提出日　令和2年06月24日

アルゴリズムとデータ構造　第4回課題レポート

学籍番号（A19117）
氏名（永尾優磨）

【課題】

1. スタックとキューについて調べ、その違いについて説明せよ。
2. スタックについて以下の作業を行い、ソースリストと実行結果をレポートにまとめよ。
   - Stackインスタンスを作成せよ。
   - データを適当な数(5～6個程度)スタックに投入せよ。
   - 投入したデータをすべて表示させよ。(投入したのと逆順で取り出される事に留意せよ。)
3. キューについて以下の作業を行い、ソースリストと実行結果をレポートにまとめよ。
   - Queueインスタンスを作成せよ。
   - データを適当な数(5～6個程度)キューに投入せよ。ただし、ラップアラウンド処理にならない程度とする
   - いくつかキューからデータを取り出して表示させてみよ。
   - さらにいくつかデータをとうにゅうせよ。今度はラップアラウンド処理が行われるようデータを投入してみよ。
   - 投入したデータをすべて表示させよ。

【作成したプログラム】

```
public class StackApp {

    public static void main(String[] args) {
        Stack obj = new Stack(5);
        obj.push(2);
        obj.push(3);
        obj.push(1);
        obj.push(5);
        obj.push(6);
        for (int z = 0; z < 5; z += 1) {
            System.out.println(obj.peek());
            obj.pop();
        }

    }


}
class Stack {

    // 属性
    private int maxSize;
    private int[] stackArray;
```

```
    private int top;

    // メソッド
    public Stack(int n) {
        maxSize = n;
        stackArray = new int[maxSize];
        top = -1;
    }

    public void push(int dt) {
        stackArray[++top] = dt;
    }

    public int pop() {
        return stackArray[top--];
    }

    public int peek() {
        return stackArray[top];
    }

    public boolean isEmpty() {
        return (top == -1);
    }

    public boolean isFull() {
        return (top == maxSize - 1);
    }

}
```

プログラム1.1　スタックのソースコード1

```
package 第04回スタックとキュー;

public class StackApp {

    public static void main(String[] args) {
        Stack obj = new Stack(5);
        obj.push(2);
        obj.push(3);
        obj.push(1);
        obj.push(5);
        obj.push(6);

        // isEmptyの返り値が空では無いとき、Trueなので、無限ループにするために！を付けて無限ループにした
```

```
        while (!obj.isEmpty()) {
            System.out.println(obj.peek());
            obj.pop();
        }

    }

}
   以下略
```

プログラム1.2　スタックのソースコード2

```
public class QueueApp {

    public static void main(String[] args) {
        Queue obj = new Queue(5);
        obj.enqueue(2);
        obj.enqueue(3);
        obj.enqueue(1);
        obj.enqueue(5);
        obj.enqueue(6);

        for (int z = 0; z < 3; z += 1) {
            System.out.println(obj.peek());
            obj.dequeue();
        }
        // 差をわかりやすくするために、一文を入れる。
        System.out.println("-------------------------");
        obj.enqueue(2);
        obj.enqueue(3);
        obj.enqueue(1);
        obj.enqueue(722);

        // isEmptyの返り値が空では無いとき、Trueなので、無限ループにするために！を付けて無限ループにした
        while (!obj.isEmpty()) {
            System.out.println(obj.peek());
            obj.dequeue();
        }
    }

}

class Queue {
    // 属性
    private int maxSize;
    private int[] queueArray;
    private int front;
    private int rear;
```

```
    private int nItems;

    // メソッド
    public Queue(int n) {
        maxSize = n;
        queueArray = new int[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }

    public void enqueue(int dt) {
        if (rear == maxSize - 1) {
            rear = -1;
        }

            queueArray[++rear] = dt;
            nItems++;



    }

    public int dequeue() {
        int dt = queueArray[front++];
        if (front == maxSize) {
            front = 0;
        }
        nItems--;
        return dt;
    }

    public int peek() {
        return queueArray[front];
    }

    public boolean isEmpty() {
        return (nItems == 0);
    }

    public boolean isFull() {
        return (nItems == maxSize);
    }

}
```

