[JAVA NOTLARI ( NİSAN 2021 ) 33](#_Toc72245508)

[KISAYOLLAR 33](#_Toc72245509)

[Shift + cmd + 7 🡺 // 33](#_Toc72245510)

[Alt + 6 🡺 | 33](#_Toc72245511)

[Alt + ç 🡺 & 33](#_Toc72245512)

[1.DERS ( VARIABLE ) 33](#_Toc72245513)

[System.out.println("Hello World!"); 33](#_Toc72245514)

[char harf ='S'; //charlara değer ataması yaparken mutlaka tek tırnak kullan! 34](#_Toc72245515)

[**int** sayi =123; 34](#_Toc72245516)

[boolean isOld= true; // boolean'lar sadece true veya false değerlerini kullanabilirsiniz. 34](#_Toc72245517)

[short nazilliNufus=28000; 34](#_Toc72245518)

[long hucreSayisi=353445435; 34](#_Toc72245519)

[**double** pi = 3.14252352352; 34](#_Toc72245520)

[System.out.println(pi); // java bütün ondalık kesirleri otomatik olarak double kabul eder. rakamın soluna f konulmazsa virgüllü sayı olduğu için bunu double yap diye uyarı verir. 34](#_Toc72245521)

[**float** para = 5.25f; 35](#_Toc72245522)

[**char** harf1= 'a'; // a nın değeri 97 36](#_Toc72245523)

[**char** harf2='b'; // b nin değeri 98 36](#_Toc72245524)

[System.out.println(harf1 + harf2);// + koyduğumuzda harflerin değerini toplayıp sonucu verir. 195 çıkar. 36](#_Toc72245525)

[int aCharDeger='a'; 36](#_Toc72245526)

[System.out.println(aCharDeger); 36](#_Toc72245527)

[﻿Keywords in a Class 37](#_Toc72245528)

[public class MyFirstClass { } 🡪 public bir Access Modifier’dır. 37](#_Toc72245529)

[Access Modifier bir Class’a kimin ulaşıp kimin ulaşamayacağını belirler 37](#_Toc72245530)

[﻿MyFirstClass Class ismidir. 37](#_Toc72245531)

[“{” ile başlayıp “}” ile biten kısım Class Body olarak isimlendirilir. 37](#_Toc72245532)

[Class isimleri her zaman büyük harfle baslar 37](#_Toc72245533)

[﻿﻿Bir Metod nasıl oluşturulur (Create)? 37](#_Toc72245534)

[﻿public int myFirstMethod ( ) { } 37](#_Toc72245535)

[int return type’dır. 37](#_Toc72245536)

[return type metodun ürettiği sonucun tipidir. 37](#_Toc72245537)

[myFirstMethod( ) metod ismidir. 37](#_Toc72245538)

[Metod isimleri küçük harfle başlar. 37](#_Toc72245539)

[Birden fazla kelime içeriyorsa diğer kelimelerin ilk harfleri büyük olur. 37](#_Toc72245540)

[public bir Access Modifier’dır. 37](#_Toc72245541)

[Access Modifier bir metod’a kimin ulaşıp kimin ulaşamayacağını belirler 37](#_Toc72245542)

[“{” ile başlayıp “}” ile biten kısım Metod Body olarak isimlendirilir. 37](#_Toc72245543)

[﻿Main Method 🡪 public static void main(String[ ] args) { } 37](#_Toc72245544)

[﻿Variable Oluşturma (Declaration) : 37](#_Toc72245545)

[Data Tipi (Type) VariableAdı (Name); 37](#_Toc72245546)

[int yaş ; 37](#_Toc72245547)

[char ilkHarf ; 37](#_Toc72245548)

[﻿Variable’a Değer Atama (Assignment) : 37](#_Toc72245549)

[Data Type VariableName = Value; 37](#_Toc72245550)

[int age = 27; 37](#_Toc72245551)

[char initial =’A’;﻿ 37](#_Toc72245552)

[Data Types in Java 38](#_Toc72245553)

[Javada temel olarak iki tip data vardır: 38](#_Toc72245554)

[1. Primitive Data Types: Primitive data type => boolean, char, byte, short, int, long, float ve double. 38](#_Toc72245555)

[2. Non-Primitive Data Types: Non-Primitive data type => String. 38](#_Toc72245556)

[“Primitive” ve “Non-Primitive” Arasındaki Farklar 38](#_Toc72245557)

[1) Primitive’ler sadece value içerir, non-primitive’ler value and methodlar içerir. 38](#_Toc72245558)

[2) Primitive’ler küçük harf ile, non-primitive’ler büyük harf ile başlar. 38](#_Toc72245559)

[3) Primitive’leri Java oluşturdu biz primitive oluşturamayız. 38](#_Toc72245560)

[Non-primitive’leri biz de oluşturabiliriz, Java da oluşturabilir.Mesela String’i Java oluşturmuştur. 38](#_Toc72245561)

[4) Primitive’lerin büyüklükleri data type’ing gore değişir, non-primitive’lerin hepsi aynı büyüklüktedir. 38](#_Toc72245562)

[2. DERS ( WRAPPERCLASSES) 38](#_Toc72245563)

[int data tipinin max ve min değerlerini bulup ekrana yazdırınız. 38](#_Toc72245564)

[System.out.println(Integer.MAX\_VALUE); 38](#_Toc72245565)

[bir Class'ın içindeki metodlara ulaşmak için class isminden sonra . koyuyoruz, çıkan listeden seçim yapıyoruz. 39](#_Toc72245566)

[﻿Kullanıcıdan Her Tür Data Almak 40](#_Toc72245567)

[﻿1. Scanner scan = new Scanner(System.in); 40](#_Toc72245568)

[2. System.out.println(“30 dan küçük bir sayı giriniz”); 40](#_Toc72245569)

[3. int num1 = scan.nextInt(); 40](#_Toc72245570)

[4. System.out.println(num1); 40](#_Toc72245571)

[3. DERS ( OPERATORS ) 49](#_Toc72245572)

[String isim=scan.nextLine(); 49](#_Toc72245573)

[Stringleri kullanıcıdan almak için ya sadece next kullanırsınız yada nextLine() kullanırız. next tek kelimelik Stringler için kullanılır. 49](#_Toc72245574)

[nextLine satırın tamamını almak, tüm cümleyi almak için kullanılır. 49](#_Toc72245575)

[byte by=101; 53](#_Toc72245576)

[int sayi=by; 53](#_Toc72245577)

[Küçük data tipini büyük data tipine çevirmek için extra bir kod yazmaya gerek yok. java bunu otomatik olarak yaparak buna Auto widining denir. 53](#_Toc72245578)

[Büyük data tipini küçük data tipine çevirmek, java tarafından yapılmaz, örnekteki gibi sağ tarafa parantez içinde dönüştürmek istediğimiz data tipini yazmalıyız. 53](#_Toc72245579)

[double sayi3=23.9; 53](#_Toc72245580)

[int by3=(int)sayi3; 53](#_Toc72245581)

[System.out.println(by3); 53](#_Toc72245582)

[// sonuc: 23 53](#_Toc72245583)

[data tipi int olduğu için yuvarlama yapmadan sadece tam sayı kısmı çıkar. 54](#_Toc72245584)

[4. DERS ( OPERATORS ) 57](#_Toc72245585)

[Javada nextInt kullandıktan sonra nextLine kullanırsak kod çalışmıyor. 57](#_Toc72245586)

[çözüm: nextInt yerine nextLine kullan sonra elde edilen String'i Integer.parsentInt kullanarak integer'a çeviririz. 57](#_Toc72245587)

[Scanner scan=new Scanner(System.in); 57](#_Toc72245588)

[System.out.println("Yaşınınız giriniz"); 57](#_Toc72245589)

[String yas=scan.nextLine(); 57](#_Toc72245590)

[int yeniYas=Integer.parseInt(yas); 57](#_Toc72245591)

[12835 sayisinin 23 e bölümünden kalanı bulan program yazınız 59](#_Toc72245592)

[Bir sayının diğer bir sayıya bölümünden kalanı bulmaya modulus deniyor. İşlemi yapmak için % işareti kullanılır 59](#_Toc72245593)

[int a=12835; 59](#_Toc72245594)

[int b=23; 59](#_Toc72245595)

[System.out.println(a%b); 59](#_Toc72245596)

[Bir tam sayının birler basamağını bulmak için sayının 10 ile modülünü alırız. 60](#_Toc72245597)

[int sonRakam=sayi1%10; 60](#_Toc72245598)

[İlk rakamı bulmak için 4 basamaklı olduğu için 1000 e böleriz. 60](#_Toc72245599)

[int ilkRakam=sayi1/1000; 60](#_Toc72245600)

[int v2=sayi1/100; 64](#_Toc72245601)

[int yuzler=v2%10; 64](#_Toc72245602)

[5. DERS ( OPERATORS ) 67](#_Toc72245603)

[sayi1++; 69](#_Toc72245604)

[Variable isminden sonra iki kere artı sembolu yazarsanız bu variablenin değerini bir artırmış olursunuz. 69](#_Toc72245605)

[6. DERS ( IF STATEMENT ) 70](#_Toc72245606)

[Parantez içindeki koşul(condition) doğru ise body içerisindeki komutu çalıştırır. değilse bir sonraki satıra geçer 70](#_Toc72245607)

[if(3>2) { 70](#_Toc72245608)

[System.out.println("if body çalıştı"); 70](#_Toc72245609)

[} 70](#_Toc72245610)

[VEYA isleminden True sonucu alabilmek bir tanesinin True olmasi yeterlidir 72](#_Toc72245611)

[True||True = True <-> True||False = True <-> False||False = False 72](#_Toc72245612)

[True=1, False=0 ==> VEYA islemi toplamaya benzer 72](#_Toc72245613)

[&& VE isleminden True sonucu alabilmek icin tum condition'lar true olamalidir 73](#_Toc72245614)

[True && True = True <-> True && False = False <-> False && False = False 73](#_Toc72245615)

[True=1, False=0 ==> VE islemi carpmaya benzer 73](#_Toc72245616)

[& ( alt+ç) kısayol 73](#_Toc72245617)

[| (alt+6) kısayol 73](#_Toc72245618)

[System.out.println("Bir character girin"); 78](#_Toc72245619)

[char harf=scan.next().charAt(0); // Kullanıcıdan char almak için next().charAt(0); kullanılır. 78](#_Toc72245620)

[IF ELSE STATEMENT 79](#_Toc72245621)

[Stringlerin eşitliğini kontrol etmek istiyorsak asla!!! == sembolu kullanmıyoruz.Bunun yerine equals methodu kullanıyoruz. "==" Stringin değerini ve adresini kontrol eder. "equals()" ise sadece değerleri kontrol eder. 79](#_Toc72245622)

[IF ELSE IF 83](#_Toc72245623)

[if(num<200000) { 85](#_Toc72245624)

[System.out.println("Kazandınız"); 85](#_Toc72245625)

[}else if(num<500000) { 85](#_Toc72245626)

[System.out.println("Amorti"); 85](#_Toc72245627)

[}else 85](#_Toc72245628)

[System.out.println("Kaybettiniz"); 85](#_Toc72245629)

[scan.close(); 85](#_Toc72245630)

[} 85](#_Toc72245631)

[7. DERS ( NESTED IF STATEMENT) 86](#_Toc72245632)

[if(num>=0) { 86](#_Toc72245633)

[if(num>9) { 86](#_Toc72245634)

[System.out.println("Sayi"); 86](#_Toc72245635)

[}else if(num<=9 && num>=0) { 86](#_Toc72245636)

[System.out.println("Rakam"); 86](#_Toc72245637)

[scan.close(); 86](#_Toc72245638)

[} 86](#_Toc72245639)

[Scanner Class'ının kullandıktan sonra en altta main method'un içinde Scanner Class'ını kapatınız. Scanner Class'ını kapatmak için scan.close(); yazmak yeterlidir. 86](#_Toc72245640)

[8.DERS ( TERNARY OPERATOR ) 92](#_Toc72245641)

[﻿Condition ? Code 1 : Code 2 92](#_Toc72245642)

[True False 92](#_Toc72245643)

[String sonuc= num>=0 ? "Poziftif veya 0" : "Negatif"; 92](#_Toc72245644)

[&& işleminde hız kazanmak için ilk karşılaştırmaya bakar yanlış ise ikinci karşılaştırmaya bakmaya gerek duymadan direk false der. Bu yazılış yükü azaltır,hatayı azaltır. 98](#_Toc72245645)

[& işleminde ise bütün karşılaştırmalar kullanılır. 98](#_Toc72245646)

[& ile && aynı sonucu verir. 98](#_Toc72245647)

[9. DERS ( SWİTCH STATEMENT ) 99](#_Toc72245648)

[String ay=scan.next(); 103](#_Toc72245649)

[ay=ay.toLowerCase(); 103](#_Toc72245650)

[switch() de long, float, double ve boolean kullanılmaz. 105](#_Toc72245651)

[switch() int, byte, short, char,String kullanılır. 105](#_Toc72245652)

[switch() de case den sonra sadece bir değer yazabilirsiniz, herhangi bir işlem yapamazsınız. 105](#_Toc72245653)

[switch (not) { 105](#_Toc72245654)

[case 0: 105](#_Toc72245655)

[System.out.println("D"); 105](#_Toc72245656)

[break; 106](#_Toc72245657)

[case 50: 106](#_Toc72245658)

[System.out.println("C"); 106](#_Toc72245659)

[break; 106](#_Toc72245660)

[case 60: 106](#_Toc72245661)

[System.out.println("B"); 106](#_Toc72245662)

[break; 106](#_Toc72245663)

[case 80: 106](#_Toc72245664)

[System.out.println("A"); 106](#_Toc72245665)

[break; 106](#_Toc72245666)

[default: 106](#_Toc72245667)

[System.out.println("Geçerli not giriniz"); 106](#_Toc72245668)

[} 106](#_Toc72245669)

[10. DERS ( STRING METHODS ) 107](#_Toc72245670)

[indexOf() method'u String içindeki karakterin indexini belirlemek için kullanılır. 107](#_Toc72245671)

[String str1="Javva World"; 107](#_Toc72245672)

[System.out.println(str1.indexOf('v')); 107](#_Toc72245673)

[Ekrana 2 yazdırır.index sayımı 0 dan başlar ve 2 olarak yazar. 107](#_Toc72245674)

[System.out.println(str1.indexOf('a',2)); 107](#_Toc72245675)

[ikinci index ten itibaren 'a'yı arayıp indexini yazdır. 107](#_Toc72245676)

[System.out.println(str1.indexOf('a',5)); 108](#_Toc72245677)

[5. indexten itibaren a olmadığı için ekrana -1 yazdırır. 108](#_Toc72245678)

[lastIndexOf() methodu aradığınız karakterin String içindeki son görünümünün indexini return eder. 109](#_Toc72245679)

[System.out.println(str1.lastIndexOf('r')); 109](#_Toc72245680)

[// ekrana 9 yazdırır. 109](#_Toc72245681)

[System.out.println(str1.lastIndexOf('i',7)); 109](#_Toc72245682)

[ilk 7 indexin içinde i harfinin son görümünün indexini return eder. 109](#_Toc72245683)

[charAt() methodu belli indexteki characteri return eder. 110](#_Toc72245684)

[System.out.println(str1.charAt(4)); 110](#_Toc72245685)

[olmayan bir index ile işlem yapmaya çalışırsanız console da exeption alırsınız. 110](#_Toc72245686)

[length() methodu Stringin içinde kac karakter olduğunu verir. 110](#_Toc72245687)

[length() saymaya 1 den başlar. 110](#_Toc72245688)

[contains() methodu bir Stringin içinde bir characterin olup olmadığını kontrol eder. 112](#_Toc72245689)

[eğer o karakter Stringin içinde varsa true yoksa false return eder.contains methodunun içine her zaman String koyuyoruz. charı kabul etmez. 112](#_Toc72245690)

[System.out.println(str1.contains("kar")); 112](#_Toc72245691)

[trim metodu bir Stringin baş ve sonundaki boşlukları siler. 112](#_Toc72245692)

[String str2=" Ali Can "; 112](#_Toc72245693)

[System.out.println(str2.trim()); 112](#_Toc72245694)

[isEmpty() methodu bir Stringin içinde character olup olmadığına bakar.character yoksa true, varsa false return eder. 112](#_Toc72245695)

[Note: isEmpty() true çıkarsa length 0 çıkar. 113](#_Toc72245696)

[Substring(4) metodu 4.indexten itibaren sonraki kısmı keser siler. Substring(2,4) 2. ve 3. İndexi alır. 116](#_Toc72245697)

[toLowerCase() metodu küçük harfe çevirir 116](#_Toc72245698)

[toUpperCase() metodu büyük harfe çevirir 116](#_Toc72245699)

[11. DERS ( STRING METHODS ) 124](#_Toc72245700)

[startsWith() methodu bir Stringin istenen harfle başlayıp başlamadğını kontrol eder. istenen harfle başlıyorsa True, istenen harfle başlamıyorsa False return eder. 124](#_Toc72245701)

[System.out.println(str1.startsWith("A")); // true 124](#_Toc72245702)

[System.out.println(str1.startsWith("a",3)); 124](#_Toc72245703)

[3.index ve sonrasında a varmı diye bakar varsa true yoksa false return eder. 124](#_Toc72245704)

[endsWith( ) methodu bir Stringin istenen harfle bitip bitmediğini kontrol eder. istenen harfle bitiyorsa true, bitmiyorsa false return eder. 124](#_Toc72245705)

[System.out.println(str2.endsWith("n")); // true 124](#_Toc72245706)

[concat() method cancatenation yapmanın yani iki stringi birleştirmenin diğer yönetmidir.str1+str2 ile aynıdır 126](#_Toc72245707)

[concat'i istediğiniz kadar peşpeşe kullanabilirsiniz. 126](#_Toc72245708)

[System.out.println(str1.concat(str2).concat(str1)); 126](#_Toc72245709)

[replace() methodu bir karakterin yerine başka bir karakter yazmaya yarar. 126](#_Toc72245710)

[String str3="ata"; 126](#_Toc72245711)

[System.out.println(str3.replace("a", "ü")); // ütü yazar 126](#_Toc72245712)

[System.out.println(str3.replace("x", "y")); // x harfi olmadığı için hiç birşey yapmaz 126](#_Toc72245713)

[System.out.println(str3.replace("t", "")); // aa return eder.t yi imha ediyor. bir harfi silmek için kullanılabilir. 126](#_Toc72245714)

[replaceFirst() methodu değiştirmek istediğimiz karakterin sadece ilk eşleşen karakteri değiştirir. 126](#_Toc72245715)

[String str4="karakartal"; 127](#_Toc72245716)

[System.out.println(str4.replaceFirst("k", "K")); // Karakartal 127](#_Toc72245717)

[substring() methodu bir String'in belli bir bölümünü kesip almak için kullanılır 128](#_Toc72245718)

[String str1="Java calışana kolaydır"; 128](#_Toc72245719)

[System.out.println(str1.substring(22)); // metin 22 index ekranda hiç bir şey göremezsiniz 128](#_Toc72245720)

[System.out.println(str1.substring(23)); // 23 olunca hata verir. 128](#_Toc72245721)

[System.out.println(str1.substring(5, 13)); 128](#_Toc72245722)

[ilk index dahil ikinci index dahil değil. 128](#_Toc72245723)

[System.out.println(str1.substring(6, 6));// ilk "6" dahil et ikinci "6" dahil etme. ekranda birşey gözükmez. 128](#_Toc72245724)

[valueOf() metodu : Integer wrapper Classının içinde olan valueof() methodu String'i integer'a çevirir. 129](#_Toc72245725)

[String'in içindeki valueof() methodu integer'i String'e çevirir. 129](#_Toc72245726)

[ForLoop 135](#_Toc72245727)

[6,5,4,3,2,1,0,-1 yi ekrana yazdıran bir for loop oluşturunuz. 135](#_Toc72245728)

[for(int i=6;i>-2;i-- ) { 135](#_Toc72245729)

[System.out.println(i); 135](#_Toc72245730)

[} 135](#_Toc72245731)

[ilk 5 sayma sayısının toplamını veren programı for loop ile yazınız. 137](#_Toc72245732)

[int sum=0; 137](#_Toc72245733)

[for(int i=1;i<6;i++) { 137](#_Toc72245734)

[sum=sum+i; 137](#_Toc72245735)

[} 137](#_Toc72245736)

[System.out.println("Toplam:"+sum); 137](#_Toc72245737)

[System.out.println(); 137](#_Toc72245738)

[9! i hesaplayan for loop yazınız 139](#_Toc72245739)

[int product3=1; 139](#_Toc72245740)

[for(int i=1;i<10;i++) { 139](#_Toc72245741)

[product3=product3\*i; 139](#_Toc72245742)

[} 139](#_Toc72245743)

[System.out.println("9!="+product3); 139](#_Toc72245744)

