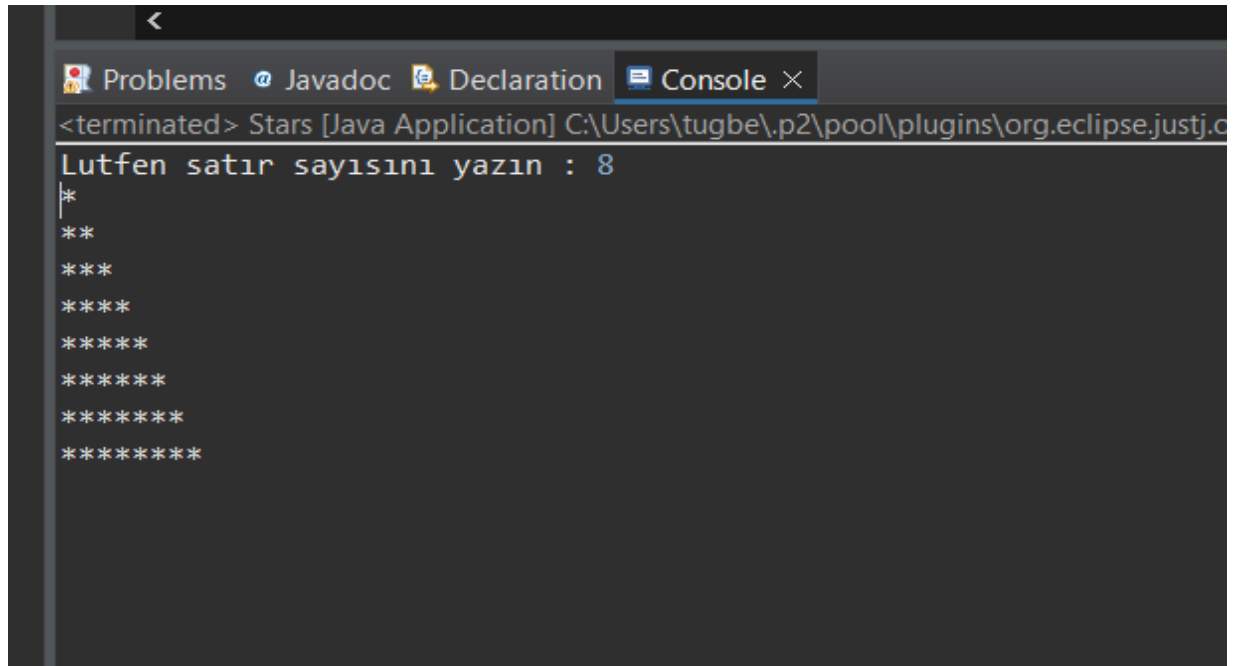
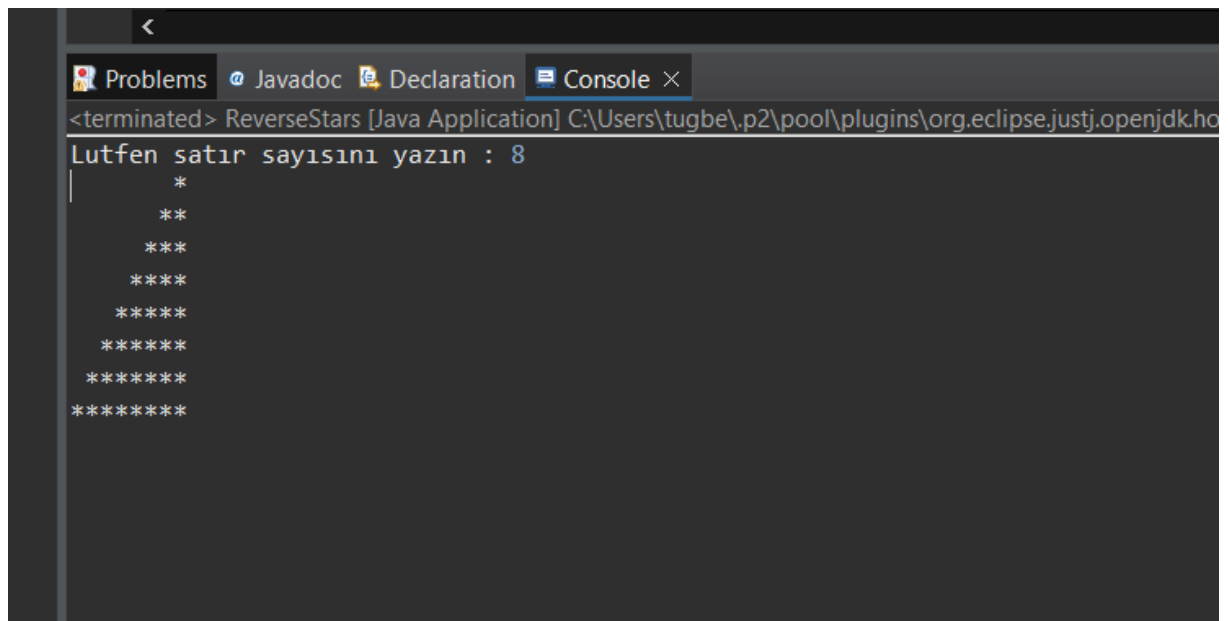


