBlock list = new LinkedListMegaBlock();
  // display list's content and size information
  display(list);
```

```
  // Add some blocks to list and display its contents and size information
  list.addBlue(new MegaBlock(Color.BLUE, 'C')); // add a blue mega block
  display(list);
  list.addBlue(new MegaBlock(Color.BLUE, 'S')); // add a blue mega block
  display(list);
  list.addYellow(0,new MegaBlock(Color.YELLOW, 'Y')); // add a yellow mega block
                                                      // at index 0 of this list
  display(list);
  list.addRed(new MegaBlock(Color.RED, 'A')); // add a red mega block to this list
  list.addRed(new MegaBlock(Color.RED, 'B')); // add a red mega block to this list
  list.addYellow(3, new MegaBlock(Color.YELLOW, 'H')); // add a yellow mega block
                                                       // at index 3 of this list
  display(list);
  // remove/add some blocks and display the list after each operation
  list.removeBlue(); // remove a blue block
  display(list);
  list.removeBlue(); // remove a blue block
  display(list);
  // add a yellow block at the end of list (the list contains zero blue blocks)
  list.addYellow(list.size(), new MegaBlock(Color.YELLOW, 'W'));
  display(list);
  list.removeRed(); // remove a red block
  display(list);
  list.removeYellow(list.size()-1); // remove the yellow block at the end of list
  display(list);
}
```

Expected output:

```
list content:
Size:  0, redCount:  0, yellowCount:  0, blueCount:  0

list content:  BLUE C -> END
Size:  1, redCount:  0, yellowCount:  0, blueCount:  1

list content:  BLUE C -> BLUE S -> END
Size:  2, redCount:  0, yellowCount:  0, blueCount:  2

list content:  YELLOW Y -> BLUE C -> BLUE S -> END
Size:  3, redCount:  0, yellowCount:  1, blueCount:  2

list content:  RED B -> RED A -> YELLOW Y -> YELLOW H -> BLUE C -> BLUE S
-> END
Size:  6, redCount:  2, yellowCount:  2, blueCount:  2

list content:  RED B -> RED A -> YELLOW Y -> YELLOW H -> BLUE C -> END
Size:  5, redCount:  2, yellowCount:  2, blueCount:  1

list content:  RED B -> RED A -> YELLOW Y -> YELLOW H -> END
Size:  4, redCount:  2, yellowCount:  2, blueCount:  0

list content:  RED B -> RED A -> YELLOW Y -> YELLOW H -> YELLOW W -> END
Size:  5, redCount:  2, yellowCount:  3, blueCount:  0

list content:  RED A -> YELLOW Y -> YELLOW H -> YELLOW W -> END
Size:  4, redCount:  1, yellowCount:  3, blueCount:  0

list content:  RED A -> YELLOW Y -> YELLOW H -> END
Size:  3, redCount:  1, yellowCount:  2, blueCount:  0
```

5 Assignment Submission

Congratulations on finishing this CS300 assignment! After verifying that your work is correct, and written clearly in a style that is consistent with the course style guide, you should submit your final work through gradescope.com. The only 3 files that you must submit include: LinkedMegaBlock.java, LinkedListMegaBlock.java, and MegaBlockBuilderTester.java. Your score for this assignment will be based on your **"active"** submission made prior to the hard deadline of Due: **9:59PM on October 23**rd. The second portion of your grade for this assignment will be determined by running that same submission against additional offline

automated grading tests after the submission deadline. Finally, the third portion of your grade for your submission will be determined by humans looking for organization, clarity, commenting, and adherence to the course style guide.

Appendices

A Appendix A

IllegalArgumentException
NullPointerException
NoSuchElementException
IndexOutOfBoundsException