[System.out.println(); 139](#_Toc72245745)

[} 139](#_Toc72245746)

[12. DERS ( SCOPE KURALLARI – INSTANCE VARIABLE – LOCAL VARIABLE ) 142](#_Toc72245747)

[1. KURAL 142](#_Toc72245748)

[Class'ın içinde metodların dışında oluşturulan variable'ları bütün metodlar istedikleri zaman istedikleri kadar kullanabilirler. Bu variable'lara "Class Variable" veya "Instance Variable" denir. Instance: görünür demektir. Class variable'lara değer ataması yapmazsak Java onlara default değerler verir. Java tüm sayılar için default olarak 0 değerini verir. 142](#_Toc72245749)

[2.KURAL 143](#_Toc72245750)

[Bir metodun içerisinde oluşturulan variable'lar sadece o metodun içinde kullanılabilir. O metodun dışındaki metodlar o variable'i kullanamazlar. Metodun içinde oluşturulan bu variable'lara "Local Variable" denir. Body içerisinde oluşturulan variable'lere değer atamak ZORUNDAYIZ. Java onlara default value vermez. 143](#_Toc72245751)

[3.KURAL 143](#_Toc72245752)

[Metodların parantezleri içerisinde oluşturulan variable'lar Local variable'lar gibi sadece o metodların bodyleri içinde kullanılabilir. Bu variable'lara "Local Variable" denir. Parantez içinde oluşturulan variable'lara ise değer ataması yapmayınız. 143](#_Toc72245753)

[4.KURAL 143](#_Toc72245754)

[Java yukarıdan aşağı çalışır. value'nin önce tipini tanımlamyıp, ardından değer ataması yapılmalıdır yoksa hata verir. 143](#_Toc72245755)

[WHILE LOOP 144](#_Toc72245756)

[While => iken demektir. 144](#_Toc72245757)

[() parantez içindeki koşula bakar true ise { } içindekini return eder döner tekrar while parantezine gider. 144](#_Toc72245758)

[num 4ten küçük iken body'yi çalıştır. 144](#_Toc72245759)

[int num=0; 144](#_Toc72245760)

[while(num<4) { 144](#_Toc72245761)

[System.out.println("Ali"); 144](#_Toc72245762)

[num++; // Bu satırı unutursanız while loop sonsuz döngüye girer.program kilitlenir. 144](#_Toc72245763)

[} 144](#_Toc72245764)

[13. DERS ( WHILE LOOPS ) 153](#_Toc72245765)

[INTERVİEWLERDE ÇOK SORULAN BİR SORU : Kullanıcıdan bir string alın ve bu stringi ekrana tersten yazdırın. 159](#_Toc72245766)

[char'i String'e cevirmek icin sonuna + "" ekleyebilirsiniz 160](#_Toc72245767)

[String num=scan.nextLine(); 160](#_Toc72245768)

[ters=ters+num.charAt(i)+""; 160](#_Toc72245769)

[14. DERS ( DO WHILE LOOP ) 161](#_Toc72245770)

[int num1=11; 161](#_Toc72245771)

[do { 161](#_Toc72245772)

[System.out.println(num1); 161](#_Toc72245773)

[num1+=2; 161](#_Toc72245774)

[}while(num1<20); 161](#_Toc72245775)

[do-while herhalükarda bir kere çalışır ama while koşulu taşımıyorsa çalışmaz. while de condition önce kontrol edilir sonra iş yapılır.do-while önce çalışır sonra kontrol edilir. 162](#_Toc72245776)

[while içindeki condition true oldukça loop devam eder.false olunca loop biter alta geçer. 163](#_Toc72245777)

[15. DERS ( METHOD CREATİON ) 172](#_Toc72245778)

[Genel usul olarak metodlar main method içine yazılmaz.main method dışında methodlar oluşturulur ve oluşturlan methodlar main method içine çağırılarak kullanılır. 172](#_Toc72245779)

[Bir metodu main içerisinden çağıracaksanız mutlaka static kelimesini kullanmalısınız. 172](#_Toc72245780)

[void bir return type dir.hiç bir şey üretmez sadece ekrana birşey yazar. 172](#_Toc72245781)

[Method parantezinin içinde oluşturulan variable'lara "parametre" denir.Methodu çağırırken method parantezinin içine yazılan değerlere "argüment" denir.Parametrenin data type ile argümentin data type'ı eşleşmeli. parametre int iken argüment String olamaz. 172](#_Toc72245782)

[public static int carpma(int num1,int num2) { 173](#_Toc72245783)

[return num1\*num2; 173](#_Toc72245784)

[} 173](#_Toc72245785)

[16. DERS ( CONSTRUCTOR, OVERLOADİNG ) 188](#_Toc72245786)

[CONSTRUCTOR 188](#_Toc72245787)

[Consructor Classtan object oluşturmaya yarar.Bir method değildir, variable değildir,bir kod parçasıdır.Constructor, Class'ın body'si içinde olmalıdır.Constructor'ın ismi Class'ın ismi ile aynı olmalıdır.Constructor'larda return type(int,String) olmaz.Consructori hangi Classta ürettiğimi unutursam cmd tuşana basıp constructor ismine tıklarsam üretildiği class'a götürüyor. 188](#_Toc72245788)

[Constructor01 hondaAraba=new Constructor01(); 188](#_Toc72245789)

[İsimden önce data type yazılır. int, String gibi. bu primitive data typlerdir, bunları java üretmiştir.yukarıdaki Constructor01 de bir data typedir. non primitive dir. biz üretmiş olduk.Bu şekilde istenildiği kadar data type üretilebilir. 188](#_Toc72245790)

[new objekct üretmeye yarar. ne zaman objeckt üretmek istersek new yazmamız gerekiyor. Bir yerda new görüyorsak orada objekt üretilmiş demektir. 189](#_Toc72245791)

[Constructor01() en sondaki bu kelime javaya bu kalıpla bir objekt üret demektir.hondaAraba yazıp nokta koyduğumuzda seçenekleri görürüz.bir insan Class oluşturuyorsa Objeckt üretmek içindır. 189](#_Toc72245792)

[Java Classla beraber bir constructor oluşturur.Buna "default constructor" denir. default Constructor'larda parametre olmaz.Scanner scan=new Scanner(System.in); içinde System.in şeklinde bir parametre olduğundan default Constructor değildir.Default Constructor Class oluşturulduğu anda Java tarafından oluşturulur. Biz kendimiz bir Constructor oluşturduğumuzda Java default Constructor'ı imha eder. 189](#_Toc72245793)

[System.out.println(hondaAraba.price); 189](#_Toc72245794)

[Her objeckt'e oluşturulduğu classtan isminin yanına koyulan bir nokta ile çağırıp içerdiği seçenekleri bulabiliriz, başka classta bunlara ulaşılmaz ama objekt ismi ile constructorı çağırabiliyoruz. 190](#_Toc72245795)

[Başka bir Class'tan ürettiğimiz Objeckt'i object ürettiğimiz class'da kullanabiliriz.Objeckt(in özelliklerini değiştirebiliriz, fakat bu değişim sadece o object ile sınırlı kalır. yeni üretilecek olan objektler bu değişimden etkilenmez.Eğer üretilecek her Objekt'in değişime uğramış olmasını istiyorsanız class'daki özelliklerini değiştirmelisiniz. 190](#_Toc72245796)

[Variableları bütün Classta kullanacağımız için main Class dışına yazıyoruz(instance variable).this bu Class'ın içindekiler demektir.Constructor'ı kendimiz tanımlayınca default Constructor iptal olur artık bizim tanımlamalarımızı kullan demektir. Constructor oluşturma main dışında, objeckt oluşturma main içinde yapılır. 191](#_Toc72245797)

[Constructor03(String isim, int yas,boolean emekli){ 191](#_Toc72245798)

[this.isim=isim; 191](#_Toc72245799)

[this.yas=yas; 191](#_Toc72245800)

[this.emekli=emekli; 191](#_Toc72245801)

[} 191](#_Toc72245802)

[public static void main(String[] args) { 191](#_Toc72245803)

[Constructor03 insan01=new Constructor03(); 191](#_Toc72245804)

[System.out.println(insan01.isim); 191](#_Toc72245805)

[OVERLOADİNG 193](#_Toc72245806)

[1. Method ismini değiştirmeden parametreleri değiştirerek aynı isimli methodlar oluşturabilirsiniz. 193](#_Toc72245807)

[2. Main method içerisine çağrılan method'larda parametrelere bakar hangi parametre hangi methoda uyuyorsa onu kullanır. 193](#_Toc72245808)

[3. Parametre sayısı değiştirerek aynı isimli ayrı method oluşturulabilir. 193](#_Toc72245809)

[4. Parametre sayısını değiştirmeden herhangi bir parametrenin type'ini değiştirerek isim değiştirmeden ayrı bir method oluşturabiliriz. 193](#_Toc72245810)

[5. öncelik olarak data typlerden birebir olarak ölçüşene bakar onu çalıştırır, birebir örtüşen yoksa Auto wideninge göre kullanabileceği methoda bakar yoksa hata verir. 193](#_Toc72245811)

[6. farklı data type'indeki parametrelerin yerlerini değiştirerek te aynı isimli farklı methodlar oluşturabilirsiniz. 193](#_Toc72245812)

[// Method 1 193](#_Toc72245813)

[public static void toplama (int num1,int num2) { 193](#_Toc72245814)

[System.out.println(num1+num2); 193](#_Toc72245815)

[}// Method 2 194](#_Toc72245816)

[public static void toplama (int num1) { 194](#_Toc72245817)

[System.out.println(num1+num1); 194](#_Toc72245818)

[Note 1: Return type'i değiştirmek overloading için yeterli değildir. Java methodların farklı olup olmadıklarını anlamak için iki şeye bakar.a) method ismine b) parametrelere bakar. 195](#_Toc72245819)

[Method ismi ve parametreler Java için methodların imzası gibidir.Java da method imzası yerine " method signature" denir. 195](#_Toc72245820)

[Note 2: Access Modifier'ları değiştirmek de methodları Java açısından farklı hale getirmez(public,privat vs. = access modifier)çünkü access modifier'lar method signature'a dahil değildir. 195](#_Toc72245821)

[Note 3: Method'ların body'sini değiştirmek de methodları java açısında farklı hale getirmez çünkü body'lar method signature'a dahil değildir. 195](#_Toc72245822)

[Note 4: Siz daha kodu yazarken yani kodu çalıştırmadan önce Java sizi yaptığınız hatalardan dolayı uyarırsa bu tip hatalara "Compile Time Error" denir. (altta kırmızı çizgi ile yazması) 195](#_Toc72245823)

[Note 5: kodu yazarken herhangi bir hata yok. fakat kodu çalıştırdıktan sonra console'da kırmızı hata mesajlar alırsanız bu hatalara "Run Time Error" denir. 195](#_Toc72245824)

[17. DERS ( CONSTRUCTOR ) 196](#_Toc72245825)

[method ile main method aynı Classta ise direk method ismi yazılarak kullanılabilir, 196](#_Toc72245826)

[Class ismini yazıp "." koyduğumuzda o classtaki Constructor variable'larına ulaşabiliriz. Eğer bir Variable veya method static olarak oluşturulmuşsa, o variable yada method'a objekt oluşturmadan sadece class ismini kullanarak ulaşabilirsiniz. 198](#_Toc72245827)

[static methodları objekt üzerinden çağırmaya gerek yok class ismi üzerinden çağırmak daha pratiktir. 198](#_Toc72245828)

[System.out.println(Constructors03.ad); // Ali Can 198](#_Toc72245829)

[int num1=12; 200](#_Toc72245830)

[static int num2=22; 200](#_Toc72245831)

[Static methodların içinde sadece static elemanlar kullanılabilir. num1 static olmadığından main method'da static olduğundan main method içerisinde kullanılamaz. 200](#_Toc72245832)

[num2 static olduğu için main metod içinde kullanmamız problem olmuyor.num1++; nın önündeki commenti kaldırırsanız Java num1++; nın altını kırmızı çizer. 200](#_Toc72245833)

[instance variable aynı classta object oluşturulmadan çağrılabilir. 201](#_Toc72245834)

[Static olmayanlar static içinde kullanılamazlar. bunun için object oluşturmanız lazım. 201](#_Toc72245835)

[Static bir variable'i object ile çağırırsak çalışır ama tavsiye edilmez,calışır ama compile error gösterir altını sarı olarak çizer. 201](#_Toc72245836)

[Return type yoksa isim class ismi ile aynı ise bu Constructor dır. 201](#_Toc72245837)

[metod static ise class ismi ile başka bir classtan bu metod çağırılabilir. 201](#_Toc72245838)

[Static method içerisinde static ile tanımlama yapmıyoruz. 201](#_Toc72245839)

[local variable'lara değer ataması yapılmak zorunda. 201](#_Toc72245840)

[Constructor da access modifier olabilir ama return type olamaz(public Constructor05() {} gibi) 201](#_Toc72245841)

[Static variable oluşturmanın diğer bir sebebi; ne kadar az variable kullanılırsa hafızada o kadar az yer kaplar static olunca daha az variable kullanılmış olur. 201](#_Toc72245842)

[18. DERS ( THİS, STATİC KEYWORD ) 202](#_Toc72245843)

[THİS 202](#_Toc72245844)

[this kullandığım zaman içinde bulunduğum class'taki instance veya static variable'lara ulaşmak istiyorum demektir.this constructorlar arasında geçiş yapmanın yoludur. 202](#_Toc72245845)

[this(); // bu classın içinde parametresiz constructor'ı çağır demektir.this() kullanılacaksa mutlaka Constructor'ın ilk satıra yazılmalıdır aksi takdirde compile Time Error verir. 202](#_Toc72245846)

[STATIC BLOCK 206](#_Toc72245847)

[Eğer bir Variable oluştururken hesap yapmamız gerekirse "static block" oluşturmak iyidir. Yani; int alan=en\*boy demek yerine "static block" oluşturup bu işlemleri static block'in içine yazmak daha iyidir. 206](#_Toc72245848)

["static block" variable'lara değer atamak içinde kullanılabilir. "static block" Class'ın içinde bütün methodların(main dahil) ve Constructorların dışında oluşturulmalıdır."static block" Class oluşturulduğu zaman oluşturulur."static block" tüm methodlardan önce çalıştırılır.Birden fazla static block varsa üstteki önce çalışır.Static class oluşturulabilir ama alt classlar(inner class) static Class olabilir, üst class'lar static olamazlar. Static class main method dışında olması gerekir. 206](#_Toc72245849)

[static { 206](#_Toc72245850)

[pi=3.14; 206](#_Toc72245851)

[} 206](#_Toc72245852)

[static { 206](#_Toc72245853)

[alan=en\*boy; 206](#_Toc72245854)

[} 206](#_Toc72245855)

[19.DERS (ARRAYS ) 207](#_Toc72245856)

[Birden fazla varriable'i aynı anda kullanmak istediğimiz zaman arrays kullanıyoruz. 207](#_Toc72245857)

[Array'ler object'tir bu yüzden Heap Memory'de depolanır. 207](#_Toc72245858)

[Array'ler object'tir. bu yüzden valueların yanında methodlar da vardır. 207](#_Toc72245859)

[Array'ler object'tir bu yüzden runtime'a oluşturulurlar. 207](#_Toc72245860)

[Primitiveleri Java oluşturur. non primitive lardan ise istediğimiz kadar üretebiliriz.Bu non primitivelerden biriside arraysdir.Arraysler heapte depolanır.Heap büyük bir alan olduğu için bulunabilmesi için adreslenmesi gerekir. 207](#_Toc72245861)

[Arrays içinde tüm variable'lar aynı data type'ınde olmalıdır. 207](#_Toc72245862)

[Arrays sadece Primitive type (byte, short...8 tane) ve objeckt referans konulabilir. 207](#_Toc72245863)

[Arrays'ın içinde Stringin adresi olur, kendisi olmaz.Array'in içine stackdekiler konulabilir. 207](#_Toc72245864)

[Bir array oluşturmadan önce o Array'in içine kaç variable koyacağımıza karar vermeliyiz.Bir Array içine koyabileceğimiz variable sayısına o Array'in "length"i denir. 207](#_Toc72245865)

[O Array içine length'den fazla variable koyamayız. 207](#_Toc72245866)

[int x; declare denir. 207](#_Toc72245867)

[Array oluşturmak için iki yol vardır: 207](#_Toc72245868)

[1) int myArray[ ]; daha cok kullanılır 207](#_Toc72245869)

[2) int [ ]myArray; Dat Type isim [ ]; 207](#_Toc72245870)

[[ ] Array sembolü. Bu yazılınca Stack Memory'de bir adres oluşturulur.Eclipse de run butonuna basılınca Array oluşturulur. 207](#_Toc72245871)

[BİR ARRAY NASIL OLUŞTURULUR? 207](#_Toc72245872)

[Array bir objeckt olduğu için new kullanılarak oluşturulur. 207](#_Toc72245873)

[int myArray [ ] = new int[7]; 207](#_Toc72245874)

[7 lenght uzunluğudur, değer yazılmazsa compile error verir. 207](#_Toc72245875)

[int olarak öncesinde bir değer atanmamışsa default değer olarak 7 tane 0 atar.7 değer atanacak siz 3 değer atarsanız gerisini yazmazsanız 2 3 4 0 0 0 0 yapar. 208](#_Toc72245876)

[boolean b[]=new boolean[3]; false false false gibi çünkü boolean default değeri falsedir. 208](#_Toc72245877)

[array'ler index kullanır. bu nedenle array içindeki elamanlardan birini görmek istiyorsak index ini yazarak print yapabiliriz, index olmadan sadece array ismini yazarsak reference'ını (adresini) görürürüz. 208](#_Toc72245878)

[Integer Array oluşturalım 208](#_Toc72245879)

[int arr1[]=new int[5]; 208](#_Toc72245880)

[System.out.println(arr1); // [I@1540e19d yazar bu adrestir. 208](#_Toc72245881)

[// array in ismini yazarak yazdırmaya çalışırsak adresi alırız. 208](#_Toc72245882)

[Array'de olmayan indexe değer ataması yapılırsa Run Time Error alınır. 209](#_Toc72245883)

[Array'de olmayan indexi kullanmaya çalışırsak ArrayIndexOutOfBoundsException hatası alırız. 209](#_Toc72245884)

[Array'in length'ini bulmak için "ArrayIsmi.length" yazmak yeterlidir. 210](#_Toc72245885)

[Stringlerde a.length() arraylerde ise sadece a.length şeklinde yazılıyor. 210](#_Toc72245886)

[Arraydeki son elemanı yazdırmak için length kullanınız. 210](#_Toc72245887)

[System.out.println(arr3[arr3.length-1]); 210](#_Toc72245888)

[FLAG / COUNT 211](#_Toc72245889)

[elemanları 1,2,3,4,5 bir int array oluşturunuz.Bu arrayde 3 elemanının olup olmadığını kontrol ediniz.3 varsa ekrana "3 var" yazdırın.3 yoksa "3 yok " yazdırın. 211](#_Toc72245890)

[Array oluşturup eleman eklemek 211](#_Toc72245891)

[int arr[]= {1,2,3,4,5}; 211](#_Toc72245892)

[int count=0; // flag(bayrak) denir. 211](#_Toc72245893)

[for (int i=0; i<5;i++) { 211](#_Toc72245894)

[if(arr[i]==3) { 211](#_Toc72245895)

[count++; 211](#_Toc72245896)

[} 211](#_Toc72245897)

[} 211](#_Toc72245898)

[if(count>0) { 211](#_Toc72245899)

[System.out.println("3 var"); 211](#_Toc72245900)

[}else { 211](#_Toc72245901)

[System.out.println("3 yok"); 211](#_Toc72245902)

[} 211](#_Toc72245903)

[Arrays.toString() methoduna parametre olarak array'in ismini yazarsanız o array'in tüm elemanlarını ekranda görürsünüz. 214](#_Toc72245904)

[System.out.println(Arrays.toString(arr)); 214](#_Toc72245905)

[[ ] işareti olmadan elemanları yazdırmak istiyorsak for loop kullanıyoruz. 216](#_Toc72245906)

[for(int i=0; i<arr.length; i++) { 216](#_Toc72245907)

[System.out.print(arr[i]+ ","); 216](#_Toc72245908)

[} 216](#_Toc72245909)

[Arrays.sort() methoduna parametre olarak array'in ismi yazılırsa o Array'in elemanlarını küçükten büyüğe sıralanmış olur. 216](#_Toc72245910)

[Arrays.sort(arr); 216](#_Toc72245911)

[charlar sıralanırken java ASCII kodlarını kullanır.Büyük harflerin ASCII kodları küçük harflerden küçüktür,bu nedenle büyük harfler öne geçer. 217](#_Toc72245912)

[booleanlar için sort methodu kullanılamaz. 217](#_Toc72245913)

[System.out.println(arr1==arr2); 218](#_Toc72245914)