2-



```
<terminated> Stars [Java Application] C:\Users\tugbe\.p2\pool\plugins\org.eclipse.justj.c
Lutfen satır sayısını yazın : 8
*
**
***
****
*****
*****
*****
*****
*****
```



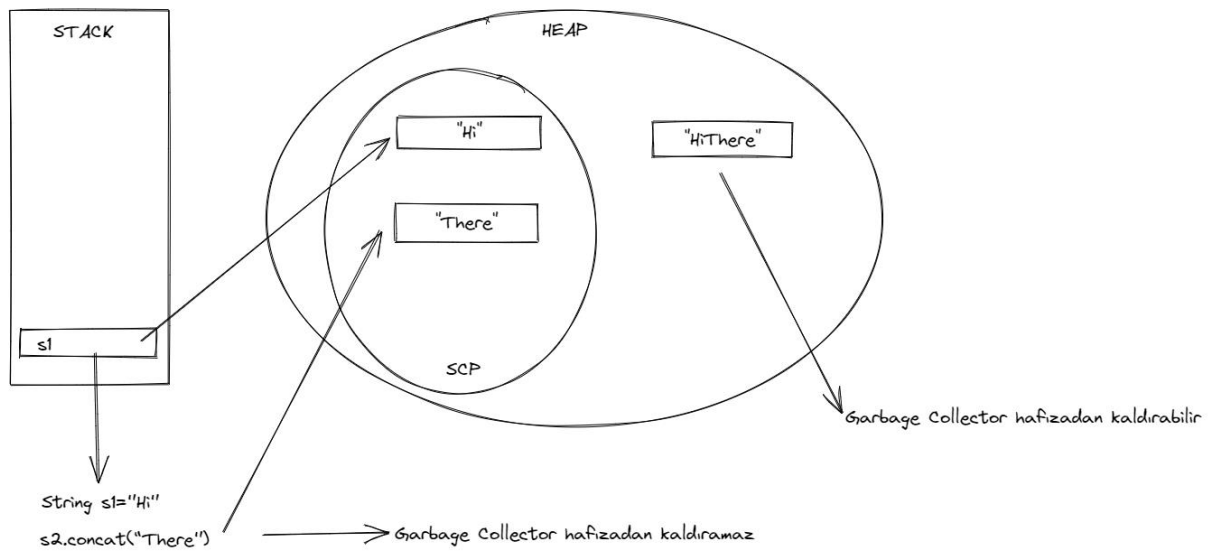
```
<terminated> ReverseStars [Java Application] C:\Users\tugbe\.p2\pool\plugins\org.eclipse.justj.openjdk.ho
Lutfen satır sayısını yazın : 8
      *
     **
    ***
   ****
  *****
 *****
*****
*****
```

