データ構造とアルゴリズム実験レポート
課題２：<Java によるプログラミングの復讐>

<201811320><1 クラス><江頭輝>

締切日：<2019 年 4 月 22 日>
提出日：<2019 年 4 月 27 日>

1　必須課題 2-1

この課題では，セルに整数値を格納する連結リストを実現するクラス List を Java 言語で実装する。

1.1　List.java の作成

1.1.1　実装の方針

まず, List クラスを定義する。セルの直後（ポインタは）‘p.next‘のような形で表現し、セルの値は ‘p.data‘のように表記する。

1.1.2　実装コードおよびコードの説明

1.1.3　実装の方針

insert_cell と insert_cell_top では即席変数である new_cell を作成しそのデータに引数を渡すことで新たなセルを作成する。top の場合はのポインタにこれまでの textttthead をよび、前者では引数として渡したのポインタがのポインタになるようにする。delete_cell ではのポインタに p のポインタのポインタを代入することでセルの削除を行うようにしている。delete_cell_top ではではなく textttthead になった。display ではのポインタが null になるまで値をループで出力する。

1.1.4　実行結果

まず，List.java を以下のコマンドでコンパイルする.

```
-------------------------------------
$ javac List.java
-------------------------------------
```

コンパイル後に以下のコマンドで実行を行った。

```
-------------------------------------
$ java List
```

```java
// List.java
public class List {
    static List head;
    List next;
    int data;

    // 新たなリストを生成 p => new_cell => p.next
    static void insert_cell(List p, int d) {
        List new_cell = new List();
        new_cell.data = d;
        new_cell.next = p.next;
        p.next = new_cell;
    }

    // head = new_cellデータは(d) => old_head
    static void insert_cell_top(int d) {
        List new_cell = new List();
        new_cell.data = d;
        new_cell.next = head;
        head = new_cell;
    }

    // p.next を p.next.next で上書きしてp.を削除next
    static void delete_cell(List p) {
        List q = p.next;
        p.next = q.next;
    }

    // head を head.next で上書き
    static void delete_cell_top() {
        List q = head;
        head = q.next;
    }

    // head のコピーを.で終わりまでループしてそれぞれのデータを表示next
    static void display() {
        List tmp = head;
        while (tmp != null) {
            System.out.print(tmp.data);
            tmp = tmp.next;
        }
        // 改行
        System.out.println();
    }

    // List クラスの動作確認
    public static void main(String[] args) {
        insert_cell_top(1);
        insert_cell(head, 3);
        insert_cell(head, 2);
        display();

        delete_cell(head);
        display();

        delete_cell_top();
        display();
    }
}
```

図 1　List.java のソースコード

```
123
13
3
---------------------------------------
```

main メゾッドでまず、1 を先頭に追加、先頭の直後に 3, 2 を順番に追加していった。なので、43 行目の display 関数で出力っされる値は 123 である。次に head セルの次のセルを削除したので 46 行目の display

関数では 13 が出力される。最後に `delete_cell_top` 関数を呼び出し、先頭のセルを削除したため、49 行目での `display` 関数では 3 のみが出力された。

1.1.5 考察

このコードだと、head の先頭にセルを追加する場合新しいリストを作る必要があります。また、head のみ仕様が異なるので、余分に `insert_cell_top` メゾッドと `delete_cell_top` メゾッドを呼ぶ必要がありました。それを踏まえると、ダミーのデータ部を持つ head を作成するほうが実装が単純になると考えられます。

2 必須課題 2-2

2.1 時間計算量

2.2 QueueArray.java の作成

2.2.1 実装の方針

実装の方針は，

2.2.2 実装コードおよびコードの説明

図 2QueueArray.java のソースコードを示す．2.2.1 節で述べた，`recursive` メソッドと `main` メソッドは，それぞれ 19〜31 行目，2〜16 行の部分に相当する．今回はデバッグのスピードを速めるために、コマンドライン引数がない場合は n に 13673 を m に 31640 を代入させるようにした。本番環境ではコメントアウトすることが望ましい。`recursive` メゾッドは再帰メゾッドなのではじめに終了条件の m がゼロの時を書き、そこで n を最大公約数として返すようにしている。24〜28 行目で入れ替えを行った後、r に n と m のあまりを代入して、m と r で再帰させている。

2.2.3 実行結果

1.1.4 節で述べたコンパイル方法と同様に，QueueArray.java を `javac` コマンドでコンパイルし，QueueArray.class を生成する．その後，以下のコマンドを入力することによって，QueueArray.java のプログラムを実行する．

```
-------------------------------------
$ java -ea QueueArray 57 76
The GCD of 76 and 57 is 19.
-------------------------------------
```

2.2.4 考察

30 行目の再帰させるとき n のところに m, m のところに r を引数とすることで、前回の List.java で行った値の最代入を省略している。それにより、計算量は同じ `O(log(n)` だが多少の改善が行われたと思う。

```java
public class QueueArray {
    int length, front, rear;
    int[] queue;

    // 指定された長さの配列を生成するコンストラクタ
    QueueArray(int len) {
        queue = new int[len];
        length = len;
        front = 0;
        rear = 0;
    }

    // queue の後ろに引数を代入する
    void enqueue(int val) {
        if (rear > length-1) {
            System.err.println("Queue␣Overflow␣!!");
            System.exit(1);
        }
        queue[rear] = val;
        rear += 1;
    }

    // queue の先頭を返し、番号を一つずらす
    int dequeue() {
        if (front == rear) {
            System.err.println("Queue␣Underflow␣!!");
            System.exit(1);
        }
        int x = queue[front];
        front += 1;
        return x;
    }

    // queue の先頭から末尾までを出力
    void display() {
        for (int i = front; i < rear; i++) {
            System.out.print(queue[i]);
        }
        System.out.println();
    }

    // QueueArray クラスの動作確認
    public static void main(String[] args) {
          QueueArray queue = new QueueArray(10);

          queue.enqueue(1);
          queue.enqueue(2);
          queue.display();                          // 12

          System.out.println(queue.dequeue());     // 1
          System.out.println(queue.dequeue());     // 2
    }
}
```

図 2　QueueArray.java のソースコード