[// false verir çünkü referenc'ları karşılaştırır. 218](#_Toc72245915)

[System.out.println(Arrays.equals(arr1, arr2)); 218](#_Toc72245916)

[// true verir çünkü Arrays.equal() methodu sadece değerleri ve indexleri kontrol eder adreslere bakmaz. 218](#_Toc72245917)

[binarySearch() methodu aranan eleman var olduğunda o elemanın indexini return eder. 219](#_Toc72245918)

[Önemi Not: binarySearch methodunu kullanmadan önce sort() methodunu kullanmak zorundasınız aksi takdirde binarySearch methodu manalı bir sonuç vermez,verdiği sonuc güvenilir olmaz. 219](#_Toc72245919)

[int arr[]= {1,12,12,8,2,4,3}; 219](#_Toc72245920)

[// yukarıda verilen array'de 3 sayısı eleman olarak var mı? 219](#_Toc72245921)

[Arrays.sort(arr); 219](#_Toc72245922)

[Arrays.binarySearch(arr, 3); 219](#_Toc72245923)

[System.out.println(Arrays.binarySearch(arr, 3)); 219](#_Toc72245924)

[sonuc 2 yazar. önce sıralama yapar sonra bize indexi döndürür çünkü binary yapıyor bize sayılar döndürecek. 219](#_Toc72245925)

[aynı elemandan birden fazla olduğunda ilkinin indexini verir. elemanı bulduktan sonra break yapar. 219](#_Toc72245926)

[System.out.println(Arrays.binarySearch(arr, 5)); //-5 219](#_Toc72245927)

[binarySearch() methodu olmayan elemanlar için negatif sayılar return eder. 219](#_Toc72245928)

[negatif olması o elemanın arrayde olmadığı anlamına gelir 219](#_Toc72245929)

[-5 o eleman olsaydı kaçıncı eleman olduğu anlamındadir. 219](#_Toc72245930)

[{1,2,3,4,18,12,12} var olsaydı 4 ten hemen sonra olurdu buda 5.eleman demektir. 220](#_Toc72245931)

[21. DERS ( MULTİDİMENSİONALARRAY ) 223](#_Toc72245932)

[split bir string methodudur. str.split( ); methodunu seçip parametre içine hangi karakteri yazarsak boşluklara göre Stringi parçalayıp kaç parça olduğunu bize yazar. 223](#_Toc72245933)

[Girilen karakteri çıkarır aradaki kelimeleri sayar. " " koyduğumuzda kelimeler ortaya çıkar. aşağıdak bu şekilde 4 tane String oluşmuş olur. Split methodu kullandığınızda mutlaka bir Array oluşturmanız gereklidir. 223](#_Toc72245934)

[MULTİDİMENSİONAL ARRAY 225](#_Toc72245935)

[int arr[][]=new int[3][4]; 225](#_Toc72245936)

[System.out.println(Arrays.toString(arr)); // [[I@1540e19d, [I@677327b6, [I@14ae5a5] 225](#_Toc72245937)

[System.out.println(Arrays.deepToString(arr)); // .deepToString sonuna kadar git tüm elemanları yazdır demektir. 225](#_Toc72245938)

[// [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]] yazdırır. 225](#_Toc72245939)

[arr.length bize iç içe kaç tane array olduğunun sayısını veriyor.(ilk index parantez ) 227](#_Toc72245940)

[System.out.println(arr.length); // kaç ayrı array grubu varsa onun sayısını verir(ilk index) 227](#_Toc72245941)

[// arr[i].length ikinci index parantezinin uzunluğu için 227](#_Toc72245942)

[System.out.println(arr[1].length);// ikinci array grubu içerisindeki eleman sayısını verir. 227](#_Toc72245943)

[ARRAY LİST 233](#_Toc72245944)

[Array'lerde lenght sabittir, değiştirilemez. ArrayListler'de (List) length esnektir. Siz eleman ekledikçe List Length'ini artırır. Siz eleman sildikçe List length'ini azaltır. Hem List hem de ArrayList üzerine gelip java.util olarak import yapıyoruz. 233](#_Toc72245945)

[ArrayList<String> list = new List<>(); OLMAZ 233](#_Toc72245946)

[ArrayList<String> list = new ArrayList<>(); OLABİLİR 233](#_Toc72245947)

[List<String> list = new ArrayList<>(); GENEL KULLANIM ŞEKLİ 233](#_Toc72245948)

[List'e eleman eklemek için add() methodu kullanılır. 233](#_Toc72245949)

[list01.add("Ali"); 233](#_Toc72245950)

[List'ler data type olarak primitive'leri kabul etmez. Primitive'lerde non-Primitive yapmak için Wrapper class'ları kullanırız. 235](#_Toc72245951)

[List<Integer> list01=new ArrayList<>(); 235](#_Toc72245952)

[Bir list'in içinde elaman olup olmadığını kontrol etmek için isEmpty methodu kullanılır. Boş ise true , dolu ise false return eder. 235](#_Toc72245953)

[System.out.println(list01.isEmpty()); 235](#_Toc72245954)

[Bu liste eleman ekleyin. list01.add(123); 235](#_Toc72245955)

[List'den eleman silmek için remove() methodu kullanılır. iki remove methodu var biri indexle çalışıyor 235](#_Toc72245956)

[list.remove("Ali"); //Birden fazla ayni eleman varsa ilkini siler 237](#_Toc72245957)

[list.remove("Kemal");//Listede olmayan elemanlar icin error vermez 237](#_Toc72245958)

[System.out.println(list.remove("Ali"));//true 237](#_Toc72245959)

[System.out.println(list.remove("Kemal"));//false 237](#_Toc72245960)

["Ayse" elemanini "Aysegul" yapin. set() methodu kullanilir. 237](#_Toc72245961)

[System.out.println(list.set(0, "Aysegul"));//Ayse yani degisimden onceki hali ekrana yazdirir. 237](#_Toc72245962)

[System.out.println(list);//[Aysegul] 237](#_Toc72245963)

[list.contains("B"); list B elemanını içeriyor mu demektir. varsa true yoksa false return eder. 239](#_Toc72245964)

[list.contains("B"); 239](#_Toc72245965)

[System.out.println(list.contains("B")); // true verir. 239](#_Toc72245966)

[Collection ==> bir araya getirilmiş parçalar. ayakkabılık, kalemlik, koleksiyon gibi. Collection bir class, içerisinde methodlar var. 239](#_Toc72245967)

[Collections.sort(list); 239](#_Toc72245968)

[clear() methodu kullanarak, list’deki tum elemanları siliniz. 241](#_Toc72245969)

[list.clear(); 241](#_Toc72245970)

[22. DERS ( ARRAY LIST) 242](#_Toc72245971)

[equals() methodu iki listin birbirnie eşit olup olmadığını kontrol eder, List'ler eşit ise "true", değilse "false" return eder. 242](#_Toc72245972)

[equals() methodu aynı index'de aynı eleman olup olmadığını kontrol eder, equals() methodu objelerin reference (adreslerine) larına bakmaz. 242](#_Toc72245973)

[System.out.println(list1.equals(list2)); // true 242](#_Toc72245974)

[toArray() methodu listi Array'e çevirmek için kullanılır. 243](#_Toc72245975)

[I. YOL: to Array() method'unun içinde parametre olarak new String[0] kullanınız. 243](#_Toc72245976)

[String arr[]=list.toArray(new String[0]); 243](#_Toc72245977)

[II. YOL: Oluşturduğunuz array'in data type'ini Object olarak seçin. 243](#_Toc72245978)

[Object class, Java'da bütün class'ların parent'idir. Yani Object class orta atadır.Object class, Java'da parenti olmayan tek class'dır.String, Object class'ın child'i olduğundan data type olarak bazı durumlarda String yerine Object kullanırız. 243](#_Toc72245979)

[Object[] arr1=list.toArray(); 243](#_Toc72245980)

[asList( ) methodu array'leri liste çevirmek için kullanılır. 243](#_Toc72245981)

[asList( ) methodu parametre olarak array'in ismini kabul eder. 243](#_Toc72245982)

[List<String> list1 = Arrays.asList(arr2); 243](#_Toc72245983)

[Array'den liste çevirme yaptığınızda, elde ettiğiniz list uzunluk olarak esnek değildir. Yani array'den oluşturduğunuz liste ekleme ve çıkarma yapamazsınız. 244](#_Toc72245984)

[FOR EACH LOOP 245](#_Toc72245985)

[int arr[]= {12, 21, 13, 43}; 245](#_Toc72245986)

[for loop kullanarak elemanları ekrana yazdır. 245](#_Toc72245987)

[for (int i = 0; i < arr.length; i++) { 245](#_Toc72245988)

[System.out.print(arr[i]); 245](#_Toc72245989)

[} 245](#_Toc72245990)

[for each loop kullanarak elemanları ekrana yazdır. 245](#_Toc72245991)

[for(int w:arr) { 245](#_Toc72245992)

[// w 0.indexten başlayıp son indexe kadar uygular. 245](#_Toc72245993)

[System.out.print(w); 245](#_Toc72245994)

[} 245](#_Toc72245995)

[// {{1, 2}, {5}, {6, 7, 8}} arrayindeki tüm elemanların toplamını bulunuz. 246](#_Toc72245996)

[int arr2[ ][ ]={{1, 2}, {5}, {6, 7, 8}}; 246](#_Toc72245997)

[int sum1=0; 246](#_Toc72245998)

[for(int[ ] w:arr2) { 246](#_Toc72245999)

[for(int z: w) { 246](#_Toc72246000)

[sum1=sum1+z; 246](#_Toc72246001)

[} 246](#_Toc72246002)

[} 246](#_Toc72246003)

[listaddAll() methodu iki listi birleştirmek için kullanılır. 251](#_Toc72246004)

[index kullanmadan addAll()yazım sırasına göre bir ekleme yaparsınız.list'i istediğiniz index'e eklemek istiyorsanız addAll(index,element) seçmeniz lazım. 251](#_Toc72246005)

[list1.addAll(list2); 251](#_Toc72246006)

[list4.addAll(1, list3);; // [2, 1, 3] 252](#_Toc72246007)

[23.DERS ( LocalDate ) 253](#_Toc72246008)

[Bugünün tarihini ekrana yazdıralım. 253](#_Toc72246009)

[System.out.println(LocalDate.now()); 253](#_Toc72246010)

[System.out.println(LocalDateTime.now()); 253](#_Toc72246011)

[2020-05-20T17:03:01.519 buradaki "T" tarih ile zamanı ayırmak için konuluyor. 253](#_Toc72246012)

[Date Manipulation (Date üzerinde değişiklikler yapmak) 254](#_Toc72246013)

[LocalDate date=LocalDate.now(); 254](#_Toc72246014)

[System.out.println(date.plusDays(1)); 254](#_Toc72246015)

[System.out.println(date.plusMonths(2)); 254](#_Toc72246016)

[System.out.println(date.plusYears(3)); 254](#_Toc72246017)

[System.out.println(date.plusWeeks(4)); 254](#_Toc72246018)

[System.out.println(date.minusDays(3)); 254](#_Toc72246019)

[Time Manipilation (Zamanı değiştirme) 255](#_Toc72246020)

[LocalTime time = LocalTime.now(); 255](#_Toc72246021)

[System.out.println(time); 255](#_Toc72246022)

[System.out.println(time.plusSeconds(45)); 255](#_Toc72246023)

[Date formatını değiştirmek 256](#_Toc72246024)

[Java minute için "m" kullanır, ay için "M" kullanır. 256](#_Toc72246025)

[LocalDate date=LocalDate.now(); 256](#_Toc72246026)

[DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd.MMM.yyyy"); 256](#_Toc72246027)

[System.out.println(dtf.format(date)); // 08.Nis.2021 256](#_Toc72246028)

[Time Formatını değiştirme 256](#_Toc72246029)

[HH kullanırsanız 24 saatlik dilimi kullanır. hh olursa pm am kullanır. 256](#_Toc72246030)

[LocalTime time=LocalTime.now(); 256](#_Toc72246031)

[DateTimeFormatter dtf3=DateTimeFormatter.ofPattern("hh:mm"); 256](#_Toc72246032)

[System.out.println(dtf3.format(time.plusHours(3))); // 12:39 256](#_Toc72246033)

[Başka ülkenin local zamanını almak 256](#_Toc72246034)

[time.now(zone); bir ülkenin zone değerini GMT-3 257](#_Toc72246035)

[System.out.println(LocalTime.now(ZoneId.of("America/Sao\_Paulo"))); // 04:42:18.784 257](#_Toc72246036)

[24.DERS ( VARIABLE ARGUMENTS ) 258](#_Toc72246037)

[Varargs: Variable Arguments 258](#_Toc72246038)

[Bir method oluşturduğumuzda argüment olarak istediğimiz kadar argüment girebilmemizi sağlar. int den sonra ... koyunca java bunun Varargs olduğunu anlar.sonu açık olan bir Array gibi düşünebiliriz. 258](#_Toc72246039)

[countNum(5,7,9); // length uzunluğu 3 258](#_Toc72246040)

[sumNums(1,2,3,4,5,6,7,8,9,10); 258](#_Toc72246041)

[printName("Ayşe Yılmaz Kahraman"); // Ayşe Yılmaz Kahraman 258](#_Toc72246042)

[public static void countNum(int... a) { // Vararg'ın ismi a 258](#_Toc72246043)

[System.out.println(a.length); 258](#_Toc72246044)

[} 258](#_Toc72246045)

[public static void sumNums(double... b) { // Varargs'ın içine koyduğumuz tüm sayıları toplayan prog. 258](#_Toc72246046)

[double sum=0; 258](#_Toc72246047)

[for(double w:b) { // w 0.indexten başlayıp son indexe kadar uygular. 258](#_Toc72246048)

[sum=sum+w; 258](#_Toc72246049)

[} 258](#_Toc72246050)

[System.out.println(sum); 258](#_Toc72246051)

[} 258](#_Toc72246052)

[public static void printName(String...n) { 258](#_Toc72246053)

[for(String w:n) { 258](#_Toc72246054)

[System.out.println(w+" "); 258](#_Toc72246055)

[}}} 258](#_Toc72246056)

[VARARGS'IN YANLIŞ KULLANIMLARI 259](#_Toc72246057)

[Kural 1: Varargs her zaman son parametre olmalıdır. çünkü varargs sonsuz elemana sahip for loop ta varargs bitmiyor ki diğer parametreye geçebilsin çünkü ilk parametreyi varargs yaparsanız girdinğiniz her değer varargs'ın içine düşer, siz asla ikinci parametreye değer verememiş olursunuz bu nedenle hata verir. 259](#_Toc72246058)

[Kural 2: parametre olarak 1'den fazla varargs kullanamazsınız çünkü varargs en sonda olmalıdır, en sonda olmak tek olmayı gerektirir. 259](#_Toc72246059)

[public static void valueChar(String s, char... c) { 259](#_Toc72246060)

[for(char w: c) { 259](#_Toc72246061)

[System.out.print(s); 259](#_Toc72246062)

[System.out.println(w); 259](#_Toc72246063)

[} 259](#_Toc72246064)

[} 259](#_Toc72246065)

[// public static void product(int... x, int y) { // yanlış 259](#_Toc72246066)

[// public static void product1(int... x, int... y){// yanlış 259](#_Toc72246067)

[ACCESS MODİFİER 261](#_Toc72246068)

[Bir class, constructor, variable, ve method’a erişimi kısıtlamak veya genişletmek icin Access Modifier’lar kullanılır. 261](#_Toc72246069)

[4 tane Access Modifier vardır. 261](#_Toc72246070)

[1) private: Class'lar private olamaz. Class içinde Constructor, method ve variable için kullanılabilir.sadece class içindekiler tarafından kullanabilir. 261](#_Toc72246071)

[2) default (Package Private) herhangi bir Access Modifier yazmazsak,Java Access Modifier’i default olarak kabul eder. Sadece aynı package içindeki classlar tarafından görülüp ulaşılabilir. 261](#_Toc72246072)

[3) protect: Class'lar protected olamaz. Aynı package içindeki classlar ve diğer package'lardan child Classlar tarafından görülüp ulaşılabilir. 261](#_Toc72246073)

[4) public (herkese açık):class’a, method’a, variable’a veya constructor’a her yerden ulaşılabilir,herhangi bir kısıtlama koymaz. 261](#_Toc72246074)

[25.DERS ( StringBuilder ) 262](#_Toc72246075)

[StringBuilder class'ı String üretmeye ve üretilen Stringleri manipüle etmeye yarar. 262](#_Toc72246076)

[String class'ı kullanarak bir String üretirseniz eğer yaptığınız değişiklikler ilk ürettiğiniz Stringi etkilemez.Ama illaki ilk ürettiğim string değişsin isterseniz atama yapmak zorundasınız. String class'ını kullanarak ürettiğiniz stringler atamasız değiştirilemedeği için String class'ına "immutable" denir. 262](#_Toc72246077)

[String ile uretilen Stringlerde yapilan degisiklikler ilk uretilen (orijinal olan) Stringi degistirmezken StringBuilder ile uretilen Stringlerde yapilan degisiklikler ilk uretilen orijinal Stringi degistirir. Yani Stringlerde degisim ilk Stringi etkilemez ama StringBuilderde etkilenir.StringBuilder Class'ını kullanarak String üretirseniz yaptığınız değişiklikler ilk ürettiğiniz String'i etkiler. Atama yapmaya gerek yoktur. StringBuilder Class'ı değişimi yansıttığı için ona "mutable" denir. 262](#_Toc72246078)

[String str="Ali"; 262](#_Toc72246079)

[str.substring(1); // ==>li 262](#_Toc72246080)

[System.out.println(str);// ==> Ali yazar çünkü atama yapılmadı. 262](#_Toc72246081)

[atama yapmadan değişiklik uygulansın istiyorsanız StringBuilder Class'ı kullanılmalıdır. 262](#_Toc72246082)

[StringBuilder strBld=new StringBuilder("Ayşe"); 262](#_Toc72246083)

[strBld.append("Can"); // append () methodu String'e yeni bir String eklemek için kullanılır. Concatenation yapar. 262](#_Toc72246084)

[System.out.println(strBld);// AyşeCan 262](#_Toc72246085)

[StringBuilder strBld3=new StringBuilder(7); // uzunluğu 7 chracter olan bir String ürettim. 263](#_Toc72246086)

[strBld3.append("Ali"); 263](#_Toc72246087)

[strBld3.append("Kahraman"); 263](#_Toc72246088)

[System.out.println(strBld3); // AliKahraman 263](#_Toc72246089)

[} 263](#_Toc72246090)

[substring() methodu atama yapmadan Stringi değiştirmez. 264](#_Toc72246091)

[StringBuilder str1=new StringBuilder("animals"); 264](#_Toc72246092)

[str1.substring(3); 264](#_Toc72246093)

[System.out.println(str1); // ekrana animals yazar 264](#_Toc72246094)

[str1=str1.substring(3); altını çizer hata verir çünkü substring methodu String class'ından gelir ve bir String return eder. halbuki str1 String değil, StringBuilder dır. java data typeleri uyuşmadığından dolayı atamayı kabul etmez.Bu şekilde atama yapmaya izin vermez ama ekrana yazdırmaya izin verir. 264](#_Toc72246095)

[insert () methodu istenen indexe istenen characteri ekmeleye yarar.insert () methodu append() gibi StringBuilder'i direkt değiştirir. 265](#_Toc72246096)

[str1.insert(0, "X"); 265](#_Toc72246097)

[System.out.println(str1); // Xanimals 265](#_Toc72246098)

[delete() methodu istenen index aralığındaki characteri siler. 265](#_Toc72246099)

[str1.delete(0,1); // StringBuilder classından geldiği için atama ihtiyacı duymadan direk değiştirir. 265](#_Toc72246100)

[deleteCharAt(); istenen indexteki characteri siler. 265](#_Toc72246101)

[str1.deleteCharAt(6);// s harfini silmeli 265](#_Toc72246102)

[System.out.println(str1);// animal 265](#_Toc72246103)

[reverse() Stringi tersten yazdırır. 265](#_Toc72246104)

[str1.reverse(); 265](#_Toc72246105)

[System.out.println(str1);// lamina 265](#_Toc72246106)

[toString() methodu StringBuilder'i String'e çevirmek için kullanılır. 265](#_Toc72246107)

[str1.toString(); 265](#_Toc72246108)

[System.out.println(str1);// str1 artık StringBuilder değil String'dir. 265](#_Toc72246109)

[Stringbuilder'lar daha hızlı çalışır.Bu yüzden StringBuilder kullanmayı tercih edelim. 265](#_Toc72246110)

[26.DERS ( ENCAPSULATION, INHERİTANCE ) 267](#_Toc72246111)

[ENCAPSULATION 267](#_Toc72246112)