3- Java'nın platform bağımsızlığı bir ara platform üzerinden sağlanmaktadır bu platforma JVM adını vermekteyiz. Java üzerinde yazdığımız kaynak kodlar **javac** derleyicisi tarafından derlenir ve byte kodu haline getirilir. Sonrasında hangi işletim sistemini kullanıyorsak o işletim sistemine ait JVM devreye girer ve byte kodunu satır satır okur ve byte kod talimatlarını makineye özgü anlaşılabilir dile dönüştürür. Bu nedenle, byte kod platformdan bağımsızdır ancak yorumlanan kod makineye özgüdür ve JVM'nin kurulu olduğu ortamda yürütülür. Bu yüzden Java programlar platformdan bağımsızdır.

4- Stack Space RAM'de primitive tiplerimizi tutan alandır, **LIFO**(List In First Out) mantığı ile çalışır. Stack Space'i bir koliye benzetebiliriz, örneğin koliyi doldurmak istediğimiz zaman kolinin en alt kısmında yukarı doğru doldururuz ve koliyi boşaltmak istediğimiz zaman en son eklediğimiz eşyaları çıkarmak ile başlarız. Stack kavramını tam olarak bu şekilde çalışmaktadır, metod içinde oluşturduğumuz değerleri sıralı bir şekilde hafıza ekleme yapar devamında işi bitenleri sondan başlamak üzere hafızadan çıkarır.

Heap Space ise objeleri ve class'ları tutmaktadır. Ek olarak Garbage Collector dediğimiz kavram bu kısımda kullanılır. Heap üzerinde oluşturulan classlar random şekilde yerleştirilir, örnek olarak Kişi adında bir class oluşturmuş olalım bu Heap memory'de tutulur. Eğer yeni bir Kişi class'ı oluşturmak istersek yine Heap kısmında oluşturmuş oluruz fakat aynı özellikli ve aynı isimli bir class daha elimizde bulunduğu için ilk oluşturduğumuz Kişi class'ı Garbage Collector tarafından hafızadan çıkarılır.

5- Java'da Immutable yani değişmez yapılar bulunmaktadır. String, Boolean, Integer gibi bütün wrapper classlar immutable'dır. Immutable classlar bir kere oluşturulur ve değişmezler.

