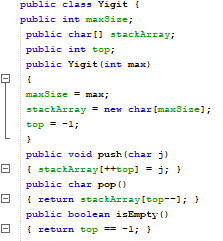
**YIĞIT UYGULAMALARI**

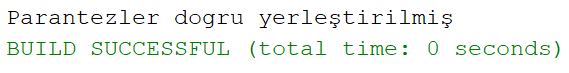
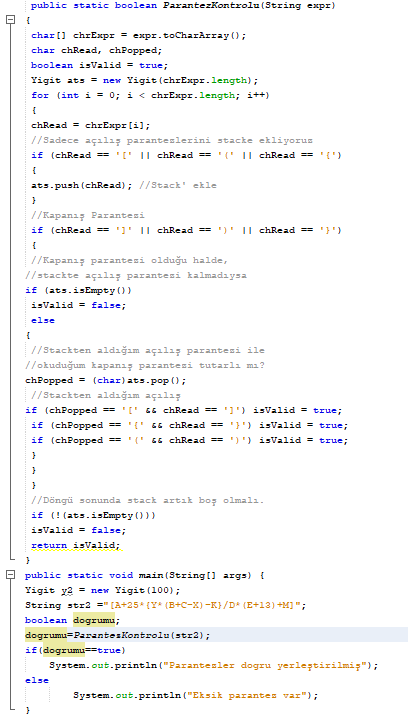
Yığıtlar eleman ekleme ve çıkarmanın en üst indisten yapıldığı veri yapılarıdır. Yığıtın en temel metotları eleman eklemeye yarayan Push ve eleman çekmeye yarayan Pop metodu ve yığıtın boş mu dolu mu olduğunu gösteren metotlardır. Bir yığıt sınıfı oluşturmak için aşağıdaki kodu kullanabilirsiniz.



**Uygulama 1-**

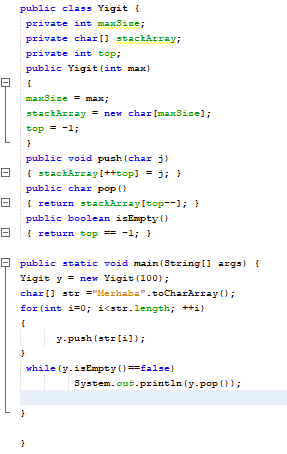
Yığıt ile bir metin içerindeki parantezlerin kontrolü yapılabilir. Aşağıdaki kod kullanılarak bir String’teki iç ve dış parantezlerin eşit sayıda olup olmadığı kontrol edilebilir. İç içe parantezler içeren bir ifadede parantezlerin geçerli olması için açılan ve kapanan toplam parantez sayısı eşit olmalıdır. Aç “(“ ve kapa “)” parantezlerin eşitliğine bakılır. Kapanan her parantezden önce bir parantez açılmış olmalıdır. Her “)” için bir “(“ olup olmadığına bakılır. Karşılaşılan parantezleri tutmak üzere yığıt kullanılabilir. Bir parantezle karşılaşıldığında yığıta eklenir. İlgili parantezlerin karşılığı ile karşılaşıldığında ise yığıta bakılır. Yığıt boş değilse yığıttan bir eleman çıkarılarak doğru karşılık olup olmadığı kontrol edilir. Doğruysa işlem sürdürülür. Değilse ifade geçersizdir. Yığıt sonuna ulaşıldığında yığıt boş olmalıdır. Aksi halde açılmış ama kapanmamış parantez olabilir. Parantezleri geçerli bir ifadede şu şartlar olmalıdır. İfadenin sonunda parantez sayısı 0 olmalıdır. İfadede ya hiç parantez yoktur veya açılan parantezlerin sayısı ile kapanan parantezlerin sayısı eşittir. İfadenin hiçbir noktasında parantez sayısı negatif olmamalıdır. Bu, parantez açılmadan bir parantezin kapanmadığını gösterir.

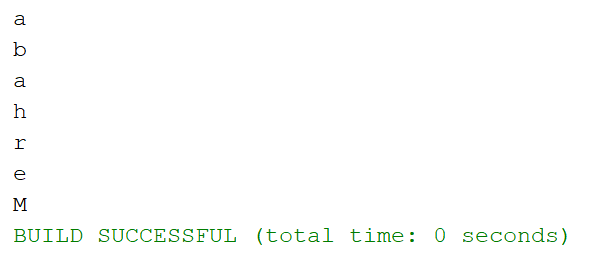
Aşağıda parantez kontrolü yapılan kod ve çıktısı verilmektedir.