[Programcılıkta herşeyi herkesin görmesi iyi değildir.Program kodları herkese açık olunca başkaları içine girip değişiklik yapıp programı bozabilir.Program yazarken bazı şeyleri mesela variable'lerı saklamak isteriz.Bu saklama işlemi başkalarına kapalı iken bizim için açık, ulaşılabilir olmalı. 267](#_Toc72246113)

[Encapsulation data saklama(data-hiding) yöntemidir. Encapsulation iki stepte yapılır. 267](#_Toc72246114)

[1) Data'yı(Variable, method) private yapmalısınız.Bu şekilde başkaları ulaşamaz. 267](#_Toc72246115)

[2) bu sakladığım dataya kendim ulaşabilmem için public olan getter() ve setter() methodlar üretmeliyim. getter()methodu data'yı sadece okumamıza yarar, getter() methodu data'da değişiklik yapamaz. setter() methodu data'yı değiştirmemize yarar. 267](#_Toc72246116)

[GETTER() METHODU ÜRETMEK İÇİN; 267](#_Toc72246117)

[1) Access Modifier public olmalı. 267](#_Toc72246118)

[2) Return type'i variable'in return type'i ile aynı olmalı. 267](#_Toc72246119)

[3) Method ismi "get+variable ismi" şeklinde olmalı return type boolean ise getter() method ismi "is" ile başlar "get" kullanılmaz. 267](#_Toc72246120)

[setter() methodu; Datayı değiştirmek için kullanılır. void iki yerde kullanılır. konsola yazdırma ve action'larda kullanılır. void kullanılınca return kullanılmaz. setter() methodun data type'i her zaman void'dir. action için. parametre yerine mutlaka birşey yazılmalıdır. 267](#_Toc72246121)

[SETTER() ÜRETMEK İÇİN; 268](#_Toc72246122)

[1) Access modifier public olmalı 268](#_Toc72246123)

[2) Return type void olmalı. 268](#_Toc72246124)

[3) Method ismi "set+variable ismi" şeklinde olmalı 268](#_Toc72246125)

[4) Parametre kullanılmalı() 268](#_Toc72246126)

[5) setter() methodların ismi boolean'lar içinde set ile başlar. 268](#_Toc72246127)

[public String getKimlikNo() { 268](#_Toc72246128)

[return kimlikNo; 268](#_Toc72246129)

[} 268](#_Toc72246130)

[public boolean isSoguk() { 268](#_Toc72246131)

[return soguk; 268](#_Toc72246132)

[} 268](#_Toc72246133)

[public void setKimlikNo(String kimlikNo) { 268](#_Toc72246134)

[this.kimlikNo =kimlikNo; 268](#_Toc72246135)

[} 268](#_Toc72246136)

[public void setSoguk(boolean soguk) { 268](#_Toc72246137)

[this.soguk = soguk; 268](#_Toc72246138)

[} 268](#_Toc72246139)

[GETTER() VE SETTER() METHODLARINI KISA YOLDAN OLUŞTURMAK 269](#_Toc72246140)

[variable "ilkHarf" üzerine sağ tıkla Source==>Generate Getter and Setter... tıkla 269](#_Toc72246141)

[Variable'a atanan değerin okunmasını istemiyorsanız o variable'la alakalı getter() oluşturmamalısınız. Variable'a atanan değerin değiştirilmesini istemiyorsanız o variable ile alakalı setter() oluşturmamalısınız. 269](#_Toc72246142)

[mesela kendi email adresimizi girdik ve başkasının okumasını görmesini istiyor, değiştirmesini istemiyorsak sadece getter() kullanır hiç Setter() kullanmazsanız setter() işaretlenmezse variable değerleri değiştirilemez demektir. Bu tarz Class'lara "Immutable Class" denir. Sadece Setter() kullanır Getter() kullanmazsanız okunamaz ama değiştirilebilir demektir. 269](#_Toc72246143)

[başka bir class'taki variable'a object oluşturarak ulaşabilirim. 269](#_Toc72246144)

[Encapsulation01 obj = new Encapsulation01(); 269](#_Toc72246145)

[System.out.println(obj.getKimlikNo()); // başka klasstaki variable okuduk. 269](#_Toc72246146)

[obj.setKimlikNo("10000000"); // başka bir class'taki variable değiştirdik.Bu değişiklik sadece bu Class'a özgüdür. asıl Classta değişiklik olmaz. 269](#_Toc72246147)

[INHERITANCE 271](#_Toc72246148)

[Inheritance kelime manası olarak Miras demek. İki Class arasında Parent - Child ilişkisi varsa inheritance'dan bahsedilebilir. Childlar tarafından parent Class'taki variable'lar kullanılabilir. Aynı package'da ise default Access modifier lar parent Class'lardan variable kullanabilir. 271](#_Toc72246149)

[Inheritance Kuralları: 271](#_Toc72246150)

[1) Inheritance Public ve Protected data'lar için problemsiz çalışır. 271](#_Toc72246151)

[2) Inheritance default data'lar(variable'lar) için child ile parent aynı package'da ise çalışır. 271](#_Toc72246152)

[3) Inheritance Private data'lar için çalışmaz. 271](#_Toc72246153)

[Child ve Parent ilişkisi nasıl kurulur? 271](#_Toc72246154)

[Class oluştururken "extends" keyword kullanırız. 271](#_Toc72246155)

[Inheritance01, Inheritance02'nin parent'ı. 272](#_Toc72246156)

[extends'den sonraki class parent, önceki class child olur. 272](#_Toc72246157)

[Bir parentin bir çok childi olabilir (hierarchical) Inheritance bir childin iki parenti olamaz.java multiple inharitance'a müsade etmiyor (single inheritance) 272](#_Toc72246158)

[Multilevel Inheretance: child==>parent Class==>parent Class çocuk dede ilişkisi gibi cocuğu babaya extends yapıyoruz sonra babayı dedeye extends yapıyoruz. 272](#_Toc72246159)

[public class Inheritance02 extends Inheritance01 { 272](#_Toc72246160)

[27.DERS ( CONSTRUCTOR CALL ) 277](#_Toc72246161)

[inheritenceda main method içerisinde object oluşturulunca önce grandparent sonra parent'ın sırasıyla constructorlarını kullanır ve çalıştırır. kaç tane üst parent varsa hepsini return eder. altta constructor oluşturulunca "Dog();" parentlerda "()" parametresiz ne kadar constructor varsa hepsini çağırıyor. en büyük atasından aşağıya doğru çalışır. 277](#_Toc72246162)

[dog: object ismi Dog():Constructor ismi 277](#_Toc72246163)

[main methodu içine yazılan methodlar çalışır, yazılmayanlar çalışmaz. 277](#_Toc72246164)

[public class Dog extends Mammal { 277](#_Toc72246165)

[public boolean sadik = true; 277](#_Toc72246166)

[public static void main(String[] args) { 277](#_Toc72246167)

[Dog dog = new Dog(); 277](#_Toc72246168)

[} 277](#_Toc72246169)

[public void bark() { 277](#_Toc72246170)

[System.out.println("Köpekler havlar"); 277](#_Toc72246171)

[} 277](#_Toc72246172)

[Dog(){ // Parent'daki parametresiz constructor'ı çağır demektir. 277](#_Toc72246173)

[System.out.println("Dog parametresiz constructor"); 277](#_Toc72246174)

[} 277](#_Toc72246175)

[} 277](#_Toc72246176)

[super(); 278](#_Toc72246177)

[// super() keyword'unu kullanırsanız tekrar parenta git demektir.super() kullanılacaksa mutlaka ilk satırda olmalıdır. super() parenta git demektir.super() ve this() aynı constructor içinde aynı anda kullanılamaz. çünkü ikisininde ilk satırda olma zorunluluğu var. aynı class içindeki parametresiz constructorı çağırmak için kullanıyoruz. 278](#_Toc72246178)

[Mammal(int age){ // inheritance olduğu için önce Animal Class'a gider Animal'da parametreli constructor olmadığı için aşağı satıra iner 278](#_Toc72246179)

[super(); // tekrar parenta yani Animal Class'ına gider orada parametresiz constructor araştırır varsa parametresiz constructor'ı çalıştırır. 278](#_Toc72246180)

[System.out.println("Mammal parametreli constructor"); 278](#_Toc72246181)

[} 278](#_Toc72246182)

[28.DERS ( OVERRİDİNG ) 280](#_Toc72246183)

[1)Parent'taki method'u method signature'ini degistirmeden method body'sini degistirerek Child class'da kullanmaya "Method Overriding" denir. 280](#_Toc72246184)

[2)Parent class'daki private method'lar override edilemezler 280](#_Toc72246185)

[3)final method'lar override edilemezler cunku; final methodlarin body'leri olabilecekleri son haldedir yani method body'leri degistirilemez. Halbuki overriding'in amaci method body'i degistirerek methodu kullanmaktir. Bu celiskiden dolayi Java final method'larin override edilmesine musaade etmez. 280](#_Toc72246186)

[4)static method'lar override edilemezler.Cunku; static methodlar ortak kullanima aciktir onun body'sini degistirmek herkesi etkiler bu yuzden Java static methodlarin override edilmesine musaade etmez. 280](#_Toc72246187)

[5)Child class'daki override edilmis method'un (Overriding Method) access modifier'i Parent class'daki override edilen method'un(Overridden Method) access modifier'indan daha dar olamaz. 280](#_Toc72246188)

[6)Child class'daki override edilmis methodun return type'i ya parent class'daki override edilmis methodun return type'i ile ayni olur veya onun child'larindan biri olabilir 280](#_Toc72246189)

[Overriding: uyarlamak demektir. 280](#_Toc72246190)

[Parent'taki bir methodu Child Class'ında kendimize uyarlayarak kullanmamıza Overriding deniyor. 280](#_Toc72246191)

[Parent içindeki method (ses) . bu methoda Method Overridden denir. Child içerisindeki method ses( ). bu methoda Method Overridding Method. 281](#_Toc72246192)

[Overriding yaptığımızda child in access modifier ile aynı olmalı veya daha geniş olmalıdır.Parent Access modifier default olduğunda, Child Access modifier default, protected, public olabilir.Child Parenta göre durum alır.Parent public olduğunda child default olamaz. Child parent'ı daraltamaz, sınırlandıramaz. 281](#_Toc72246193)

[IS-A VE HAS-A 281](#_Toc72246194)

[Child dan parenta giderken İs-a ilişkisi, parent'dan child'a gidersek has-a ilişkisi vardır. 281](#_Toc72246195)

[Dog IS-A Mammal == True Mammal HAS-A Dog ==> True 281](#_Toc72246196)

[Return typeları kullanırken bu is-a ve has-a ilişkisi olması lazım. Overriding'de is-a ilişkisi olması lazım. Child daki return type child olması lazım. 281](#_Toc72246197)

[OVERRİDİNG KURALLARI 281](#_Toc72246198)

[1) Method signature'ı(name ve parameters) değiştirmeyin. 281](#_Toc72246199)

[2) 4 Access Modifier var. private => default => protected => public. Overriding yaptığımızda child in access modifier ile aynı olmalı veya daha geniş olmalıdır. 281](#_Toc72246200)

[3) Overriding COVARİANT return type olmalıdır. yani İS-A olmaldırı. Child'dakinin return type parent'takinin return type'nın child'i olmalıdır. 281](#_Toc72246201)

[4) private,static and final methodları overriding yapılamazlar. private olan birşey o Class'a özel bir şeydir,Class'a özel olan başka Class'ta kullanılamaz.private overloading yapılabilir.static bütün objelere açıktır, herkese hitap eden bir özelliği var bu özelliği değiştiremezsiniz.final method artık son haline getirilmiş, tamamlanmış, değişime açık değil demektir. overriding olmaz. 281](#_Toc72246202)

[methodun başına final yazılarak kullanılır. 281](#_Toc72246203)

[OVERRİDİNG KONTROLÜ 281](#_Toc72246204)

[1) parent child ilişkisini kontrol et. extends yazıyor mu? 281](#_Toc72246205)

[2) method ismi aynımı 281](#_Toc72246206)

[3) parametreler aynı mı 281](#_Toc72246207)

[4) Accesse Modifier aynı olmalı veya kapsamalı 281](#_Toc72246208)

[5) return type'lar int is-a void ? değil false compilation error verir. 281](#_Toc72246209)

[Parametresiz Constructor kullanılmışsa parent varsa önce parent'a gider oradaki parametresiz constructor'ı çalıştırır. 282](#_Toc72246210)

[Mammal mammal = new Mammal(); 284](#_Toc72246211)

[önce parente gider parametresiz constructor arar orada birşey olmadığından işlem yapamaz döner Mammal() Constructora gider this.birth ü yazdırır. 284](#_Toc72246212)

[THİS VE SUPER KULLANIMI 284](#_Toc72246213)

[public Mammal() { 285](#_Toc72246214)

[System.out.println(this.birth); // true 285](#_Toc72246215)

[this.feed(); 285](#_Toc72246216)

[System.out.println(this.age); 285](#_Toc72246217)

[System.out.println(super.age); // 4 285](#_Toc72246218)

[System.out.println(super.name); // Karabas 285](#_Toc72246219)

[super.move(); // Hayvanlar hareket eder. 285](#_Toc72246220)

[Parantezsiz super ise sadece parent'lardaki variable ve methodlara ulaşmanızı sağlar. 285](#_Toc72246221)

[Parentteki variable veya methodlara super kullanarak ulaşmak daha güvenlidir. çünkü child ve parent'lardaki aynı isimli variable veya methodlarda problem yaşayabilirsiniz.bu nedenle genel kullanımdak this class içindekiler için, super ise parent içindekiler için kullanılır. 285](#_Toc72246222)

[29.DERS ( POLYMORPHISM) 286](#_Toc72246223)

[Polymorphism : çoklu yapı demek. 286](#_Toc72246224)

[Bir metodu çoklu yapı haline getiriyorsanız polymorphism yapıyorsunuz demektir.Bunun için Overloading ve Overriding kullanılır.Polymorphism'i Overloading ile yaparsanız "Compile Time Polymorphism" denir.Diğer adı Static Polymorphism'dir. 286](#_Toc72246225)

[Polymorphism'i Overriding ile yaparsanız "Run Time Polymorphysm" denir.Diğer adı Dinamic polymorphism'dir. 286](#_Toc72246226)

[public void eat() { 286](#_Toc72246227)

[System.out.println("Ye!"); 286](#_Toc72246228)

[} 286](#_Toc72246229)

[Parametreyi değiştirerek Overlaoding yapıp Polymorphism yaptık. 286](#_Toc72246230)

[public void eat(String name) { 286](#_Toc72246231)

[System.out.println(name+ "Ye!"); 286](#_Toc72246232)

[} 286](#_Toc72246233)

[Body'yi değiştirerek Overriding yapıp polymorphism yaptık. 286](#_Toc72246234)

[public void eat() { 286](#_Toc72246235)

[System.out.println("Lütfen Ye!"); 286](#_Toc72246236)

[} 286](#_Toc72246237)

[EXEPTİONS 287](#_Toc72246238)

[﻿Java code çalıştırılırken, farklı error’lar oluşabilir: 287](#_Toc72246239)

[1) Code’u yazan hatalı yazmış olabilir 287](#_Toc72246240)

[2) Kullanıcı hatalı data girişi yapıyor olabilir 287](#_Toc72246241)

[3) Öngörülemeyen hatalar oluşabilir 287](#_Toc72246242)

[Hata oluştuğunda Java çalısmayı durdurur ve console’a bir hata mesajı yazdırır. 287](#_Toc72246243)

[Buna exception atma manasına gelen “throw an exception” denir. 287](#_Toc72246244)

[Exception: istisnai problemler demektir.mesela sayılar birbirine bölünebilir ama sadece 0'a bölümünde hata verir. Bu problemli duruma exception diyoruz. Bazı problemler vardır, halledilebilir, handling exeption, bazılarıda halledilemez throwing exeption. yardım istemeye throw denir. 287](#_Toc72246245)

[Temelde iki türlü exception var. 287](#_Toc72246246)

[1)Exeption'ların bir kısmı Compile Time Error (Checked Exeption) 287](#_Toc72246247)

[﻿Checked Exception’lar kesinlikle code yazılırken halledilmelidirler. Yanlış code yazıldığında altı kırmızı olarak çizilir. Compile Time Error'lara Checked Exeption'da denir. 287](#_Toc72246248)

[2) Bir kısmıda Run Time Error'dur. 287](#_Toc72246249)

[Compile Time Error (Checked exeption)'a örnek: 287](#_Toc72246250)

[1) FileNotFoundExceptions Programıma bir dosyayı almak istediğimde dosya bulunamadığı zaman verilen hatadır. 287](#_Toc72246251)

[2) IOException: input output exception yazdığım programıma dışardan bir şey almak veya dışarıya bir şey yollamak istediğimde bu hatayı verir. 287](#_Toc72246252)

[Not: FileNotFoundException, IOException'ın Child'ıdıdır. 287](#_Toc72246253)

[Compile Time Error(Checked Exception)'lar nasıl halledilir? 288](#_Toc72246254)

[1) Method isminden sonra "throws" keyword kullanılır. 288](#_Toc72246255)

["throws" keywordunu kullanırsanız konsolda teknik mesajlar görürsünüz. Bu yüzden çok tercih edilmez. 288](#_Toc72246256)

[public static void main(String[] args)throws FileNotFoundException { 288](#_Toc72246257)

[Yazdığımız bir programın içerisine bir dosya eklemek istediğimizde FileInputStream Class'ını kullanıyoruz. 288](#_Toc72246258)

[FileInputStream file = new FileInputStream("C:/ogrenci.txt"); 288](#_Toc72246259)

[// e.printStackTrace(); OTOMATİK OLUŞAN BU SATIRI SİLİYORUZ. ALTTAKİNİ YAZIYORUZ 288](#_Toc72246260)

[} 288](#_Toc72246261)

[FileInputStream() parametresini boş bırakırsanız FileInputStream Class'ının parametresiz Constructor'ı olmadığından altını çizer exception verir içine "C:/ogrenci.txt" yazınca altını yine çizer. altına gidip "add throws" linkini tıklıyoruz. main methoda "throws FileNotFoundException" yazısını ekleyecektir.çalıştırınca altta bir sürü hata mesajları gösterecek. bunlar yabancı birileri için readable olmadığı için bu 1. yolu çok tercih etmiyoruz. 288](#_Toc72246262)

[2) try-catch block kullanılır. 288](#_Toc72246263)

[﻿1)“try block” exception üretme ihtimali olan code içerir. 288](#_Toc72246264)

[2) “try block”dan sonra her zaman “catch block” yazılır, “catch block” exception’i handle eder. 288](#_Toc72246265)

[3) “try block”dan sonra 1’den fazla “catch block” kullanilabilir, istenirse “finally block” da kullanilabilir. “finally block” exception oluşsa da oluşmasa da çalışır. 289](#_Toc72246266)

[﻿4) Program yazarken bazı code’ların hata üretebileceğini öngörüyorsanız bu code’u try block’un icine koyup hatayı catch block kullanarak yakalayabilirsiniz. 289](#_Toc72246267)

[﻿5) Bir try block 1’den fazla catch block içerebilir. Her catch block’da farklı bir exception yakalayabilirsiniz. 289](#_Toc72246268)

[﻿a) Eğer exception ilk catch block’da yakalanırsa sonraki catch block’lar çalışmaz. 289](#_Toc72246269)

[b) Sonraki catch block’lar kesinlikle öncekilerin parent’i olmalıdır. Aksi takdirde Compile Time Error alırsınız. 289](#_Toc72246270)

[try-catch block kullanmak daha iyidir, çünkü readable'dır. Bunu kullanırsanız konsolda teknik mesajlar yerine kendi mesajlarımızı görebiliriz. Bu yüzden try-catch kullanmak tercih edilir. 289](#_Toc72246271)

[throws : yardım iste (denemeden yardım iste) throw yapılırsa java mesajarı kendi yazar, bu çok readable olmuyor. 289](#_Toc72246272)

[try-catch blocks: (try:)dene eğer hata çıkarsa (catch:) onu yakala ve konsola yazdır.lastiği değiştirmeyi dene yapamazsan bana yaz. 289](#_Toc72246273)

[try kullanılırsa ekrana çıkacak hatayı siz yazarsınız daha anlaşılır olur. 289](#_Toc72246274)

[FileInputStream() parametresini boş bırakırsanız FileInputStream Class'ının parametresiz Constructor'ı olmadığından altını çizer exception verir içine "C:/ogrenci.txt" yazınca altını yine çizer. altına gidip "add try" linkini tıklıyoruz. 289](#_Toc72246275)

[// try { 289](#_Toc72246276)

[// FileInputStream file = new FileInputStream("C:/ogrenci.txt"); 289](#_Toc72246277)

[// } catch (FileNotFoundException e) { 289](#_Toc72246278)

[yazacaktır. 289](#_Toc72246279)

[// e.printStackTrace(); OTOMATİK OLUŞAN BU SATIRI SİLİYORUZ. ALTTAKİNİ YAZIYORUZ 289](#_Toc72246280)

