Hafta 5: Ek Kaynak

Heap - Stack(Yığın)

Heap ve stack, programlama dillerinin çalışma zamanlarında kullanılan iki farklı bellek alanıdır.
https://www.javatpoint.com/stack-vs-heap-java

Stack ; metot yürütme sırasını ve yerel değişkenleri depolamak için kullanılır.

*Primitif tip dediğimiz **int, short, byte, long, decimal, double, float** gibi tipler value type (değer tipi) olarak adlandırılır ve stack de tutulur.*

- Stack, programda o anda çalışan, aktif olan metotlar için ayrılan bellek alanların olduğu yerdir. Bu sebeple **call stack** de denir.

- Her **metodun stackteki bellek alanına pencere (frame)** denir ve pencereler stackte üst üste açılırlar.
- Stack (yığın), "**last-in, first-out (LIFO)**" çalışır.

• Stackteki bellek pencereleri, kendisi için açıldığı metodun parametreleri, yerel değişkenleri vs. için bellek ihtiyaçlarını karşılar.
• Stack büyüklüğü, metodunun parametreleri ve o metotta tanımlanan değişkenler vb. bellek tüketen yapıların sayısı ve büyüklüğüne bağlıdır.
• Stack silinince, içindeki tüm veriler de silinir.

Heap; JVM tarafından nesne oluşturmak için kullanılan bellek alanıdır. Stackten çok daha büyüktür

- Nesneleri, JRE sınıflarını depolar. dinamik bellek ayırma ve bırakmayı yapar/kullanır.

StackDemo.java

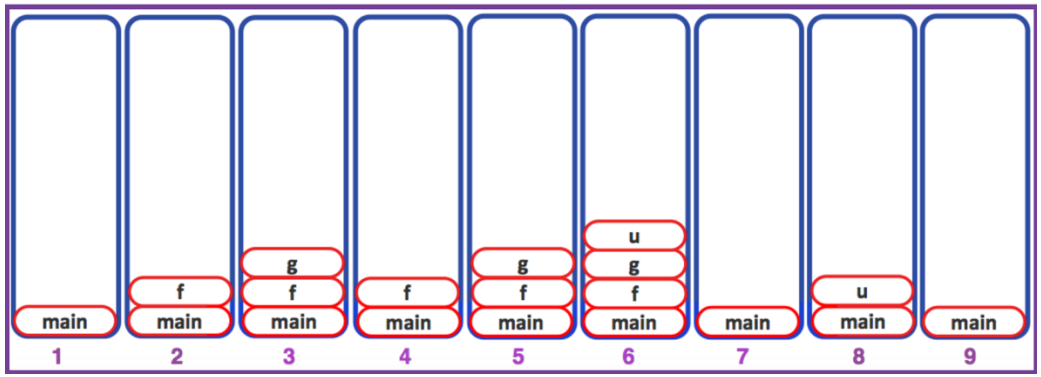
```java
//created by ikok-27.03.2022
public class StackDemo {
    public static void main(String[] args) {
        System.out.printf("1: Main çalıstı\n");
        f();
        u( deger: 8);
    }
     public static void f(){
         System.out.printf("2: F çalıstı\n");
         g( b: false);
         System.out.printf("4: g'den F'e dönüldü\n");
         g( b: true);
     }
    public static void g(boolean b){
        if(b==true){
            System.out.printf("5: g()-> b=true - u tetikle\n");
            u( deger: 6);
            System.out.printf("7: u() sonrası g-f kapanış- \n");
        }
        else {
            System.out.printf("3: g()-> b=False - f()'ye geri dön\n");}
    }
    public static void u(int deger){
        System.out.printf("%d: U(%d) çalıstı\n",deger,deger);}
}
```

StackDemo

```
1: Main çalıstı
2: F çalıstı
3: g()-> b=False - f()'ye geri dön
4: g'den F'e dönüldü
5: g()-> b=true - u tetikle
6: U(6) çalıstı
7: u() sonrası g-f kapanış-
8: U(8) çalıstı
```

StackDemo-Frame/pencere değişimine DİKKAT

Stack Demonun çalışmasıyla Stack'in değişimi



Scope-Kapsam

Değişkenlerin erişilebildiği, görülebildiği alana, kapsam (scope) denir.
• Java'da kapsam, bloklar tarafından belirlenir.