Uygulama 2- Yığıt Kullanılarak String’i Tersten Yazma

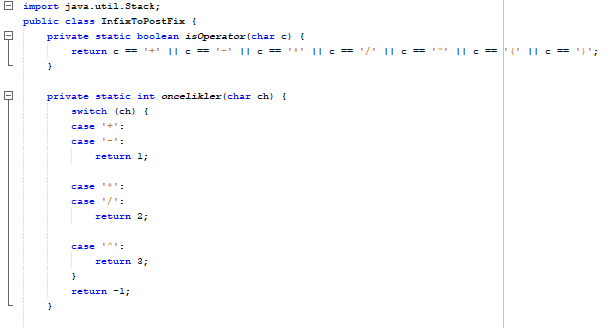
Yığıt veri yapısını kullanarak verilen bir string ifadeyi ters çevirecek algoritma aşağıda verilmektedir. String ifadedeki her bir karakter yığıta Push metodu ile aktarılır. Yığıt son eklenen eleman ilk çekileceği için yığıta eklenen her bir karakterin Pop metodu ile stack’den okunması ile ifade ters çevrilebilmektedir.

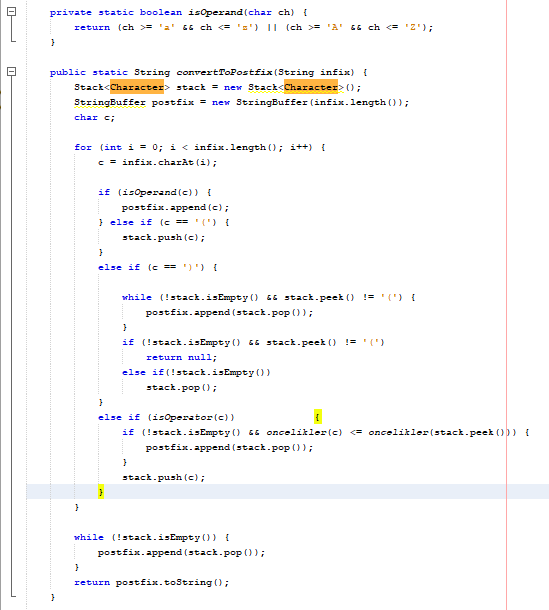


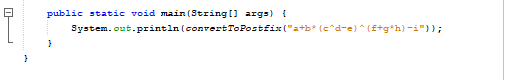


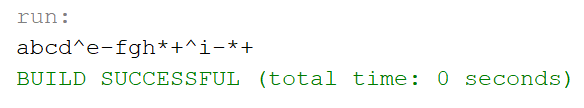
**Uygulama 3- Infix-Postfix dönüşümü**

Polish notasyonu Bilgisayar Bilimleri alanındaki önemli konulardan bir tanesidir. Operatörleri, operandlardan önce veya sonra gösterme metodu olarak tanımlanabilir. Infix, bildiğimiz klasik gösterimdir. Postfix, operatörlerin operandlardan sonra yazıldığı gösterimdir. Postfix formda parantez kullanımına gerek yoktur. Infix formdan Postfix forma çevrilen bir ifadede operandların bağlı olduğu operator’leri (+,-,\*,/) görmek zorlaşır (3 4 5 \* + ifadesinin sonucunun 23’e, 3 4 + 5 \* ifadesinin sonucunun 35’e karşılık geldiğini bulmak Infix gösterime alışık olduğumuz için zor gibi görünür). Fakat parantez kullanmadan tek anlama gelen bir hale dönüşür. İşlemleri, hesaplamaları yapmak kolaylaşır. Birçok derleyici 3\*2+5\*6 gibi bir Infix ifadenin değerini hesaplayacağı zaman Postfix forma dönüştürdükten (belirsizliği ortadan kaldırdıktan sonra) sonucu hesaplar : “3 2 \* 5 6 \* +” - Hem Infix-Postfix dönüşümünde hem de Postfix hesaplamasında yığıt kullanılabilir. Aşağıdaki kod ile infix bir ifadenin postfix gösterimine dönüşmesini sağlamaktadır.









**Uygulama 4- Labirentte gezinme**

Bir N \* M boyutunda labirent matrisi olarak verilmektedir ve başlangıç(0, 0) noktasında bir fare olduğu düşünülmektedir. Fare, labirentte belirli bir blokta (fx, fy) bulunan yiyecekleri yemek ister. Bir labirent matrisinde 0, yol olmadığı anlamına gelir ve 1, hedefe giden yolda o noktanın kullanılabileceği anlamına gelir. Fare, bloğun çıkmaz bir uç olmaması koşuluyla herhangi bir bloğa herhangi bir yönde (çapraz olarak değil) hareket edebilir. Görev, farenin yiyeceğe ulaşıp ulaşamayacağı için herhangi bir yol olup olmadığını kontrol etmektir. Yolun yazdırılmasına gerek yoktur.