[// System.out.println("Dosya bulunamadı, Allah bilir kim sildi"); 289](#_Toc72246281)

[Çalıştırınca bu yazı çıkacaktır. biz bir kod yazdığımızda muhtemel hatayı tahmin ederiz. } 289](#_Toc72246282)

[// yeni bir Class oluşturuyoruz. 290](#_Toc72246283)

[class ExceptionTryCatch{ 290](#_Toc72246284)

[public static void main(String[] args) { 290](#_Toc72246285)

[try { 290](#_Toc72246286)

[FileInputStream file = new FileInputStream ("C:/ogrenci.txt"); 290](#_Toc72246287)

[} catch (FileNotFoundException e) { 290](#_Toc72246288)

[System.out.println("Dosya'nın path'i yanlış veya dosya silinmiş olabilir"); 290](#_Toc72246289)

[// e.printStackTrace(); BU SATIRI SİLİYORUM 290](#_Toc72246290)

[class ismi ExceptionTryCatch üstüne gidip sağ tıkla run as git ==> java yı seç altta try yazanı seç ve tamam de. 290](#_Toc72246291)

[üstteki mesaj değilde "Dosya'nın path'i yanlış veya dosya silinmiş olabilir"); mesajı altta mesaj olarak paylaşılır. 290](#_Toc72246292)

[ArithmeticException Run Time Exception'dir. Matematik kurallarina uygun olmayan bir islem yapildiginda bu exception alinir. 291](#_Toc72246293)

[try'dan sonra 1'den fazla catch kullanilabilir.Ama ustteki catch alttaki catch'in child'i olmalidir. catch'leri Specific'den Genel'e dogru dizmelisiniz yoksa Compile time Error alırsınız. 291](#_Toc72246294)

[int num1 = 6; 291](#_Toc72246295)

[int num2 = 0; 291](#_Toc72246296)

[try { // try:dene 291](#_Toc72246297)

[System.out.println(num1/num2); 291](#_Toc72246298)

[}catch(ArithmeticException e) { // catch: yakala 291](#_Toc72246299)

[System.out.println("Sifir ile bolme yapilmaz"); 291](#_Toc72246300)

[}catch(Exception e) { // Exception: parametrenin data type'i e: ismi 291](#_Toc72246301)

[System.out.println("Her exception'i yakala"); 291](#_Toc72246302)

[30.DERS( EXEPTİONS ) 292](#_Toc72246303)

[COMMEN RUN TİME EXCEPTİONS 292](#_Toc72246304)

[1) ArithmeticException: 292](#_Toc72246305)

[System.out.println(12/0); 292](#_Toc72246306)

[2) ArrayIndexOutOfBoundsException : 292](#_Toc72246307)

[illegal index durumunda, olmayan index istendiğinde 292](#_Toc72246308)

[// int arr[] = {a,b,c} System.out.println{arr[12]); olmayan 12.indexi yazdır diyor. 292](#_Toc72246309)

[3) ClassCastException : 292](#_Toc72246310)

[String'i integer'a çevirme. 292](#_Toc72246311)

[// String name = "Ali"; 292](#_Toc72246312)

[// Integer number=(Integer) name; 292](#_Toc72246313)

[Run Time Exception verir. 292](#_Toc72246314)

[4) IllegalArgumentException: 292](#_Toc72246315)

[bir insanın yaşından bahsediyorsunuz -3 diyorsunuz. 292](#_Toc72246316)

[// if(yumurtasayısı<0){ 292](#_Toc72246317)

[// throw new IllegalArgumentException("yumurta sayısı negatif olmamalı") 292](#_Toc72246318)

[normalde java bunun hatalı olduğunu anlamaz biz throw new ile hata mesajı oluşturuyoruz. 292](#_Toc72246319)

[5) NullPointerException: 292](#_Toc72246320)

[String name = null; 292](#_Toc72246321)

[// public void printLength(){ 292](#_Toc72246322)

[system.out.println(name.length()); } 292](#_Toc72246323)

[String'e null atandığı veya String instance variable'a değer atanmadığı zaman length() methodu çalışmaz ve NullPointerException alırsınız. 292](#_Toc72246324)

[6) NumberFormatException: 292](#_Toc72246325)

[int num = Integer.parsenlnt("abc"); 292](#_Toc72246326)

[Bir String'i integera çevirmeye çalışırken ortaya çıkan exception'dır. 292](#_Toc72246327)

[getMessage: eException'ı yakala sadece mesajı yazdır. 294](#_Toc72246328)

[public static void main(String[] args) { 294](#_Toc72246329)

[try { 294](#_Toc72246330)

[hop(); 294](#_Toc72246331)

[}catch (Exception e) { 294](#_Toc72246332)

[System.out.println(e.getMessage()); // getMessage eException'ı yakala sadece mesajı yazdır.teknik hata mesajını yazma demektir. sadece e yazılsaydı java.lang.RuntimeException yazacaktı. 294](#_Toc72246333)

[} 294](#_Toc72246334)

[} 294](#_Toc72246335)

[private static void hop() { 294](#_Toc72246336)

[throw new RuntimeException ("cannot hop"); 294](#_Toc72246337)

[} 294](#_Toc72246338)

[e.printStackTrace(); hangi satırlarda Exception aldındığını gösterir. 295](#_Toc72246339)

[public static void main(String[] args) { 295](#_Toc72246340)

[try { 295](#_Toc72246341)

[hop(); 295](#_Toc72246342)

[} catch ( Exception e) { 295](#_Toc72246343)

[e.printStackTrace(); // bunu kullanırsak hangi satırlarda Exception aldığımızı gösterir. Exception hataları oluşum sırasına göre consolda gösterir. 295](#_Toc72246344)

[// at day30exceptions.ThrowSorusu03.hop(ThrowSorusu03.java:19) 295](#_Toc72246345)

[// at day30exceptions.ThrowSorusu03.main(ThrowSorusu03.java:8) 295](#_Toc72246346)

[} 295](#_Toc72246347)

[} 295](#_Toc72246348)

[private static void hop() { 295](#_Toc72246349)

[throw new RuntimeException ("cannot hop"); 295](#_Toc72246350)

[} 295](#_Toc72246351)

[e yazmazsak altını çizer. ArithmeticException olası aritmatik tüm hataları yakalar.istediğimiz kadar catch kullanabiliriz tek şartla yukarıdaki alttakinin child'i olmak şartıyla. 296](#_Toc72246352)

[catch'den sonra istersek finally yazabiliriz.finally{} block Exception olsa da olmasa da her zaman çalışır.finally{} block mutlaka try-catch block'dan sonra kullanılır. 296](#_Toc72246353)

[public static void main(String[] args) { 296](#_Toc72246354)

[int num1 = 12; 296](#_Toc72246355)

[int num2 = 0; 296](#_Toc72246356)

[try { 296](#_Toc72246357)

[System.out.println(num1/num2); 296](#_Toc72246358)

[}catch(ArithmeticException e) { 296](#_Toc72246359)

[System.out.println("Sıfır ile bölme yapılamaz"); 296](#_Toc72246360)

[}finally { 296](#_Toc72246361)

[System.out.println("İşleminiz bitmiştir"); 296](#_Toc72246362)

[} 296](#_Toc72246363)

[INTERVEİW SORUSU: 296](#_Toc72246364)

[final, finally, finalize keyword'larının farkı nedir? 296](#_Toc72246365)

[1) finally{} (block) ==> try-catch block'dan sonra kullanılır. Exception olsa da olmasada çalışır. 296](#_Toc72246366)

[2) final; (keyword) ==> Variable'larda final kullanılırsa, o variable'ın değeri değiştirilemez.mesela pi sayısı final yapıldığı zaman değeri değiştirilemez. 296](#_Toc72246367)

[// final String code = "POS"; 296](#_Toc72246368)

[Method'larda final kullanılırsa, o methodun{} body kısmı değiştirilemez.yani Overriding yapamam demektir. Inheritance mümkün değildir. Class'lard final kullanılırsa o Class extends edilemez.(parent olamaz) 296](#_Toc72246369)

[// public class Animal{} ==> public class Dog extends Animal{}; 296](#_Toc72246370)

[// hatalı==> public final class Animal{}==> public class Dog extends Animal{} olmaz. 297](#_Toc72246371)

[3) finalize() (method) ==> finalize() methodu, Garbage Collector'ın imha edeceği dataları imha edilecek hale getirir. çöpleri topla sonra atarım. 297](#_Toc72246372)

[Garbage Collection 297](#_Toc72246373)

[Ogrenci ogr1 = new Ogrenci(); 297](#_Toc72246374)

[Ogrenci ogr2 = new Ogrenci(); 297](#_Toc72246375)

[ogr1 = ogr2; 297](#_Toc72246376)

[Bu örneğimizde Ogrenci sınıfından iki adet nesne oluşturduk. 1. ve 2. satırlarda ilk oluşturduğumuz nesnenin adresini “ogr1” işaretçisine, ikinci oluşturduğumuz nesnenin adresini “ogr2“ işaretçisine aktardık. 3. satırda ise ikinci nesnenin adresini, ilk nesnenin adresini tutan “ogr1” işaretçisine aktardık. Yani artık ogr1 işaretçisi, ilk nesnenin adresini değil ikinci nesnenin adresini tutmaktadır. İlk oluşturulan nesnenin adresi kaybedildiği için artık o nesneye ulaşmak mümkün olmayacaktır. Nesneye erişimin mümkün olmadığı durumlarda nesnenin bellekte kalmasının bir anlamı yoktur. Bu yüzden Garbage Collection tarafından silinecektir. 297](#_Toc72246377)

[public static void main(String[] args) { 298](#_Toc72246378)

[String s = ""; 298](#_Toc72246379)

[System.out.println(s.length()); 298](#_Toc72246380)

[try { // dene bunu çalıştır her yerde kullanılabilir ama genel kullanım exception ihtimalinde kullanılıyor. 298](#_Toc72246381)

[s+="t"; 298](#_Toc72246382)

[}catch(Exception e) { 298](#_Toc72246383)

[s+="c"; 298](#_Toc72246384)

[}finally { 298](#_Toc72246385)

[s+="f"; 298](#_Toc72246386)

[} 298](#_Toc72246387)

[s+="a"; 298](#_Toc72246388)

[System.out.println(s); 298](#_Toc72246389)

[String n = null; 298](#_Toc72246390)

[String m; 298](#_Toc72246391)

[// String'e null atandığı veya String instance variable'a (o) değer atanmadığı zaman length() methodu çalışmaz ve NullPointerException alırsınız.NullPointerException Run Time Exception'dır. n bir bir objedir, obje için uzunluk ölçülemez. String m bir methodun içinde olduğu için local variable'dir mutlaka değer atanması lazım yoksa hata verir. 298](#_Toc72246392)

[// System.out.println(n.length()); 298](#_Toc72246393)

[// System.out.println(m.length()); 298](#_Toc72246394)

[// System.out.println(o.length()); 298](#_Toc72246395)

[INTERVİEW SORUSU: 299](#_Toc72246396)

[throw ==> 299](#_Toc72246397)

[java bir satıra geldiğinde normalde hata vermese bile burada hata yap diyebiliriz, bunun için throw kullanırız. mesela girilmemesi gereken arazilere mayın döşenir, aynı şekilde bir satıra girilmemesi isteniyorsa orada hata versin isteyebiliriz, bunun için throw kullanılır. 299](#_Toc72246398)

[throw , throws keywordlerinin farkı : 299](#_Toc72246399)

[1) throw, method body'si içine yazılır, throws method parentezi ile curly brace'in {} arasına yazılır. throw'u method body'si içinde istediğimiz yerde kullanabiliriz. 299](#_Toc72246400)

[2) throw'u method body'si içinde exception almak istediğimiz yerde kullanabiliriz. throws ise mecburen method isminden sonra bir kere kullanılabilir. 299](#_Toc72246401)

[3) bir throw sadece bir tane exception üretir. bir throws ile birden fazla exception üretebilirsiniz yani "throws NullPointerException, ArithmeticException" mümkün. 299](#_Toc72246402)

[4) throw için syntax ==> throw new NullPointerException; 299](#_Toc72246403)

[throws için syntax ==> throws NullPointerException 299](#_Toc72246404)

[public static void main(String[] args) { 299](#_Toc72246405)

[System.out.println(exceptions()); 299](#_Toc72246406)

[} 299](#_Toc72246407)

[@SuppressWarnings("finally") // 48,49,50. satırların altını sarı renk çiziyordu bunu yazarak kaldırdık. 299](#_Toc72246408)

[public static String exceptions() { 299](#_Toc72246409)

[String result = ""; 299](#_Toc72246410)

[String v = null; 299](#_Toc72246411)

[try { 299](#_Toc72246412)

[try { 299](#_Toc72246413)

[result += "before"; 299](#_Toc72246414)

[v.length(); // null uzunluğu istenince NullPointException hatası alınır ve alt satıra geçmeden satır 29'a geçer. 300](#_Toc72246415)

[result += "after"; 300](#_Toc72246416)

[}catch( NullPointerException e) { 300](#_Toc72246417)

[result += "catch"; 300](#_Toc72246418)

[throw new RuntimeException(); 300](#_Toc72246419)

[// burada Exception oluştursun diyor ama altta onu yakalayacak catch yok bu nedenle bir alt satıra geçer. 300](#_Toc72246420)

[}finally { 300](#_Toc72246421)

[// alttaki üç satırın altını sarı renk çiziyordu üstte @SuppressWarnings("finally") yazarak bunu kaldırdık. 300](#_Toc72246422)

[result +="finally"; 300](#_Toc72246423)

[throw new Exception(); // yeni bir Exception oluşturur ve bunu alttaki catch yakalar. 300](#_Toc72246424)

[} 300](#_Toc72246425)

[}catch(Exception e) { // Bu parent Exception dır. satır 31'u kapatırsak 300](#_Toc72246426)

[// bu sefer bu 300](#_Toc72246427)

[result += "done"; 300](#_Toc72246428)

[} 300](#_Toc72246429)

[return result; // result'u çalıştırır, return'u ekrana yazdırır. 300](#_Toc72246430)

[// method'da void kullanılmadığı için return 300](#_Toc72246431)

[} 300](#_Toc72246432)

[} 300](#_Toc72246433)

[Konsolda “beforecatchfinallydone” yazar 300](#_Toc72246434)

[Throwable 301](#_Toc72246435)

[Throwable: atılabilir demektir. bütün exception'ların ortak faaliyetlerinin bulunduğu classtır. java bazen exception bazen de error diye consola hata yazar. 301](#_Toc72246436)

[Out of Memory Error: 301](#_Toc72246437)

[Javada kullandığımız iki memory var Stack ve Heap bunlandan birisi dolarsa Java memory'ye bir şey koyamaz ve Out of Memory Error verir. hafıza dolmuş demektir 301](#_Toc72246438)

[Error'lar handle edilemez, çözülemezler. Error ile Exception arasındaki fark budur. 301](#_Toc72246439)

[Error'lar Run Time Error'dur. 301](#_Toc72246440)

[StackOverflowError : 301](#_Toc72246441)

[Stack memory'den cok fazla data çağırımı yapılırsa memory kilitlenir ve bu hatayı verir. bu nedenle kodun kısalığı Stack ve heap memorydeki yükü azaltır. 301](#_Toc72246442)

[System Crash Error: 301](#_Toc72246443)

[kullanılan sistemin çökmesinde ortaya çıkan Error'dur.Bu Errorlar kodumuzun dışında ortaya çıkan arızalardır. 301](#_Toc72246444)

[birden fazla catch görüyorsanız child parent ilişkisini soruyordur muhtemelen 301](#_Toc72246445)

[31.DERS( ABSTRACTION ) 302](#_Toc72246446)

[Abstract: soyut Concrete: somut-beton 302](#_Toc72246447)

[abstract class'lar obje üretemeyen class'lardır. Bir Abstract Class'dan obje üretilemez, Constructor'ları yoktur. 302](#_Toc72246448)

[public abstract class Animal(){} ==> Animal animal = new Animal(); 302](#_Toc72246449)

[// new Animal altını çizer hata gösterir. obje üretilemez çünkü abstract method {} body si olmayan methoddur. 302](#_Toc72246450)

[body si olmayan methodlara abstract method denir.bizim şu ana kadar bildiğimiz method lar body si olan Concrete methodlardı ikisi arasındaki fark birisinde body si var ne yapılacağı tarif ediliyor. diğerinde body si yok ne yapılacağı tarif edilmiyor. yapılacak işin ismi söylenir nasıl yapılacağından bahsedilmez. 302](#_Toc72246451)

[Abstract class concrete ve abstract child'i olur ama child'in içinden parent'daki abstract methodlar mutlaka override edilmelidir. 302](#_Toc72246452)

[override: bir methodun body si değiştirilerek yapılır. 302](#_Toc72246453)

[Child abstract ise herhangi bir override zorunluluğu yok. araba için abstract class oluşturdunuz, motorun çalışması gibi zorunlu methodlar vardır. bütün arabalar motoru çalışır olması lazım, bunun için Child Class'lar override ederek mecburen motor methodunu kullanıp bunu çalıştırması istenir yoksa child olarak kabul etmez. 302](#_Toc72246454)

[abstract olunca bu method mutlaka bulunmak zorunluluğu koyulmuş oluyor. 302](#_Toc72246455)

[concrete class ın icine abstract method konmaz,sadece concrete konur. Abstract Class'lar concrete(body'li) ve abstract(body'siz) methodlar içerebilir. 302](#_Toc72246456)

[abstarct methoda body koymaya calısırsam CompiletimeError alırım. 302](#_Toc72246457)

[concrete methoda body koymazsam CompiletimeError alırım. 302](#_Toc72246458)

[Abstract Class'ların hem concrete class hem de abstract Class olan child'leri olabilir. 302](#_Toc72246459)

[Abstract Class'ın child'i concrete class ise Abstract class'daki abstract methodalrı mutlaka override etmeli ve onlara body eklemelidir. 302](#_Toc72246460)

[// ==> bunun faydası şudur; child Class'larda mecburen olması gereken methodları bu şekilde belirlemiş oluruz. 302](#_Toc72246461)

[Abstract Class'ı child'i abstract Class ise Abstract Class'daki herhangi bir şeyi override etmek zorunda değildir. 302](#_Toc72246462)

[Java’da kalıtım(inheritence) bir sınıfın kendine ait özellikleri başka bir sınıfa aynen aktarması ya da bazı özelliklerin kullanılmasına diğer sınıflarda izin vermesidir. 302](#_Toc72246463)

[package day31abstraction; 303](#_Toc72246464)

[public abstract class Abstraction01 { 303](#_Toc72246465)

[public static void main(String[] args) { 303](#_Toc72246466)

[} 303](#_Toc72246467)

[public void concreteMethod() { 303](#_Toc72246468)

[System.out.println("Ben concrete'im"); 303](#_Toc72246469)

[} 303](#_Toc72246470)

[public abstract void abstractMethod(); 303](#_Toc72246471)

[} 303](#_Toc72246472)

[Bir methodun bir class'ta mutlak surette çalışmasını istiyorsak bu yolu izleriz. 304](#_Toc72246473)

[Abstraction01 abstract class oldugundan herhangi bir override zorunlulugu yok. Compile Time Error almiyorum. ama Abstraction02 concrete Class olduğundan ve bu Class’ı abstract bir Class olan Abstraction01’in Classı yapmak için yanına extends Abstraction01 yazarsam altını çizer ve hata verir. Abstract bir Class içinde abstract bir method var bunu inherit etmediğimizden hata veriyor. bu methodun inherit edilmesi lazım.(abstract methodların body si olmuyor, bu methoda body ekleyerek methodu concrete methoda dönüştürme) 304](#_Toc72246474)

[bunun için yazının üzerine gidip Add unimplemented methods 304](#_Toc72246475)

[tıklayınca aşağıdaki methodu body si ile birlikte kendisi oluşturuyor.dikkat ederseniz bu methodu oluşturmadan, kullanmadan bu Class'ı child olarak kullanmamıza izin vermiyor. Honda parent'i bir child class'ına motor methodu olmadan çalışmana izin vermem demek gibi birşey. 304](#_Toc72246476)

[public class Abstraction02 extends Abstraction01{ 304](#_Toc72246477)

[public static void main(String[] args) { 304](#_Toc72246478)

[Abstraction02 obj = new Abstraction02(); 304](#_Toc72246479)

[obj.abstractMethod(); 304](#_Toc72246480)

[obj.concreteMethod(); 304](#_Toc72246481)

[} 304](#_Toc72246482)

[@Override 304](#_Toc72246483)

[public void abstractMethod() { 304](#_Toc72246484)

[} 304](#_Toc72246485)

[} 304](#_Toc72246486)

[32.DERS ( ABSTRACTİON ) 306](#_Toc72246487)

[bir Class'ı final yaptığımız zaman daha onun child'i olamaz. 306](#_Toc72246488)