プログラム2.1　キューのソースコード1

```
public class QueueApp {

    public static void main(String[] args) {
        Queue obj = new Queue(5);
        obj.enqueue(2);
        obj.enqueue(3);
        obj.enqueue(1);
        obj.enqueue(5);
        obj.enqueue(6);

        for (int z = 0; z < 3; z += 1) {
            System.out.println(obj.peek());
            obj.dequeue();
        }
        // 差をわかりやすくするために、一文を入れる。
        System.out.println("--------------------------");
        System.out.println(obj.isFull());
        obj.enqueue(2);
        obj.enqueue(3);
        obj.enqueue(1);
        obj.enqueue(722);

        // isEmptyの返り値が空では無いとき、Trueなので、無限ループにするために！を付けて無限ループにした
        while (!obj.isEmpty()) {
            System.out.println(obj.peek());
            obj.dequeue();
        }
    }

}
中略
class Queue {
    // 属性
    private int maxSize;
    private int[] queueArray;
    private int front;
    private int rear;
    private int nItems;

    // メソッド
    public Queue(int n) {
        maxSize = n;
        queueArray = new int[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
```

```
    }

    public void enqueue(int dt) {
        if (rear == maxSize - 1) {
            rear = -1;
        }
// iキューの領域の値が、最大の時に、値を入力できない仕様にした。
        if (!isFull()) {
            queueArray[++rear] = dt;
            nItems++;

        }

    }
以下略
```

プログラム2.2　キューのソースコード2

```
public class QueueApp {

    public static void main(String[] args) {
        Queue obj = new Queue(5);
        obj.enqueue(2);
        obj.enqueue(3);
        obj.enqueue(1);
        obj.enqueue(5);
        obj.enqueue(6);

        for (int z = 0; z < 3; z += 1) {
            System.out.println(obj.peek());
            obj.dequeue();
        }
        // 差をわかりやすくするために、一文を入れる。
        System.out.println("-------------------------");
        System.out.println(obj.isFull());
        obj.enqueue(533);
        obj.enqueue(35);
        obj.enqueue(51);
        obj.enqueue(722);

        // isEmptyの返り値が空では無いとき、Trueなので、無限ループにするために！を付けて無限ループにした
        while (!obj.isEmpty()) {
            System.out.println(obj.peek());
            obj.dequeue();
        }
    }

}
```

以下省略

プログラム2.3　キューのソースコード3

【プログラムの解説】

1. スタックプログラムの解説

【プログラムの流れ】

1，配列に数字を追加する
2，追加した数字を表示する
3，表示した数字を削除する
4，2，3を数字が無くなるまで、繰り返す
という流れでプログラムが書かれている。

実行するためのStackAppとStackのクラスを分けた。Stackの引数に数字を入れることで、配列の長さが決まるようにしてある。

工夫点としては、元々プログラム1.1のようにfor文を使って、4のループを回していたが、プログラム1.2のようにWhile文で回せるようにしたことがある。
そうすることで、配列の長さがわかったときにも、対応できるようにしてある。

2. キュープログラムの解説

【プログラムの流れ】

1，配列に数字を追加する
2，for文を用いて、配列内の数字を減らす
3，追加した数字がわかるようにするために、線の出力をする
4，配列に数字を追加する
5，数字を表示する
6，表示した数字を削除する
7，5，6を数字が無くなるまで繰り返す
という流れでプログラムを書いている。

実行するためのQueueAppとQueueクラスを分けて記述した。Queueの引数に数字を入れることで、配列の長さが決まるようにしてある。

プログラム1.1、プログラム1.2で得た経験をもとにfor文とWhile文の両方を用いて、効率的に記述できるようにした。

【追記1】
「プログラム2.2」へいただいたフィードバックを基に、enqueue()内に、if文を追加した。出力結果は、図2.2に掲載した結果となった。これを付け加えたことで、配列が最大値になっているときは、値を追加出来ないようにした。~~出力は、図2.2参照~~
【追記2】

「プログラム2.3」へいただいたフィードバックを基に、入れる値を変更した。出力結果は、図2.3に掲載した結果となった。
キューの領域以上のデータを入力しようとすると、データはキューへ格納されずに消滅する仕様となっている。