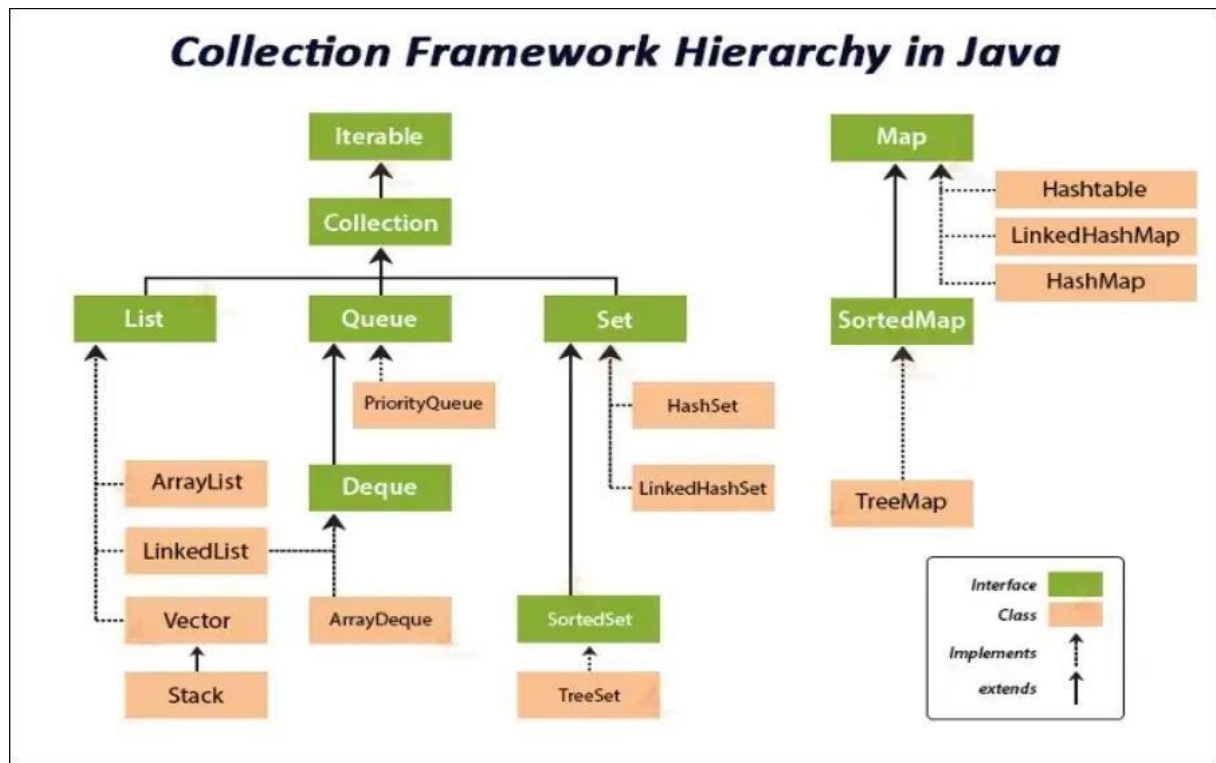


6- Karmaşayı atlatmak ve önlemek için java birden fazla kalıtım almayı desteklemez. Bunu örnek bir senaryo üzerinden açıklamak gerekirse elimizde **A**, **B**, **C** adında üç sınıfımız olsun. **C** sınıfı **A** ve **B** sınıflarını miras alır, **A** ve **B** sınıfları aynı metotlara sahipse ve siz onu sınıf nesnesinden çağırırsanız, **A** ve **B** sınıfından hangi metodu çağırıldığını ile ilgili belirsizlik yaşamış olursunuz bunun sonucunda compiler time error alırsınız. Kısacası aynı metotlara veya farklı methodlara sahip olsanız da compiler time error alırsınız.

7- Build tool bir yazılım projesini oluşturmak, compile etmektikten sonra ihtiyaç duyulan librarylerle birleştirmek için ihtiyaç duyulan toollardır. Java için **Gradle**, **Maven**, **Ant** gibi toolları örnek verebiliriz. Bir yazılım compile edildiğinde, ihtiyaç duyulan libraryler de compile edilmiş olan kodla beraber birleştirilir, build tool'un görevi de bu birleştirme işlemidir.

- **Apache Maven**, Java geliştirme ortamında kullanılan bir build tooludur. Maven, proje yapılandırması için bir XML kullanır.
- **Gradle**, proje oluşturma otomasyonu için yazılım geliştirmede kullanılan modern bir build tooludur. Java ve Android tarafında kullanılır.
- **Ant**, geçmişte yaygın olarak kullanılan bir derleme aracı olan Maven'in yerini alacak şekilde geliştirilmiş bir build tooludur. Bir XML dosyası kullanan Ant, derleme görevlerini otomatikleştirmek için kullanılır.

8-



Java'daki **Collection Framework**, nesne grubunu depolamak ve işlemek için bir mimari sağlayan bir frameworkdür. Collection, Interface olup içinde benzer türden nesneleri belirli şekilde tutacak, nesnelere ait temel davranışları belirler.

Collection Interface'e ait bazı özellikler;

- `int size()` : Collection içinde bulunan elemanların sayısını verir.
- `boolean isEmpty()` : Collection içerisinde eleman yoksa **true** değerini return eder.
- `boolean contains()` : Parametre olarak verilen eleman Collection içerisinde mevcutsa **true** değerini döndürür.
- `boolean add()` : Collection yapısına object ekler . Collection'ı implemente eden sınıflar bu metodu implemente etmek zorunda değildir.
- `boolean remove()` : Collection yapısından object remove eder . Collection ı implemente eden sınıflar bu metodu implemente etmek zorunda değildir.

Collection Interface'inin altında ise 3 farklı **Interface** vardır bunlar **Set**, **List** ve **Queue**'dir.

