Veri Yapıları Laboratuvar Föyü
Hafta 2
Yığıt İşlemleri ve Postfix Notasyonu

Şevket Umut ÇAKIR

1 Giriş

Bu deneyde java.util.Stack sınıfını kullanarak iki yığıtın eşitliğini test eden, yığıtı tersine çeviren, postfix notasyonundaki ifadelerin değerlendirilmesi ve infix notasyonundaki ifadelerin postfix notasyonuna dönüştürülmesi amaçlanmaktadır.

2 Konu Anlatımı ve Deney Hazırlığı

2.1 Java Util Paketindeki Stack Sınıfı

Java Util paketinde generic olarak tanımlanan bir Stack sınıfı bulunmaktadır. Bu Stack sınıfının kullanımı derste gerçekleştirilen MyStack sınıfı ile çok benzerdir ve örnek bir uygulama aşağıdadır.

```
import java.util.Stack;
public class StackOrnek {
    public static void main(String[] args) {
        Stack<Integer> s=new Stack<>();
        s.push(1);
        s.push(7);
        s.push(5);
        s.push(21);
        int toplam=0;
        while (!s.isEmpty()){
            toplam+=s.peek();
            System.out.println(s.pop());
        }
        System.out.println("Toplam="+toplam);
    }
}
```

2.2 Infix İfadeyi Sembollerine Ayırma

Infix biçiminde verilen metin sembollerine aşağıda splitToTokens metodunda olduğu gibi StringTokenizer sınıfı kullanılarak ayrılabilir. Bu sembollere dönüştürme işleminde işaret belirten - ve + sembolleri dikkate alınmamaktadır(-12 gibi). Bunun için daha karmaşık bir çözümlemeye ihtiyaç vardır.

```
private static String[] splitToTokens(String girdi)
{
    StringTokenizer t=new StringTokenizer(girdi, "+-*/^() ",true);
    List<String> tokenList=new ArrayList<>();
    while (t.hasMoreTokens()){
        String s=t.nextToken().trim();
        if(!s.equals(""))
            tokenList.add(s);
    }
    String [] tl=new String[tokenList.size()];
    tokenList.toArray(tl);
    return tl;
}
```

Postfix ifadeleri değerlendirmek için aşağıdaki algoritma kullanılabilir.

İşlenenler(operands) için bir yığıt(stack) oluştur
while *Girdi metninde değer olduğu sürece d değerini oku* **do**
  if *okunan değer(d) sayı ise* **then**
    Değeri yığıta it
  else
    Yığıttan d1 ve d2 değerlerini çek
    d2 işlem d1 yap
    Sonucu yığıta it
  end
end
Yığıttaki tek değer sonucu verir

Algorithm 1: Postfix değerlendirme algoritması

Infix ifadeleri postfix'e dönüştürmek için aşağıdaki algoritma kullanılabilir.

İşlemler için opstack yığıtını ve çıktı için output listesinioluştur
Girdi metnini sembollerine ayır
while *Girdi sembollerini soldan sağa oku(s)* **do**
  if *Okunan sembol sayı ise* **then**
    Sembolü çıktı listesine ekle
  else if *Sembol parantez açma ise "(" ise* **then**
    Sembolü opstack yığıtına it
  else if *Sembol parantez kapama ise ")" ise* **then**
    opstack yığıtındaki "(" gelene kadar yığıttan elemanları çek ve çıktı listesine ekle
  else
    `// Sembol işlem ise(+,-,*,/)`
    opstack yığıtında önceliği sembolden(s) daha büyük olan işlem olduğu sürece yığıttan çek ve çıktı listesine ekle
    Sembolü(s) opstack yığıtına it
  end
end
opstack yığıtında kalan tüm işlemleri çek ve çıktı listesine ekle
Çıktı listesi postfix ifadeyi verir

Algorithm 2: Infix ifadeyi Postfix'e dönüştürme

3 Deneyin Uygulanması

Deneyde iskelet olarak size verilen `Islemler` sınıfının 4 adet metodunun yazılması istenmektedir. Bu uygulamada main metodu yazılmasına, metodlar `static` olduğu için de `Islemler` sınıfından bir nesne oluşturmaya gerek yoktur.

İlk olarak `public static <T> Stack<T> tersCevir(Stack<T> s)` yapısında olan tesineCevir metodu parametre olarak gönderilen yığıtın elemanlarının sırasını tersine çevirmelidir. Parametre olarak gönderilen yığıtta değişiklik yapılmamalıdır. Yığıtların bir kopyasını oluşturmak için `clone()` metodu kullanılabilir.

İkinci olarak `public static <T> boolean esit(Stack<T> s1, Stack<T> s2)` yapısında olan `esit` metodunun yazılması istenmektedir. `esit` metodu parametre olarak gönderilen iki yığıtın içindeki elemanların eşit olup olmadığını test eder. `esit` metoduna parametre olarak gelen yığıtların içeriği değiştirilmemelidir.

Üçüncü olarak `public static int postfixDegerlendir(String girdi)` yapısındaki postfix ifadeleri değerlendiren metodun yazılması istenmektedir. Bu metodda `java.util.Stack<Integer>` tipinde bir nesne kullanılmalıdır. Yapılan bölme işlemleri tamsayı bölmesi olmalıdır.

Dördüncü olarak da `public static String infixToPostfix(String girdi)` yapısındaki infix ifadeleri postfix ifadelere dönüştüren Algoritma 2'deki yöntemle metodun yazılması istenmektedir. Girdi metnini sembollerine ayırmak için `splitToTokens` metodu kullanılabilir.

4 Deney Soruları

- Bir infix ifadeye karşılık gelen birden fazla postfix ifade olabilir mi? Eğer olabilirse Algoritma 2'deki yöntemin sonucunda farklı postfix ifadeler ortaya çıkma ihtimali var mıdır?

Kaynaklar

[1] Infix, Prefix ve Postfix Expressions,

`http://interactivepython.org/runestone/static/pythonds/BasicDS/InfixPrefixandPostfixExpressions.html`