【結果】

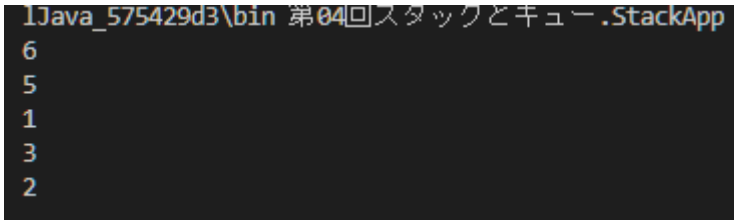

図1　スタックプログラムの出力結果

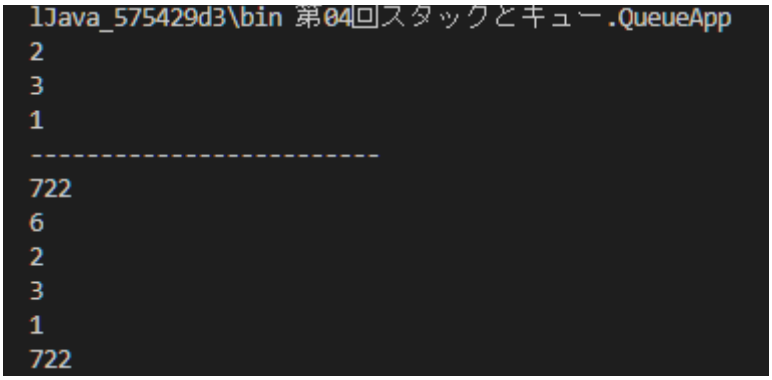

図2.1　キュープログラムの出力結果

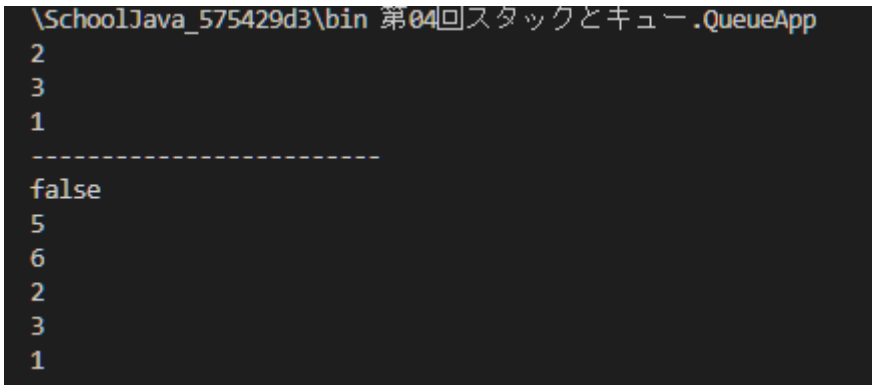

図2.2　キュープログラムの出力結果

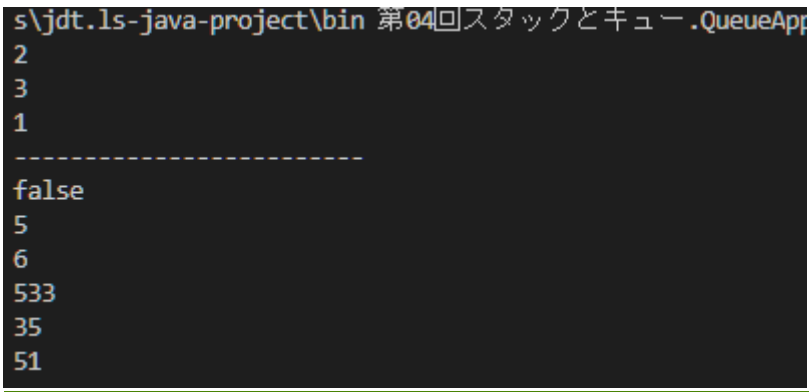

図2.3　キュープログラムの出力結果

【考察】

1. スタックとキューについての違い

スタックは、データを上から積んでいき、上から取り出すというデータ配列のこと。

スタックを例えると、積み木のようなものである。積み木は、上から積むことが出来るが、崩さないようにしたから取り出すのは至難の業である。そのため例えるなら、積み木のようなデータ配列である。

キューは、データを上から積んでいき、下から取り出すというデータ配列のこと。

キューを例えると、チケットの販売所である。チケットの販売所は、後ろから人が増えていき、前から人が減っていく。そのため例えるなら、チケットの販売所である。

2. スタックプログラムの考察

スタックプログラムは、最初for文で、削除をしていた。しかしそのままだと、Stack()に入れる引数を変更すると、for文の回数を変更しなければ、入っている中身を全て取り出すことが出来なくなる。そのため、Stack()の引数が変わったとしても、While文を用いる事で、全てを取り出せるようにした。

関数を実際に書いてみたことで、簡単にスタックを記述できることが分かった。

ライブラリ内で今まで、自動的にやっていたことが、記述出来るようになったため、理解が深まったと感じる。

3. キュープログラムの考察

Queueインスタンスにenqueueで追加していき、Queue()の引数に入れた数値を超えると、値が上書きされることがわかった。これは、配列の末尾に追加しようとしたが、配列が存在しないため、最初の文字に値が上書きされてしまったために生じる現象だと考えられる。

このような問題が実際の開発現場で起こってしまった場合、実行は出来るが、バグの原因となってしまうことが予想される。

【参考文献】

1. 「アルゴリズムとデータ解析の授業スライド、スタックとキューについて」
2. 増井敏克(2020)「Pythonで始めるアルゴリズム入門」翔泳社
3. 「参考文献の書き方」〈http://www7a.biglobe.ne.jp/nifongo/ron/ron_04.html〉