- **Set:** Set arayüzü Collection arayüzünü **extend** etmiştir. Yani **Collection Interface** metodlarını kalıtım alır. Set Interface'ini implemente eden tüm sınıfların ortak özelliği şudur; **duplicate** objeler set içerisine atılmaz. Aynı objeyi set içerisine atmak istediğimizde set bunu **ignore** eder.

HashSet: HashSet elemanlarını bir hash tablosunda tutar. Hash tablosu elemanlarına erişimi kolaylaştırmak için elemanlarına key vasıtasıyla ulaşılmasını sağlar.

LinkedHashSet: HashSet sınıfı ile aynı işi görmektedir. Bu sınıf elemanları koleksiyona eklenme sırasına göre tutmaktadır.

TreeSet: TreeSet, **Set**'ten türeyen **SortedSet** arabirimini uygulamaktadır. **SortedSet** arabirimi koleksiyona ait elemanları sıralı olacak şekilde tutmasını sağlar.

- **List:** List sayesinde koleksiyonumuzda tekrarlanan ya da tekrarlanmayan elemanları sıralı olarak tutabiliriz. Aynı zamanda elemanlara **index** yoluyla erişebiliriz.

ArrayList: Bu sınıf elemanlarını boyutu değişen dinamik dizilerde tutmaktadır. Yani bu sınıfa ait dizilerin boyutları azaltılabilir ya da arttırılabilir. Bir başka ifade ile bir dizinin boyutunu değiştirmede ilk dizideki elemanlar yeni diziye kopyalar.

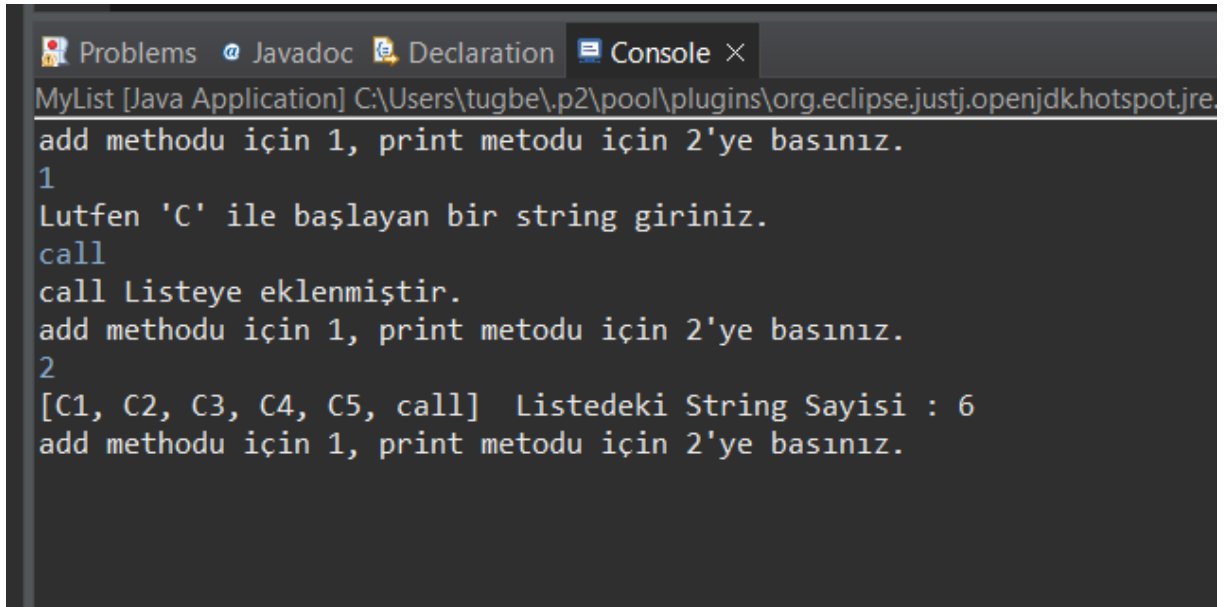
LinkedList: Bağlı listeler, bu sınıf C dilindeki bağlı listelerin javadaki görüntüsüdür. Bu tip listelerde elemanlar kendisinden sonra gelen elemanların bilgilerini ya da adreslerini tutarlar. Bu sınıf herhangi bir listeye eleman ekleme konusunda ArrayList sınıfından daha etkilidir.