Değişkenlerin kapsamları, tanıtıldıkları yerden, içinde bulundukları bloğun sonuna kadardır.

- Üye değişkenlerin (member variables) kapsamı, tanıtıldıkları yerden sınıfın sonuna kadardır yani tüm sınıftır.

- Yerel değişkenlerin (local variables) kapsamı, tanıtıldıkları yerden, içinde bulundukları bloğun sonuna kadardır.

- Kapsama bloğun içindeki tüm alt kapsamlar da dahildir.
- Aynı kapsam içinde birden fazla aynı isimde, üye ya da yerel değişken olamaz.

• Fakat aynı kapsamda aynı isimde bir üye ve bir yerel değişken olabilir. • Bu durumda yerel değişken, üye değişkeni **gölgeler** (**shadowing**).

• Java bu durumdaki bir değişkene erişimde, en yakında tanıtılana yani yerel değişkene erişir, üye değişkene ulaşmak için ayrı bir yapı kullanmak gerekir.

• Bir değişkene kapsamının dışında erişilemez.
    - Derleme hatası alınır.

Bu durumda o değişkenin erişilemediği, görülemediği, kapsam dışı (out of scope) olduğu söylenir.

Çevrim(Conversion)-Casting

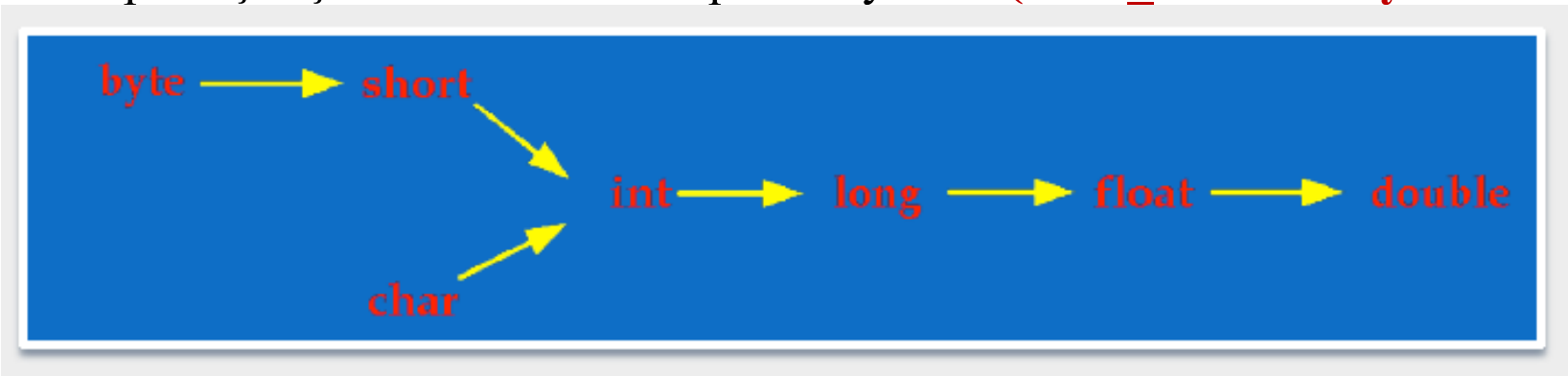
Bir değeri, sahip olduğu tipten farklı tipteki bir değişkene atamaya **çevrim** (**conversion**) denir.

1- Imkansız çevrim Boolean b=12;
(java: incompatible types: int cannot be converted to boolean)

2- Genişleten çevrim

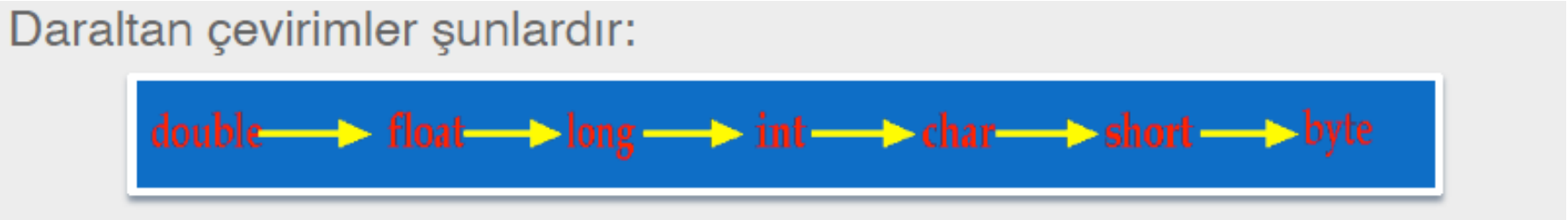
```
int uzun =20;
double d1=uzun;   // genişleten çevrim
```