[Bizim abstract Class oluşturmamızın sebebi içine methodlar koyalım o metodlarıda child Class'lar kullansın amaçlı. bu nedenle abstract Class'lar final olamazlar. aynı şekilde abstract methodlar final edilemezler. abstract method private da olamaz, private methodlar override edilemezler. 306](#_Toc72246489)

[﻿Abstract Class Kurallari: 307](#_Toc72246490)

[1. Bir Abstract Class’dan object oluşturulamaz. 307](#_Toc72246491)

[2. Abstract class’lar abstract ve concrete method’lar icerebilir. 307](#_Toc72246492)

[3. Abstract class private ve final olamazlar. 307](#_Toc72246493)

[4. Bir abstract class başka bir another abstract class’a extends ettiginde abstract 307](#_Toc72246494)

[method’lari override etmek zorunda degildir. 307](#_Toc72246495)

[5. Abstract class’a extends eden ilk concrete class abstract class’daki tum abstract 307](#_Toc72246496)

[method’lari override etmek zorundadir. 307](#_Toc72246497)

[Abstract Method Kurallari: 307](#_Toc72246498)

[1. Abstract method’lar sadece abstract classlar icinde olusturulabilir. 307](#_Toc72246499)

[2. Abstract methodlar private ve final olamazlar. 307](#_Toc72246500)

[3. Abstract method’larin method body’si olamaz. 307](#_Toc72246501)

[INTERFACE 310](#_Toc72246502)

[Interface Class'a benzer ama Class değildir. Abstract Classların içine hem abstract hem constract method koyabiliyorduk, Interface'e sadece abstract method konulabilir. 310](#_Toc72246503)

[INTERFACE METHOD 310](#_Toc72246504)

[interface'de method oluştururken kesinlikle body yazmayın interface'deki methodlar kesinlikle public ve abstract olmalıdır.interface'deki abstract methodları üretirken abstract keyword kullansanızda olur kullanmasanızda olur. 310](#_Toc72246505)

[çünkü interface kendi içine yazılan methodları otomatik olarak abstract olarak kabul eder. 310](#_Toc72246506)

[public void add(); veya public abstract void add(); 310](#_Toc72246507)

[INTERFACE VARIABLE 310](#_Toc72246508)

[Interface içerisine variable koyabaliriz ama bu 310](#_Toc72246509)

[variable'lar mutlaka public static ve final olmalıdır. bu kuralı dinlemez variable'ı private veya protected yaparsanız Compile Time Error alınır. 310](#_Toc72246510)

[interface içindeki variable'ları mutlaka initialize etmek(değer atamak) zorundasınız, aksi takdire Compile Time Error alırsınız. 310](#_Toc72246511)

[int a=12; gibi yapılmalı . interface'de variable'lar initialize, public, static, final olmalı ve mutlaka initialize edilmelidir. initialize : değer atanmalıdır. Variable oluştururken public,static,final keywordları kullanılsada olur kullanılmasada olur. kullanılmasada kendisi default olarak public, static, final olarak atar. 310](#_Toc72246512)

[package day32abstraction; 310](#_Toc72246513)

[public interface Interface01 { 310](#_Toc72246514)

[public void add(); 310](#_Toc72246515)

[public abstract void subtract(); 310](#_Toc72246516)

[public static final int a = 12; 310](#_Toc72246517)

[int b = 11; 310](#_Toc72246518)

[public int c = 13; 310](#_Toc72246519)

[final int d = 14; 310](#_Toc72246520)

[static int e = 15; 310](#_Toc72246521)

[} 310](#_Toc72246522)

[33.DERS ( INTERFACE ) 311](#_Toc72246523)

[Interface: arayüz demektir. 311](#_Toc72246524)

[Interface Access modifier sadece public ve default olabilir. 311](#_Toc72246525)

[multipleInheritence mümkün değil yani Java'da bir child'in birden fazla parenti olamaz ama bazen bir child'in birden fazla parenti olması gerekebilir bu sorun Java'da Interface ile çözülmüş. bir çok parent'e ihtiyac duyduğumuzda Interface Animal, Interface Mammal yapıyoruz.böylelikle concrete dog child extends olarak Animal ve Mammal'e ulaşabiliyor. 311](#_Toc72246526)

[body {} si olan methodlara concrete method, body'si olmayan methodlara abstract method denir. 311](#_Toc72246527)

[Interface kullanımının diğer bir sebebi yapılması gereken zorunlu methodlar Interface içerisine yazılır, Interface içerisine isteğe bağlı methodları koyamıyoruz, mecburi olan methodlar yazılır. 311](#_Toc72246528)

[bir child'i birden fazla Inheritence bağlamak için implements yazıp Inheritence isimleri virgül ekleyerek yazıyoruz. 311](#_Toc72246529)

[public class LuxHonda extends Araba implements IcAraba, DisAraba { 311](#_Toc72246530)

[Class ==> Interface : implements Class'ları Interface'e bağlamak için implements 311](#_Toc72246531)

[Class ==> Class : extends Class'ları Class'a bağlamak için extends 311](#_Toc72246532)

[Interface ==> Interface : extends Interface'leri Interface'lere bağlamak için extends 311](#_Toc72246533)

[Farklı Interface'lerde aynı isimde farklı Return type'lerdeki metodlar yazılırsa Compile Time Error verir. 311](#_Toc72246534)

[Farklı Interface'lerde aynı isim ve aynı Return type'lerdeki metodlar olursa implements durumunda sorun olmaz. 311](#_Toc72246535)

[Farklı Interface'lerde Variable'lar için data type ve isimler aynı ama değerler farklı ise implements durumunda bu variable'lara Interface isimleri ile ulaşabiliriz. 311](#_Toc72246536)

[Bir variable final ise java onu mavi olarak yazar, onu değişteremeyeceğimi anlarım. 311](#_Toc72246537)

[Normalde Inheritence içinde sadece abstracat method olabilir demiştik ancak java 8 den sonra bir özellik eklenmiş default ve static ekleyerek {} body li concrete method üretebilme imkanı tanımıştır. 311](#_Toc72246538)

[public default void döseme() { 311](#_Toc72246539)

[System.out.println("Interface'de default keyword ile döseme yaptım"); 311](#_Toc72246540)

[interface interface'e child olacaksa extends kullanılır 311](#_Toc72246541)

[interface'lerin Access modifier'i public ve default olabilir, protected ve private olamaz. 312](#_Toc72246542)

[public default void döseme() { // obje kullanarak child'de bu methodu çağırabilirim. 314](#_Toc72246543)

[System.*out*.println("Interface'de default keyword ile döseme yaptım"); 314](#_Toc72246544)

[} 314](#_Toc72246545)

[public static void isitma() { // direk Class ismi ile herhangi bir Class'tan bu methodu çağırabilirim. 314](#_Toc72246546)

[System.*out*.println("Static ile yan ayna isitma özelliği koyduk"); 314](#_Toc72246547)

[} 314](#_Toc72246548)

[System.*out*.println("Ic araba Interface'ınden"+IcAraba.*price*); // iki ayrı interfacede price isminde variable olduğundan interface ismi ile variable’ı çağırdık. 316](#_Toc72246549)

[System.*out*.println("Dış araba Interface'ınden"+ DisAraba.*price*);// 3000 316](#_Toc72246550)

[System.*out*.println(*old*); // iki tane old isimli variable olmadığından interface adını kullanmadan variable’ı çağırdık. 316](#_Toc72246551)

[luxHonda.döseme(); // Inheritence'daki bodyli default methodu obje oluşturarak çağırdık 316](#_Toc72246552)

[IcAraba.*isitma*(); // Inheritence'daki bodyli static methodu obje oluşturmadan class ismi ile çağırdık 316](#_Toc72246553)

[34.DERS ( COLLECTIONS(Interface) ) 318](#_Toc72246554)

[Collections: yığın, birikim demektir. 318](#_Toc72246555)

[Javada temelde üç tane list var. List, Queue, Set. 318](#_Toc72246556)

[Collection bu üçünün parent'ı durumunda. Java her zaman ortak özellikleri parent'a koyar. 318](#_Toc72246557)

[Iterable Collection Interface'ının parentıdır. değiştirilebilir demektir ve bunların hepsi Interface'dır. 318](#_Toc72246558)

[Child Interface ise parent Class olamaz. 318](#_Toc72246559)

[Queue sıra demek. 318](#_Toc72246560)

[Iterator bir collection'ın elemanları arasında gezer ve o elemanları değiştirir. for each loop Collection elemanları arasında gezer ama değiştiremez. 318](#_Toc72246561)

[Collection olarak biz sadece List'i öğrenmiştik 318](#_Toc72246562)

[List<String> list1 = new ArrayList<>(); yazdığımızda hem List hem de ArryList'in altını cizer 318](#_Toc72246563)

[List ayrı bir Interface ArrayList ayrı bir Class, Iterator Javada bir Interface'dir. 318](#_Toc72246564)

[.iterator methodu başka bir objeyi interface'e uygun hale getirmektir. 318](#_Toc72246565)

[interface'de normalde Constructor olmaz, obje oluşturulamaz ama iterator methodu sayesinde başka yerde oluşturulmuş objeyi interface'de kullanılabilir haline getiriyor. 318](#_Toc72246566)

[.hasNext methodu bir sonraki elaman var mı yok mu, kontrol eder, eleman varsa true atar. 318](#_Toc72246567)

[System.out.print(iterator.next());// yazdırmak için next methodunu kullanmamız gerekiyor. 318](#_Toc72246568)

[.next methodu iterator'a git pointer’i bir sonraki elemanın öncesine götür demektir. 318](#_Toc72246569)

[Eğer siz bir Iterator'da değişimin kalıcı olmasını istiyorsanız Iterator kullanmak durumundasınız, for-each loop elamanlar arasında gezer ama Collection'ı değiştiremez. iterator ise elemanlar arasında gezer ve istenirse Collection'ı değiştirir. 318](#_Toc72246570)

[Data type ya kendisi gibi yada parenti gibi olmalı, obje oluşturabilmesi içinde class olmalı 318](#_Toc72246571)

[SortedSet obj2 = new TreeSet(); obj2 objesi TreeSet Class'ında oluşturuluyor, SortedSet onun parenti 318](#_Toc72246572)

[public static void main(String[] args) { 319](#_Toc72246573)

[List<String> list1 = new ArrayList<>(); 319](#_Toc72246574)

[list1.add("X"); 319](#_Toc72246575)

[list1.add("Y"); 319](#_Toc72246576)

[list1.add("Z"); 319](#_Toc72246577)

[System.out.println(list1);// [X, Y, Z] 319](#_Toc72246578)

[Iterator<String> iterator = list1.iterator(); 319](#_Toc72246579)

[// .iterator methodu başka bir objeyi interface'e uygun hale getirmektir. 319](#_Toc72246580)

[// Tipi Iterator interface olan iterator isimli bir obje oluşturduk. 319](#_Toc72246581)

[35.DERS ( COLLECTİONS ) 321](#_Toc72246582)

[LISTITERATOR 321](#_Toc72246583)

[ListIterator Iterator'a göre daha zengin daha çok method içeriyor. 321](#_Toc72246584)

[iterator'larda elemanlar arasında gezmek için her zaman While'ı kullanıyoruz. 321](#_Toc72246585)

[.next()bir sonraki elemana geç return et demektir. 321](#_Toc72246586)

[.hasPrevious pointer'ı en sondan başlayıp geriye doğru bir önceki elemana bak varsa true ver demektir. 321](#_Toc72246587)

[öncelikle .hasNext'i kullanıp pointi sona getiriyoruz.en sona gelen point .hasPrevious ile kaldığı yerden devam edip geriye doğru gider. .hasNext kullanmadan .hasPrevious kullanırsak en baştan başlar geride hiçbirşey olmadığından devam etmez. 321](#_Toc72246588)

[.hasPrevious() ve previous() methodlarını kullanarak bir list'in elamanlarını tersten yazmak isterseniz önce hasNext() ve next() kullanmalısınız. 321](#_Toc72246589)

[list oluşturmadan ListIterator oluşturup ona eleman ekleyemiyorum. 321](#_Toc72246590)

[List<String> list = new ArrayList<>(); 321](#_Toc72246591)

[list.add("A"); 321](#_Toc72246592)

[list iterator oluşturma 321](#_Toc72246593)

[ListIterator<String> listIterator = list.listIterator(); 322](#_Toc72246594)

[while(listIterator.hasNext()) { 322](#_Toc72246595)

[Object element = listIterator.next(); 322](#_Toc72246596)

[// data type herşeye uygun olsun diye object yaptık. 322](#_Toc72246597)

[System.out.print(element + " "); 322](#_Toc72246598)

[} 322](#_Toc72246599)

[while(listIterator.hasPrevious()) { 322](#_Toc72246600)

[Object element = listIterator.previous(); 322](#_Toc72246601)

[System.out.print(element + " "); 322](#_Toc72246602)

[} 322](#_Toc72246603)

[set methodu bu indextekinin yerine bunu ata,uygula demektir. 323](#_Toc72246604)

[while(listIterator.hasNext()) { 323](#_Toc72246605)

[Object harf = listIterator.next(); 323](#_Toc72246606)

[listIterator.set(harf + "W"); 323](#_Toc72246607)

[} 323](#_Toc72246608)

[LINKEDLIST(Class) 324](#_Toc72246609)

[LinkedList bir Class'tır. LinkedList'te koyduğumuz her elemanın sağ tarafında bir pointer varmış gibi düşünüyoruz bu pointer onu sağ tarafındaki elemana bağlar.linked zaten bağlanmak demektir.bu şekilde tren vagonları gibi her eleman bir sonraki elemana bağlıdır. 324](#_Toc72246610)

[Çok fazla eleman silmemiz veya eleman eklememiz gereken durumlarda LinkedList kullanılır. Eleman ekleme ve çıkarma durumlarında LinkedList başarılıdır. 324](#_Toc72246611)

[LinkedList'te head hariç her eleman için bir data bir de pointer var.Head ilk eleman, head'de sadece pointer vardır. 324](#_Toc72246612)

[Her eleman pointer'i bir sonraki elemanı gösterir, son elemanın pointer'ı(tail) pointer'i null(boş) gösterir. . A . B . C . . Z . null ilk pointe head diyoruz,A'yı gösterir. son noktaya tail diyoruz null gösterir. 324](#_Toc72246613)

[Data ve pointer içeren her elemanın diğer adı "node" dur. " . A " birlikte bir node'dur. 324](#_Toc72246614)

[public static void main(String[] args) { 324](#_Toc72246615)

[LinkedList<String> linklist = new LinkedList<>(); 324](#_Toc72246616)

[linklist.add("Mark"); 324](#_Toc72246617)

[linklist.add("Amanda"); 324](#_Toc72246618)

[linklist.add("John"); 324](#_Toc72246619)

[linklist.add("Ann"); 324](#_Toc72246620)

[linklist.add("Pamela"); 324](#_Toc72246621)

[System.out.println(linklist); // [Mark, Amanda, John, Ann, Pamela] 324](#_Toc72246622)

[linklist.remove(2); 324](#_Toc72246623)

[linklist.removeFirst(); 324](#_Toc72246624)

[linklist.removeLast(); 325](#_Toc72246625)

[linklist.add(1, "Ali"); 325](#_Toc72246626)

[linklist.addFirst("Kemal"); 325](#_Toc72246627)

[linklist.addLast("Zeynep"); 325](#_Toc72246628)

[linklist.set(1, "Ajanda"); 325](#_Toc72246629)

[linklist2.addAll(linklist); // iki listi birbirine ekliyoruz 325](#_Toc72246630)

[linklist.addAll(3, linklist2); 325](#_Toc72246631)

[System.*out*.println(linklist2.contains("Can"));// True veya False return eder. 325](#_Toc72246632)

[linklist2.clear(); // clear methodu listin elemanlarını siler, sizde boş bir list kalır. 325](#_Toc72246633)

[SET(Interface) 326](#_Toc72246634)

[Set interface'i duplication'a(tekrarlı elemanlara) müsaade etmez. Set içinde aynı elemandan iki tane olamaz. 326](#_Toc72246635)

[Set'i tekrarlı eleman bulunmasını istemediğimiz zaman kullanırız. 326](#_Toc72246636)

[HashSet, LinkedHashSet ve TreeSet Class'ları Set interface'inin childleridir ve Set'te tekrarlı elemanlar kullanılamadığından dolayı bu Class'lardada hiç bir surette bu Class'larda tekrarlı eleman olmaz. Set gördüğümüzde no dublication diye düşünübeliriz. 326](#_Toc72246637)

[TreeSet aynı zamanda SortedSet interface'inin child'didir. 326](#_Toc72246638)

[SortedSet interface'inin özelliği elemanlarının sıralanmış olmasıdır. 326](#_Toc72246639)

[TreeSet bu interface'in child'i olduğundan inheritence'dan(miras) dolayı aynı özellik TreeSet Class'ında da mevcuttur. 326](#_Toc72246640)

[Hem tekrarlı olmasın, hemde sıralı olsun istiyorsak TreeSet Class'ını kullanırız. 326](#_Toc72246641)

[3 tane bilmemiz gereken Set var. 326](#_Toc72246642)

[1) HaschSet(No dublication) 326](#_Toc72246643)

[2) LinkedHashSet(No dublication) 326](#_Toc72246644)

[3) TreeSet(No dublication+Sorted) 326](#_Toc72246645)

[36.DERS ( COLLECTIONS ) 327](#_Toc72246646)

[HashSet(Class) 327](#_Toc72246647)

[1) Tekrarlı eleman kullanımına(Duplication) izin vermez. 327](#_Toc72246648)

[2) Key - Value yapısını kullanır. Key'ler ve Value'lar HashSet için unique'dir(benzersiz). normalde Key'ler her zaman unique'dir. Value'lar kullanılan Class'a bağlı olarak unique’de olabilir, olmayadabilir. 327](#_Toc72246649)

[Kullanılan Class HashSet ise Value'larda unique olur ama ArrayList ise unique olma zorunluluğu yoktur. 327](#_Toc72246650)

[Key'i java kendi üretir, tekrarı bir benzeri yoktur, aynı şekilde vaule'larında benzeri olmamalı. Key-Value'lara en güzel örnek sözlüklerdir. Bir kelimenin manası bir kere yazılır başka yazılmaz. kelimeler Key'dir, Value'ler ise tekrar kullanılabilir. 327](#_Toc72246651)

[yürek = kalp, gönül = kalp gibi... bu nedenle mesela dijital bir sözlük oluşturduğunuz da ArrayList makul olur HashSet makul olmaz yürek için kalp yazdığınızda gönül için kalp yazamazsınız yani unique'dır. 327](#_Toc72246652)

[HashSet'i ekrana yazdırdığımızda key ve value'yu birlikte görürüz. 327](#_Toc72246653)

[[Dil=kalp, Yürek=Kalp, Gönül=Kalp, Yüz=Sayı] 327](#_Toc72246654)

[mesela öğrenci numarasını girdiğinizde bu keye uygun sadece ilgili kişiyi bulur. 327](#_Toc72246655)

[Bizim her yazdığımız karaktere Java bir reference değeri atar, buna Hashcode diyoruz. 327](#_Toc72246656)

[uniqe numara: herkesin essiz bir numarasi var. mesela universite numarasi 9017004 90(yil)17(bolum)004(kazanma sirasi)console de cikan referanslar javanin hashing yontemi buna hashcode denir. 327](#_Toc72246657)

[Java bizim verdiğimiz her value için hashing tekniğini kullanarak bir unique oluşturur.key her zaman unique dir. 327](#_Toc72246658)

[Normalde List'lerde add() methodu hep yeni eklenen elemanı sona ekler ama HashSet'lerde öyle bir zorunluluk yok.HashSet'ler elemanları ekrana rastgele yazdırır, kural yoktur. 327](#_Toc72246659)

[Java arka planda her vaule'ya verdiği key unique değerine göre bir reference atar ve ona göre bir sıraya dizer. buda random rastgele olur. 327](#_Toc72246660)

[HashSet'te bir Value'ya aynı değer eklenmek istendiğinde Compile Time veya Run Time Error alınmaz. son yüklediğimizi eskisinin üzerine yazar, tekrara izin verilmez. 327](#_Toc72246661)

[Class Data type constructor ismi 328](#_Toc72246662)

[HashSet<String> hSet = new HashSet<>(); 328](#_Toc72246663)

[hSet.add("Apple"); 328](#_Toc72246664)

[.hashCode() methodu bir valuye Java'nın atamış olduğu hashCode değerini çalıştırır. 328](#_Toc72246665)

[System.out.println(hSet.hashCode()); 328](#_Toc72246666)

[LinkedHashSet(Class) 329](#_Toc72246667)

[1) duplication'a izin vermez, key-value yapısı var. 329](#_Toc72246668)

[2) elemanları programcının girdiği sıraya göre dizer. 329](#_Toc72246669)