Vector: Boyutları dinamik olarak değişebilen dizilerdir. ArrayList'e benzetilebilir.

Stack: Vector sınıfından türetilmiştir. Bu sınıfın özelliği en son hangi eleman kaydedilmişse sorguda ilk onu dönmesidir. (LIFO — last in first out). Kendisine ait bazı metotları;

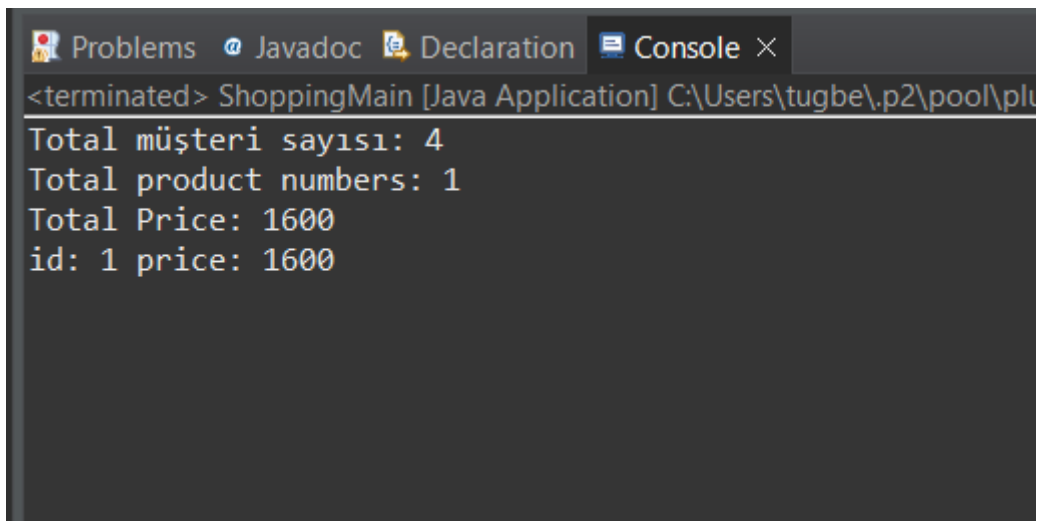
- **pop():** En son elemanı döner ve listeden çıkartır.
 - **push():** Verilen elemanı stack'in sonuna ekler.
 - **empty():** Stack boş ise **true**, değilse **false** döner.
 - **search():** Verilen elemanın stack'teki yerini döner.
- **Queue:** Bu sınıf stack sınıfından farklı olarak özel durumlar dışında ilk giren ilk çıkar mantığı ile çalışmaktadır. **FIFO** (first in first out). Bu sınıfa ait metotlar;
 - **add():** Verilen elemanı kuyruğa ekler.
 - **offer():** Verilen elemanı kuyruğa ekler.
 - **poll():** Queue'nin başındaki elemanı kuyruktan çıkartır.
 - **peek():** Sıradaki elemana ulaşmak için kullanılır.

9-



```
Problems Javadoc Declaration Console ×
MyList [Java Application] C:\Users\tugbe\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre
add methodu için 1, print metodu için 2'ye basınız.
1
Lutfen 'C' ile başlayan bir string giriniz.
call
call Listeye eklenmiştir.
add methodu için 1, print metodu için 2'ye basınız.
2
[C1, C2, C3, C4, C5, call] Listedeki String Sayisi : 6
add methodu için 1, print metodu için 2'ye basınız.
```

10-



```
Problems Javadoc Declaration Console ×
<terminated> ShoppingMain [Java Application] C:\Users\tugbe\.p2\pool\plu
Total müşteri sayısı: 4
Total product numbers: 1
Total Price: 1600
id: 1 price: 1600
```