- Java'da genişleten çevirimler otomatik olarak yapılır, çevirimin olması için atama yapmak dışında başka bir şeye gerek yoktur.
- Genişleten çevirimler hiçbir zaman çalışma zamanı hatası vermezler.
- İlkel tipler için çevrim-izin verilen promosyonlar **(NYP_Hafta5-slayt no:35-Fig.6.4)**



3- Daraltan çevrim

Bit açısından daha geniş olan bir tipten daha dar olan bir tipe yapılan çevirimlerdir.



Java'da daraltan çevirimler otomatik olarak yapılmaz, derleyici hata verir. Daraltan çevirim yapabilmek için çevirme işlemcisi (cast operator) olan "()" kullanılır:

- Atama yapılırken, çevirilen tip, çevirme işlemcisi içine yazılır.

```
double d2=12.45;
int i2= (int) d2; //daraltan çevrim-casting
```

Çevirme işlemcisi ile çalışma zamanında hata oluşmaz.

- Fakat çevirme sonucunda bir veri kaybı olabililir.

Çevirim sonucunda, çevirilen verinin büyüklüğüne bağlı olarak, geniş tipteki bitlerin hepsi yeni tipe aktarılamayabilir.

"final" anahtar kelimesi

- final, bir niteleyicidir (modifier) ve nitelediği şeyin bir anlamda değişmemesi gerektiğini ifade eder.
- final, değişken tanıtılırken niteleyici olarak kullanılır.
- **final**, değişkenlerle kullanıldığında onları **sabit**, **değişmez** (**constant**) yapar.
- Java'da sabit değerli değişkenler için **final** kullanıldığından **const** anahtar kelimesinin kullanımı yoktur.
- Java Language Specification "sabit değişken" (constant variable) tamlamasını kullanmaktadır.
- Eğer değişkenin değeri çalışma zamanında belirlenecekse, değişken boş sabite (blank final) olarak tanıtılabilir.

```
final int i = 5;
i = 8;  // Compiler error.
```

```
final int i;
i = 8;   // Not a compiler error.
i = -3;  // Compiler error.
```

```
final boolean b = !true;
final int i = 1 * 2 * 3 * 4 * 5 * 6;
final float f = (float) (7 * Math.PI) / 2;
final double d = 2.0 * Math.PI;
final String s1 = "Java is good!";
final String s2 = "The integer is" + 5;
```

Enumarations

Numaralandırmalar, bir programlama dilinde bir grup adlandırılmış sabiti temsil etme amacına hizmet eder.

- **Java'da Enum türleri önceden tanımlanmış sabit değerleri ifade etmek için kullanılır.**
- **İçerisindeki değerler sabit olduğu için tamamen büyük harf kullanıldı.** Küçük harflerle enum sabitleri tanımlamak mümkündür ancak kod okunabilirliği açısından Büyük olarak tanımlanması daha uygundur.
- Java'da enum bildirimi: Enum bildirimi bir Sınıfın dışında veya bir Sınıfın içinde yapılabilir, ancak bir metodun içinde yapılamaz.

```
public enum Gun {
    PAZARTESI,
    SALI,
    CARSAMBA,
    PERSEMBE,
    CUMA,
    CUMARTESI,
    PAZAR
}
Gun gun1 = Gun.PERSEMBE;    // doğru kullanım
Gun gun2 = Gun.PAZAR;       // doğru kullanım
```

- Enumlar if/else ve switch içerisinde kullanılabilirler, == operatörü ile karşılaştırılabilirler
- Enum sabitleri otomatik olarak static ve final tanımlanırlar, yaratıldıktan sonra değiştirilemezler
- Enum türleri bir sınıfın içinde veya dışında yaratılabilirler.
- Derleyici tarafından eklenen `values()` metodunu kullanarak bir Enum türü içerisindeki bütün sabitleri alabiliriz.