[HashSet hepsinden hızlıdır. sıralama ile hiç uğraşmaz. 329](#_Toc72246670)

[LinkedHashSet'te sizin yazdığınız sıraya göre dizmeye çalışır. 329](#_Toc72246671)

[LinkedHashSet<String> lhSet = new LinkedHashSet<>(); 329](#_Toc72246672)

[lhSet.add("Z"); 329](#_Toc72246673)

[lhSet.add("A"); 329](#_Toc72246674)

[System.out.println(lhSet); // sıralama olmadan bizim girdiğimiz sıraya göre yazar. 329](#_Toc72246675)

[LinkedHashSet'i TreeSet'e çevirme: 329](#_Toc72246676)

[oluşturduğumuz HashSet'i TreeSet' constructor'ına parametre olarak koyup TreeSet'e çeviririz. böylelikle TreeSet'in elemanları natural order(sıralama) yapma özelliğinden istifade etmiş oluruz. 329](#_Toc72246677)

[TreeSet<Integer> tSet = new TreeSet<>(lhSet1); 329](#_Toc72246678)

[System.out.println(tSet); 329](#_Toc72246679)

[TreeSet(Class) 330](#_Toc72246680)

[1) HashSet ile hemen hemen aynıdır. Duplication'a müsaade etmez 330](#_Toc72246681)

[2) TreeSet elemanlarını natural order( sayı ise küçükten büyüğe, String ise a dan z ye) göre dizer. 330](#_Toc72246682)

[3) HashSet ile farkı: TreeSet elemanlarını natural ordera göre dizer. HashSet ise rastgele elemanları dizer. 330](#_Toc72246683)

[4) HashSet ile farkı: HashSet bu sıralama meselesinden dolayı daha hızlıdır. 330](#_Toc72246684)

[5) Natural Order'a sahip bir Set lazım olduğunda normalde TreeSet kullanılması lazımken yavaşlığından dolayı HashSet oluşturulur sonra bu HashSet TreeSet'e dönüştürülür böylelikle TreeSet'in hız problemi çözülmüş olur.Direk TreeSet oluşturma genellikle kullanılmaz. 330](#_Toc72246685)

[Sıralama benim için önemli olduğunda TreeSet kullanırım. 330](#_Toc72246686)

[HashSet oluşturup Java'nın elemanların hızlı eklemesini sağlarız. 330](#_Toc72246687)

[TreeSet<String> tSet = new TreeSet<>(); 330](#_Toc72246688)

[tSet.add("ABC"); 330](#_Toc72246689)

[System.out.println(tSet); // alfabetik sırada yazdıracaktır. 330](#_Toc72246690)

[HashSet oluşturup Java'nın elemanların hızlı eklemesini sağlarız. 330](#_Toc72246691)

[HashSet<String> hSet1 =new HashSet<String>(); 330](#_Toc72246692)

[hSet1.add("ABC"); 330](#_Toc72246693)

[System.out.println(hSet1);// sıralama olmadan rastgele yazar. 331](#_Toc72246694)

[HashSet'i TreeSet'e çevirme: 331](#_Toc72246695)

[oluşturduğumuz HashSet'i TreeSet' constructor'ına parametre olarak koyup TreeSet'e çeviririz. böylelikle TreeSet'in elemanları natural order(sıralama) yapma özelliğinden istifade etmiş oluruz. 331](#_Toc72246696)

[TreeSet<String> tSet1 = new TreeSet<>(hSet1); 331](#_Toc72246697)

[System.out.println(tSet1); 331](#_Toc72246698)

[Queue(Interface) 332](#_Toc72246699)

[Queu'ya eklenen elemanlar listlerde olduğu gibi en sona eklenir. Queue'da silinen elemanlar en baştaki elemandan başlayarak silinir. 332](#_Toc72246700)

[[1,2,3] ==> bu Queue'dan eleman silmeye başladığınızda önce 1 silinir. FIFO ==> meşhur tabirdir. first in first out. yemekhanelerde ilk gelen ekmeği önce bitirirler. 332](#_Toc72246701)

[Queue bir interface'dir onun altında iki Class vardır. Birisi LinkedList, diğeri PriorityQueue dir. 332](#_Toc72246702)

[Elemanlar natural order dizilsin istersek PriorityQueue Constructor'ını kullanırız yok eğer elemanlar girdiğimiz sıraya göre dizilsin istersek LinkedList Constructor'ını kullanırız. 332](#_Toc72246703)

[.element() methodu ilk elemanı döndürür. Queue'da ilk eleman çok önemlidir. 332](#_Toc72246704)

[remove() methodunda index yoktur ilk elemanı siler veya remove(object) kullanılırsa parametre içine yazılan elemanı siler. 332](#_Toc72246705)

[.poll() methodu ilk elemanı siler. 332](#_Toc72246706)

[remove() methodu boş Queue'lar için kulllanılırsa Exception verir. poll() methodu boş Queue'ler için kullanılırsa null verir. Dolu Queue'lar için ikiside aynı işi yapar ilk elemanı silerler farkları yoktur. 332](#_Toc72246707)

[PriorityQueue Constructor'ı elemanları natural order'a göre sıralar. 332](#_Toc72246708)

[Queue<String> q = new PriorityQueue<>(); 332](#_Toc72246709)

[q.add("B"); 332](#_Toc72246710)

[q.add("A"); 332](#_Toc72246711)

[q.add("C"); 332](#_Toc72246712)

[System.out.println(q); 332](#_Toc72246713)

[Constructor olarak LinkedList kullanırsak girdiğimiz sıraya göre dizer. 332](#_Toc72246714)

[Queue<String> qll = new LinkedList<>(); 332](#_Toc72246715)

[qll.add("B"); 332](#_Toc72246716)

[qll.add("A"); 333](#_Toc72246717)

[qll.add("C"); 333](#_Toc72246718)

[System.out.println(qll); 333](#_Toc72246719)

[remove methodu kullanımı 333](#_Toc72246720)

[System.out.println("Silinen eleman: "+ q.remove()); 333](#_Toc72246721)

[// bu yazımda remove methodu sildiği elemanı return eder ve her zaman ilk elemanı siler. 333](#_Toc72246722)

[System.out.println("Kalan q: "+ q); // geriye kalan elemanları görürürüz. 333](#_Toc72246723)

[System.out.println(q.remove("C")); 333](#_Toc72246724)

[// true verir, çünkü biz Java'ya hangi eleman silinmesi gerektiğini söyledik, Java true return ederek işlem tamam mesajı verir. 333](#_Toc72246725)

[System.out.println(q); // [B] 333](#_Toc72246726)

[System.out.println("Silinen eleman: "+ q.remove()); // B'yi silecektir. 333](#_Toc72246727)

[System.out.println(q); // [] 333](#_Toc72246728)

[// System.out.println("Silinen eleman: "+ q.remove()); 333](#_Toc72246729)

[// remove methodu() boş Queue'lar için Exception verdiğinden bu kodu comment yaptık. 333](#_Toc72246730)

[.element() methodu ilk elemanı döndürür. Queue'da ilk eleman çok önemlidir. 333](#_Toc72246731)

[System.out.println(qll.element()); 333](#_Toc72246732)

[.poll methodu kullanımı 333](#_Toc72246733)

[System.out.println(qll.poll()); // B ==> ilk elemanı sildi ve return etti. 333](#_Toc72246734)

[System.out.println(qll);// [A,C] 333](#_Toc72246735)

[System.out.println(qll.poll());// A 333](#_Toc72246736)

[System.out.println(qll);// [C] 333](#_Toc72246737)

[System.out.println(qll.poll());// C 333](#_Toc72246738)

[System.out.println(qll);// [] 333](#_Toc72246739)

[System.out.println("Boş Queue için poll(): "+ qll.poll()); // null 333](#_Toc72246740)

[37.DERS ( COLLECTIONS ) 334](#_Toc72246741)

[Deque(Interface) 334](#_Toc72246742)

[Açılımı Double ended Queue (iki uçlu Queue) 334](#_Toc72246743)

[Queue'larda ekleme sona yapılır, silme bastan yapılır. Queue kullanırsanız eklemeyi hem başa hem de sona yapabilirsiniz. 334](#_Toc72246744)

[Aynı şekilde silmeyi hem baştan, hem de sondan yapabilirsiniz. Dequeue'da FİFO(First In First Out) var, bunun yanında LIFO'da var.(Last In First Out) 334](#_Toc72246745)

[Queue'lara eleman olarak null eklenebilir ama Deque'lara null eleman olarak eklenemez. 334](#_Toc72246746)

[Deque resizable'dır. Yani kapasitesi değiştirilebilir. 334](#_Toc72246747)

[﻿Deque LinkedList’den hizlidir. 334](#_Toc72246748)

[.pop() methodu Deque'larda baştaki elemanı siler ve baştaki elemanı return eder. 334](#_Toc72246749)

[.offer() methodu en sona elemena eklemek için kullanılır. 334](#_Toc72246750)

[sona ekleyince true döner. addLast() ekleyemeyince exception verir..offerFirst() methodu başa ekleyince true döner, addFirst() ekleyemeyince exception verir. 334](#_Toc72246751)

[.push() methodu en başa ekleme yapar. push() başa ekleyemeyince false döner. 334](#_Toc72246752)

[MAPS 336](#_Toc72246753)

[Map bir interface'dir. Map interface'inin üç tane child Class'ı vardır. 1) HashMap 2) HashTable 3) TreeMap 336](#_Toc72246754)

[HashMap 336](#_Toc72246755)

[HashMap key-value yapısını kullanır.Key ve Value programcı tarafından yazılır.Key ve Value'da null değeri kullanılabilir. Key'de 1'den fazla null değeri kullanılırsa Java son kullanılan null'u kabul eder, value'da birden fazla null değeri kullanılabilir.HashMap console'a yazdırıldığında hem key değerleri hem de value değerleri aralarına = sembolü konularak yazdırılır. HashMap'ler ekrana yazdırırken rastgele yazdırır. HashMap, Map'ler arasında en hızlısıdır. 336](#_Toc72246756)

[Normalde Java kendisi Hashing tekniği kullanarak kendisi key atar. Maps'lerde bu keyleri biz kendimiz atıyoruz. 336](#_Toc72246757)

[HashMap'ler "therad safe" değildir. Bazen bir application'da aynı anda iki app birlikte çalışır. bir application buna müsaitse buna "thread safe" deniyor, yani aynı anda başka şeyler yapabiliyor. 336](#_Toc72246758)

[List<String> isimler ahmet, mehmet nazmi 337](#_Toc72246759)

[List<Integer>boy 180,200,190 yazacağımıza 337](#_Toc72246760)

[hashMap'le ahmet 180, mehmet 200, nazmi 190 olarak çalışabiliyoruz. 337](#_Toc72246761)

[.put() Methodu 337](#_Toc72246762)

[Bir HashMap'e eleman eklemek istediğimizde put() methodu kullanırız, burada iki parametre kullanırız key ve Value. key'de sayı yerine null koyabiliriz ama sadece bir kere çünkü keylerde bunu dublication sayar, Exception vermez, eskisinin üzerine yazar. 337](#_Toc72246763)

[HashMap<Integer,String> hashMap = new HashMap<Integer, String>(); 337](#_Toc72246764)

[hashMap.put(3333, "Ali"); 337](#_Toc72246765)

[hashMap.put(1, "Veli"); 337](#_Toc72246766)

[hashMap.put(55, "Mine"); 337](#_Toc72246767)

[hashMap.put(4, "Kemal"); 337](#_Toc72246768)

[System.out.println(hashMap); // {1=Veli, 4=Kemal, 3333=Ali, 55=Mine} 337](#_Toc72246769)

[.remove() Methodu 337](#_Toc72246770)

[System.out.println(hashMap.remove(4)); // Kemal 337](#_Toc72246771)

[.get() Methodu 337](#_Toc72246772)

[{ } ve key numarası olmadan value okumak için get kullanıyoruz. bu key numarasında value yoksa null yazar. 337](#_Toc72246773)

[System.out.println(hashMap.get(55)); // Mine 338](#_Toc72246774)

[.keySet() Methodu 338](#_Toc72246775)

[Map'teki key numalarını Set halinde görmek için .keySet() methodunu kullanıyoruz. 338](#_Toc72246776)

[System.out.println(hashMap.keySet()); // [3333, 55] 338](#_Toc72246777)

[.values() Methodu 338](#_Toc72246778)

[Map'teki vauleleri Set halinde yazdırmak için .values() methodunu kullanıyoruz. 338](#_Toc72246779)

[System.out.println(hashMap.values()); // [Ali, Mine] 338](#_Toc72246780)

[.size() Methodu 338](#_Toc72246781)

[Map'te kaç eleman var bunu yazdırmak için .size() methodu kullanılır. 338](#_Toc72246782)

[System.out.println(hashMap.size()); // 2 338](#_Toc72246783)

[containsKey() Methodu 338](#_Toc72246784)

[Map içerisinde aradığımız key numarasında value var mı diye bakar, true veya false verir. 338](#_Toc72246785)

[System.out.println(hashMap.containsKey(3333)); // true 338](#_Toc72246786)

[.containsValue() Methodu 338](#_Toc72246787)

[Map içerisinde aradığımız value varmı diye arattırmak için .containsValue() methodu kullanılır, True veya false döner. 338](#_Toc72246788)

[System.out.println(hashMap.containsValue("Mine")); // true 338](#_Toc72246789)

[.replace() Methodu 338](#_Toc72246790)

[key numarasındaki value yerine yeni value yazar. 338](#_Toc72246791)

[hashMap.replace(55, "Emin"); 338](#_Toc72246792)

[.clear() Methodu 338](#_Toc72246793)

[Tüm elemanları siler. void bir method olduğu için console'a yazar. 338](#_Toc72246794)

[hashMap.clear(); // {3333=Ali, 55=Emin} 338](#_Toc72246795)

[HashTable 339](#_Toc72246796)

[HashMap ile hemen hemen aynıdır.Farkları: 339](#_Toc72246797)

[1) Hashtable console'a yazdırıldığında HashMap gibi rastgele sıralamada yazdırılır. 339](#_Toc72246798)

[2) Hashtable Key ve Value'larda null'a müsade etmez. 339](#_Toc72246799)

[3) Hashtable thread safe'dir.Bazen bir application'da aynı anda iki app birlikte çalışır.bir application buna müsaitse buna "thread safe" deniyor, yani aynı anda başka şeyler yapabiliyor. 339](#_Toc72246800)

[4) HashTable daha yavaştır. 339](#_Toc72246801)

[HashMap'te kullanılan tüm Methodların tamamı Hashtable'da da kullanılabilir. 339](#_Toc72246802)

[Hashtable<String, String> hTable = new Hashtable<>(); 339](#_Toc72246803)

[hTable.put("Dil","Kalp"); 339](#_Toc72246804)

[// hTable.put(null, "Agızlik"); // Run Time Error verir 339](#_Toc72246805)

[// hTable.put("Kanaat", null); // Run Time Error verir 339](#_Toc72246806)

[hTable.put("Ali", ""); // null olmadığından problem olmaz. 339](#_Toc72246807)

[System.out.println(hTable); // {Ali=, Dil=Tad alma organı, Gönül=Kalp} 339](#_Toc72246808)

[TreeMap 340](#_Toc72246809)

[TreeMap key'lerde null kullanmaya müsade etmez ama value'larda istediğimiz kadar null kullanmaya müsade eder. 340](#_Toc72246810)

[Collection'larda ne zaman Tree duyarsanız natural order aklımıza gelmeli. TreeMap elemanlarını console'a yazdırmak istediğimizde key'ler natural order'a göre sıralanır.TreeMap en yavaşlarıdır. 340](#_Toc72246811)

[TreeMap<String, Integer> tMap = new TreeMap<>(); 340](#_Toc72246812)

[tMap.put("Kanaat", null); 340](#_Toc72246813)

[tMap.put("Bezelye", 7); 340](#_Toc72246814)

[tMap.put("Elma", 10); 340](#_Toc72246815)

[tMap.put("Armut", 8); 340](#_Toc72246816)

[System.out.println(tMap); // alfabetik sırada yazar 340](#_Toc72246817)

[// {Armut=8, Bezelye=7, Elma=10, Kanaat=null} 340](#_Toc72246818)

[Enumerator 341](#_Toc72246819)

[iterator Collection'da elemanlar arasında gezer ve Collection'ı update ederdi. 341](#_Toc72246820)

[Enumeration java.util package içinde bir interface'dir. 341](#_Toc72246821)

[Enumaration, iterator Farkları: 341](#_Toc72246822)

[1) iterator ile hemen hemen aynı işi yapar ama Enumeration'da remove() methodu yoktur. programdan verileri silmek bazen tehlikeli olur, bu tür bir önlem almak istediğimizde Enumeration kullanılması işe yarayabilir. silme yetkisi vermemiş oluyoruz. 341](#_Toc72246823)

[2) Iterator'da hasNext(), next(), remove(), hasPrevious(), previous() methodları vardır. Enumaration'da aynı işi yapan farklı isimli methodlar vardır. Mesela; hasMoreElement(), nextElement() gibi 341](#_Toc72246824)

[Enumeration names = hTable.keys(); 341](#_Toc72246825)

[while (names.hasMoreElements()) { 341](#_Toc72246826)

[String key = (String) names.nextElement(); 341](#_Toc72246827)

[System.out.println("Key: " + key + " & Value: " + hTable.get(key)); 341](#_Toc72246828)

[// Key: Ali & Value: 341](#_Toc72246829)

[// Key: Dil & Value: Tad alma organı 341](#_Toc72246830)

[// Key: Gönül & Value: Kalp 341](#_Toc72246831)

[} 341](#_Toc72246832)

[38.DERS ( INTERVIEW HAZIRLIK ) 342](#_Toc72246833)

# JAVA NOTLARI ( NİSAN 2021 )

# KISAYOLLAR

### Shift + cmd + 7 🡺 //

### Alt + 6 🡺 |

### Alt + ç 🡺 &

# 1.DERS ( VARIABLE )

**package** day01variable;

**public** **class** FirstClass {

**public static void main(String[] args){**

//Ekrana Hello World! yazin

System.out.println("Hello World!"); // ; ==> semicolon

// Ekrana Merhaba Dunya! yazin

System.***out***.println("Merhaba Dunya!");

System.***out***.println("A");

System.***out***.print("B");

System.***out***.println("C");

System.***out***.print("A");

System.***out***.println("B");

System.***out***.println("D");

}

}

**package** day01variable;

**public** **class** Variable01 {

**public** **static** **void** main(String[] args) {

**int** maas=3000;

##### char harf ='S'; //charlara değer ataması yaparken mutlaka tek tırnak kullan!

System.***out***.println(maas);

System.***out***.println(harf);

### **int** sayi =123;

System.***out***.println(sayi);

##### boolean isOld= true; // boolean'lar sadece true veya false değerlerini kullanabilirsiniz.

System.***out***.println(isOld); // true

**boolean** isNew = **false**;

System.***out***.println(isNew); // false

##### short nazilliNufus=28000;

System.***out***.println(nazilliNufus);

##### long hucreSayisi=353445435;

System.***out***.println(hucreSayisi);

### **double** pi = 3.14252352352;

##### System.out.println(pi); // java bütün ondalık kesirleri otomatik olarak double kabul eder. rakamın soluna f konulmazsa virgüllü sayı olduğu için bunu double yap diye uyarı verir.

### **float** para = 5.25f;

System.***out***.println(para);

String ogrenciIsmi = "Ali Can";

System.***out***.println(ogrenciIsmi);

}

}

**package** day01variable;

**public** **class** Variable02 {

**public** **static** **void** main(String[] args) {

### **char** harf1= 'a'; // a nın değeri 97

### **char** harf2='b'; // b nin değeri 98

### 

##### System.out.println(harf1 + harf2);// + koyduğumuzda harflerin değerini toplayıp sonucu verir. 195 çıkar.

System.***out***.println(harf1);

System.***out***.println(harf2);

System.***out***.print(harf1);

System.***out***.println(harf2);

##### int aCharDeger='a';

##### System.out.println(aCharDeger);

**int** bCharDeger='!';

System.***out***.println(bCharDeger);

**int** yuzdeCharDeger='%';

System.***out***.println(yuzdeCharDeger);

**int** yildizCharDeger='\*';

System.***out***.println(yildizCharDeger);

}

}

﻿

## ﻿Keywords in a Class

### public class MyFirstClass { } 🡪 public bir Access Modifier’dır.

### Access Modifier bir Class’a kimin ulaşıp kimin ulaşamayacağını belirler

### ﻿MyFirstClass Class ismidir.

### “{” ile başlayıp “}” ile biten kısım Class Body olarak isimlendirilir.

### Class isimleri her zaman büyük harfle baslar

## ﻿﻿Bir Metod nasıl oluşturulur (Create)?

### ﻿public int myFirstMethod ( ) { }

### int return type’dır.

### return type metodun ürettiği sonucun tipidir.

### myFirstMethod( ) metod ismidir.

### Metod isimleri küçük harfle başlar.

### Birden fazla kelime içeriyorsa diğer kelimelerin ilk harfleri büyük olur.

### public bir Access Modifier’dır.

### Access Modifier bir metod’a kimin ulaşıp kimin ulaşamayacağını belirler

### “{” ile başlayıp “}” ile biten kısım Metod Body olarak isimlendirilir.

## ﻿Main Method 🡪 public static void main(String[ ] args) { }

## ﻿Variable Oluşturma (Declaration) :

### Data Tipi (Type) VariableAdı (Name);

### int yaş ;

### char ilkHarf ;

## ﻿Variable’a Değer Atama (Assignment) :

### Data Type VariableName = Value;

### int age = 27;

### char initial =’A’;﻿

## Data Types in Java

### Javada temel olarak iki tip data vardır:

### 1. Primitive Data Types: Primitive data type => boolean, char, byte, short, int, long, float ve double.

### 2. Non-Primitive Data Types: Non-Primitive data type => String.

## “Primitive” ve “Non-Primitive” Arasındaki Farklar

### 1) Primitive’ler sadece value içerir, non-primitive’ler value and methodlar içerir.

### 2) Primitive’ler küçük harf ile, non-primitive’ler büyük harf ile başlar.

### 3) Primitive’leri Java oluşturdu biz primitive oluşturamayız.

### Non-primitive’leri biz de oluşturabiliriz, Java da oluşturabilir.Mesela String’i Java oluşturmuştur.

### 4) Primitive’lerin büyüklükleri data type’ing gore değişir, non-primitive’lerin hepsi aynı büyüklüktedir.

# 2. DERS ( WRAPPERCLASSES)

**package** day02wrapperclasses;

**public** **class** Wrapper01 {

**public** **static** **void** main(String[] args) {

##### int data tipinin max ve min değerlerini bulup ekrana yazdırınız.

// int in wrapper class'ı Integer

System.***out***.println("DATA TYPLERİNİN MAX VE MİN DEĞERLERİNİ YAZDIRMA");

System.***out***.println();

System.***out***.println("INTEGER");

##### System.out.println(Integer.MAX\_VALUE);

System.***out***.println(Integer.***MIN\_VALUE***);

System.***out***.println();

##### bir Class'ın içindeki metodlara ulaşmak için class isminden sonra . koyuyoruz, çıkan listeden seçim yapıyoruz.

// long data tipinin max ve min değerlerini bulup ekrana yazdırınız

System.***out***.println("LONG");

System.***out***.println(Long.***MAX\_VALUE***);

System.***out***.println(Long.***MIN\_VALUE***);

System.***out***.println();

// ODEV: TÜM PRİMİTİVLERİN MAX VE MİN DEĞERLERİNİN WRAPPER CLASSLARINI KULLANARAK EKRANA YAZDIRINIZ

System.***out***.println("BYTE");

System.***out***.println(Byte.***MAX\_VALUE*** );

System.***out***.println(Byte.***MIN\_VALUE*** );

System.***out***.println();

System.***out***.println("SHORT");

System.***out***.println(Short.***MAX\_VALUE***);

System.***out***.println(Short.***MIN\_VALUE*** );

System.***out***.println();

System.***out***.println("FLOAT");

System.***out***.println(Float.***MAX\_VALUE***);

System.***out***.println(Float.***MIN\_VALUE***);

System.***out***.println();

System.***out***.println("DOUBLE");

System.***out***.println(Double.***MAX\_VALUE***);

System.***out***.println(Double.***MAX\_VALUE***);

System.***out***.println();

System.***out***.println("CHAR");

System.***out***.println(Character.***MAX\_VALUE***);

System.***out***.println(Character.***MIN\_VALUE*** );

}

}

## ﻿Kullanıcıdan Her Tür Data Almak

### ﻿1. Scanner scan = new Scanner(System.in);

### 2. System.out.println(“30 dan küçük bir sayı giriniz”);

### 3. int num1 = scan.nextInt();

### 4. System.out.println(num1);

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** daire {

**public** **static** **void** main(String[] args) {

// Dairenin alan ve çevresini hesaplama

Scanner scan= **new** Scanner(System.***in***);

System.***out***.println("Dairenin yarıçapını giriniz");

**float** r=scan.nextFloat();

System.***out***.print("Dairenin alanı = ");

System.***out***.println(3.14159\*r\*r);

System.***out***.println();

System.***out***.print("Dairenin çevresi = ");

System.***out***.println(3.14159\*2\*r);

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** mil {

**public** **static** **void** main(String[] args) {

// Mil'i km'ye çevirme

System.***out***.println("MİL'İ KM'YE ÇEVİRME");

System.***out***.println();

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("MİL DEĞERİNİ GİRİNİZ");

**double** mil=scan.nextDouble();

System.***out***.print(mil );

System.***out***.print( "mil = ");

System.***out***.print(mil\*1.6);

System.***out***.println("km");

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** prizma {

**public** **static** **void** main(String[] args) {

// Prizmanın alanını hesaplama

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Prizmanın En değerini giriniz");

**int** en=scan.nextInt();

System.***out***.println();

System.***out***.println("Prizmanın Boy değerini giriniz");

**int** boy=scan.nextInt();

System.***out***.println();

System.***out***.println("Prizmanın Yükseklik değerini giriniz");

**int** yuk=scan.nextInt();

System.***out***.print("Prizmanın alanı = ");

System.***out***.println(en\*boy\*yuk);

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** prizma {

**public** **static** **void** main(String[] args) {

// Prizmanın alanını hesaplama

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Prizmanın En değerini giriniz");

**int** en=scan.nextInt();

System.***out***.println();

System.***out***.println("Prizmanın Boy değerini giriniz");

**int** boy=scan.nextInt();

System.***out***.println();

System.***out***.println("Prizmanın Yükseklik değerini giriniz");

**int** yuk=scan.nextInt();

System.***out***.print("Prizmanın alanı = ");

System.***out***.println(en\*boy\*yuk);

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** Scanner01 {

**public** **static** **void** main(String[] args) {

// kullanıcı iki tam sayı versin

// program bu tam sayıların toplamını ve çarpımını ekrana yazdırsın.

Scanner scan = **new** Scanner(System.***in***);

// Scanner yazısının üstüne gelip çıkan ekrandan importu seç

System.***out***.println("lütfen iki tam sayı giriniz.");

**int** sayi1 = scan.nextInt();

**int** sayi2 = scan.nextInt();

System.***out***.println(sayi1+sayi2);

System.***out***.println(sayi1\*sayi2);

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** Scanner04 {

**public** **static** **void** main(String[] args) {

/\*

Kullanıcıdan dikdörtgenin boyutlarını alan ve sonra diktörtgenin alan ve

cevresini hesaplayıp ekrana yazdıran bir program yazınız.

Not: Alan: Boy x En

Not: Çevre: 2x (Boy + En)

\*/

Scanner scan = **new** Scanner(System.***in***);

System.***out***.println("Dikdörtgenin en ve boyunu giriniz");

**int** sayi1 = scan.nextInt();

**int** sayi2 = scan.nextInt();

System.***out***.println("Dikdörtgenin alanı:");

System.***out***.println(sayi1\*sayi2);

System.***out***.println("Dikdörtgenin çevresi");

System.***out***.println(sayi1\*2+sayi2\*2);

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** Scanner05 {

**public** **static** **void** main(String[] args) {

// Dairenin alanı ve çervresini bulunuz.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen dairenin yarı çapını giriniz");

**double** yaricap= scan.nextDouble();

System.***out***.print("Dairenin alanı = ");

System.***out***.println(3.14159\*yaricap\*yaricap);

System.***out***.print("Dairenin çevresi = ");

System.***out***.println(2\*3.14159\*yaricap);

// data tipi int ise nextInt() kullanılır

// data tipi double ise nextDouble() kullanılır

scan.close();

}

}

**package** day02wrapperclasses;

**import** java.util.Scanner;

**public** **class** ucgen {

**public** **static** **void** main(String[] args) {

// Ücgenin çevresini hesaplama

System.***out***.println("ÜÇGENİN ÇEVRESİNİ HESAPLAMA");

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("üçgenin ilk kenarını giriniz");

**byte** a=scan.nextByte();

System.***out***.println("üçgenin ikinci kenarını giriniz");

**byte** b=scan.nextByte();

System.***out***.println("üçgenin üçüncü kenarını giriniz");

**byte** c=scan.nextByte();

System.***out***.print("ÜÇGENİN ÇEVRESİ = ");

System.***out***.println(a+b+c);

scan.close();

}

}

# 3. DERS ( OPERATORS )

**package** day03operators;

**import** java.util.Scanner;

**public** **class** Wrapper01 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ad ve soyadını alıp ekrana yazdıran bir program yazınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen Ad ve Soyadınız'ı giriniz");

##### String isim=scan.nextLine();

##### Stringleri kullanıcıdan almak için ya sadece next kullanırsınız yada nextLine() kullanırız. next tek kelimelik Stringler için kullanılır.

##### nextLine satırın tamamını almak, tüm cümleyi almak için kullanılır.

// nextLine girilen tüm String'i almak için kullanılır

System.***out***.print("AD SOYAD :");

System.***out***.println(isim);

scan.close();

}

}

**package** day03operators;

**import** java.util.Scanner;

**public** **class** IsimAdres {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ad, soyad ve adresini alıp ekrana yazdıran bir program yazınız.

// Ad ve soyad ilk satırda, adres ikinci satırda olsun.

//(String kullanınız)

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("adınızı giriniz");

String ad=scan.nextLine();

System.***out***.println("soyadınızı giriniz");

String soyad=scan.nextLine();

System.***out***.println("adresinizi giriniz");

String x=scan.nextLine();

System.***out***.println("Ad soyad: "+ ad + " " + soyad);

System.***out***.print("Adres: " + x);

scan.close();

}

}

**package** day03operators;

**import** java.util.Scanner;

**public** **class** Wrapper02 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ad ve soyadını alalım

// ekrana yazdıralım

// kullanıcıdan adresi alıp ekrana yazdıralım

// kullanıcıdan yaşını alıp ekrana yazdıralım

// kullanıcıya Türkiye'de yaşıyorum doğru/yanlış diye sorup cevabını ekrana yazdıralım

Scanner scan=**new** Scanner(System.***in***);

System.***out***.print("Adı Soyadı:");

String ad=scan.nextLine();

System.***out***.println(ad);

System.***out***.println("Lütfen adresinizi giriniz");

String adres=scan.nextLine();

System.***out***.print("Adres:");

System.***out***.println(adres);

System.***out***.println("Yaşınızı giriniz");

**int** yas=scan.nextInt();

System.***out***.println(yas);

System.***out***.println("Türkiye'demi oturuyorsunuz true/false?");

**boolean** blTr=scan.nextBoolean();

System.***out***.println(blTr);

scan.close();

}

}

**package** day03operators;

**import** java.util.Scanner;

**public** **class** Wrapper03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ilk ismini alın ekrana yazdırın

// Kullanıcıdan soyismini alın ekrana ilk isminin altına yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen isminizi yazınız");

String ad=scan.nextLine();

System.***out***.println("Lütfen soyadınızı giriniz");

String soyad=scan.next();

System.***out***.print("Adı: ");

System.***out***.println(ad);

System.***out***.print("Soyadı: ");

System.***out***.println(soyad);

scan.close();

}

}

**package** day03operators;

**public** **class** TypeCasting01 {

**public** **static** **void** main(String[] args) {

##### byte by=101;

##### int sayi=by;

System.***out***.println(sayi);

##### Küçük data tipini büyük data tipine çevirmek için extra bir kod yazmaya gerek yok. java bunu otomatik olarak yaparak buna Auto widining denir.

**int** sayi2=53;

**byte** by2= (**byte**)sayi2;

System.***out***.println(by2);

##### Büyük data tipini küçük data tipine çevirmek, java tarafından yapılmaz, örnekteki gibi sağ tarafa parantez içinde dönüştürmek istediğimiz data tipini yazmalıyız.

##### 

##### double sayi3=23.9;

##### int by3=(int)sayi3;

##### System.out.println(by3);

##### // sonuc: 23

**float** sayi4=-23.9f;

**short** by4=(**short**)sayi4;

System.***out***.println(by4);

**double** sayi5=4.8;

**double** sayi6=1.4;

**double** sonuc=sayi5/sayi6;

System.***out***.println(sonuc);

//sonuc: 3.428571428571429

**int** sonuc2=(**int**)sonuc;

System.***out***.println(sonuc2);

**int** sayi7=5;

**int** sayi8=3;

**int** sonuc3=sayi7/sayi8;

System.***out***.println(sonuc3);

##### data tipi int olduğu için yuvarlama yapmadan sadece tam sayı kısmı çıkar.

**int** sayi9=255;

**byte** by5=(**byte**) sayi9;

System.***out***.println(by5);

// sonuç -1 verdi.

// Normalde byte da (-128/127)256 sayı var.

// biz 255 yazdık, bir eksik olunca -1 yazdı.sınırı geçen sayılara

// böyle bir mulamele yapıyor.

// 255 i 256 ya böler kalanı yazar

}

}

**package** day03operators;

**public** **class** Ab {

**public** **static** **void** main(String[] args) {

**double** sayi1=5;

**double** sayi2=2;

**double** sonuc=sayi1/sayi2;

System.***out***.println(sonuc);

//sonuc: 2.5

}

}

**package** day03operators;

**public** **class** OdevTypeCasting1 {

**public** **static** **void** main(String[] args) {

// Niçin hata veriyor?

// Bu hatayı iki farklı yolla çözünüz.

// float num1=3.23

// double num2=3.23

**float** num1=3.23f;

**double** num2=num1;

System.***out***.println(num2);

**double** num3=3.235345353457897897983;

**float** num4=(**float**) num3;

System.***out***.println(num4);

}

}

# 4. DERS ( OPERATORS )

**package** day04operators;

**import** java.util.Scanner;

**public** **class** OdevSorusuCozumu01 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan integer aldıktan sonra String alamıyoruz.

// Javadaki bir sorunun çözümü

// Scanner scan=new Scanner(System.in);

//System.out.println("Yaşınınız giriniz");

//int yas=scan.nextInt();

// System.out.println(yas);

// System.out.println("Adınızı ve Soyadınızı giriniz");

// String isim= scan.nextLine();

// System.out.println(isim);

// isim yazdırmıyor, sorun çıkıyor

// " " içine koyduktan sonra her şey String olabilir

// int ve String beraber kullanılınca mı sorun çıkıyor.

//

##### Javada nextInt kullandıktan sonra nextLine kullanırsak kod çalışmıyor.

##### çözüm: nextInt yerine nextLine kullan sonra elde edilen String'i Integer.parsentInt kullanarak integer'a çeviririz.

##### Scanner scan=new Scanner(System.in);

##### System.out.println("Yaşınınız giriniz");

##### String yas=scan.nextLine();

##### int yeniYas=Integer.parseInt(yas);

System.***out***.println("Adınızı ve Soyadınızı giriniz");

String isim= scan.nextLine();

System.***out***.println();

System.***out***.println(isim);

System.***out***.println(yeniYas+" yaşında");

scan.close();

}

}

**package** day04operators;

**public** **class** Modulus01 {

**public** **static** **void** main(String[] args) {

##### 12835 sayisinin 23 e bölümünden kalanı bulan program yazınız

##### Bir sayının diğer bir sayıya bölümünden kalanı bulmaya modulus deniyor. İşlemi yapmak için % işareti kullanılır

##### 

##### int a=12835;

##### int b=23;

##### System.out.println(a%b);

// hard code olarakta yapılır ama tasvip edilmiyor.

// System.out.println(12835%23); şeklindede yapılabilirdi.

//sonuc: 1

}

}

**package** day04operators;

**import** java.util.Scanner;

**public** **class** Modulus02 {

**public** **static** **void** main(String[] args) {

// Kullanıcının girdiği sayının ilk ve son rakamının toplamını veren program yazınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen 4 rakamlı bir tam sayı giriniz");

**int** sayi1=scan.nextInt();

##### Bir tam sayının birler basamağını bulmak için sayının 10 ile modülünü alırız.

##### 

##### int sonRakam=sayi1%10;

##### İlk rakamı bulmak için 4 basamaklı olduğu için 1000 e böleriz.

##### int ilkRakam=sayi1/1000;

System.***out***.println("Son rakam ile ilk rakamın toplamı= "+ (sonRakam+ilkRakam));

scan.close();

}

}

**package** day04operators;

**import** java.util.Scanner;

**public** **class** Modulus03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan 3 basamaklı bir tam sayı alın bu sayının rakamları toplamını bulunuz.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("3 basamaklı bir tam sayı giriniz");

**int** sayi1=scan.nextInt();

**int** sonRakam=sayi1%10;

**int** ilkRakam=sayi1/100;

**int** v=sayi1/10;

**int** ortaRakam=v%10;

System.***out***.println(sonRakam+ilkRakam+ortaRakam);

scan.close();

}

}

**package** day04operators;

**import** java.util.Scanner;

**public** **class** Deneme {

**public** **static** **void** main(String[] args) {

// 3 basamaklı bir sayının rakamları toplamını yazdırma

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("üç basamaklı tam bir sayı giriniz");

**int** sayi1=scan.nextInt();

**int** birler=sayi1%10;

**int** yuzler=sayi1/100;

**int** kalan=sayi1%100;

**int** onlar=kalan/10;

System.***out***.println("girilen sayının rakamları toplamı = "+ (birler+onlar+yuzler));

scan.close();

}

}

**package** day04operators;

**import** java.util.Scanner;

**public** **class** Odev1 {

**public** **static** **void** main(String[] args) {

// Kullanıcının girdiği 4 basamaklı sayının ilk ve son rakamının toplamını

// ekrana yazdıran bir program yazınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("dört basamaklı tam bir sayı giriniz");

**int** sayi1=scan.nextInt();

**int** bir=sayi1%10;

**int** bin=sayi1/1000;

System.***out***.println("Birler ve Binler basamağının toplamı= "+(bir+bin));

scan.close();

}

}

**package** day04operators;

**import** java.util.Scanner;

**public** **class** Odev2 {

**public** **static** **void** main(String[] args) {

// Kullanıcının girdiği 4 basamaklı sayının tum rakamlarının toplamını

// ekrana yazdıran bir program yazınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("dört basamaklı tam bir sayı giriniz");

**int** sayi1=scan.nextInt();

**int** birler=sayi1%10;

**int** binler=sayi1/1000;

**int** v=sayi1/10;

**int** onlar=v%10;

##### int v2=sayi1/100;

##### int yuzler=v2%10;

System.***out***.println("Bu sayının tüm rakamları toplamı= "+(birler+onlar+yuzler+binler));

scan.close();

}

}

**package** day04operators;

**import** java.util.Scanner;

**public** **class** Interviewsorusu01 {

**public** **static** **void** main(String[] args) {

// iki tane variable ın değerlerini yer değiştiriniz.

// sayi1=12, sayi2=30 == sayi1=30, sayi2=12

// bu işleme swap denir, yer değiştirme.

// 1.YOL

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Yerlerini değiştirmek için iki sayı giriniz");

**double** sayi1=scan.nextDouble(); // 12.0

**double** sayi2=scan.nextDouble(); // 30.0

System.***out***.println("Yer değiştirmeden önce");

**double** gecici=0.0;

gecici=sayi1; // Variable oluştururken data tip sadece bir kere yazılır.

// ikinci kez yazarsanız altta kırmızı çizgi çıkar.

sayi1=sayi2;

sayi2=gecici;

System.***out***.println("Yer değiştirdikten sonra");

System.***out***.println(sayi1); // 30.0

System.***out***.println(sayi2); // 12.0

// 2.YOL

// SAYILARIN YERİNİ DEĞİŞTİR AMA GEÇİCİ VARİABLE KULLANMA

System.***out***.println("YER DEĞİŞTİRMEK İÇİN İKİ SAYI DAHA GİRİNİZ");

**double** sayi3=scan.nextDouble();

**double** sayi4=scan.nextDouble();

sayi3=sayi3+sayi4;

sayi4=sayi3-sayi4;

sayi3=sayi3-sayi4;

System.***out***.println("Yer değiştirdikten sonra");

System.***out***.println(sayi3); // 30.0

System.***out***.println(sayi4); // 12.0

scan.close();

}

}

# 5. DERS ( OPERATORS )

**package** day05operators;

**public** **class** Increment01 {

**public** **static** **void** main(String[] args) {

// int data type'inde iki variable oluşturun

// birincisini 3 ikincisini 7 artırın

**int** sayi1=12;

sayi1= sayi1+3; // bu tanımlamaya increment deniyor.

System.***out***.println(sayi1);

**int** sayi2=18;

sayi2=sayi2+7;

System.***out***.println(sayi2);

}

}

**package** day05operators;

**public** **class** Increment02 {

**public** **static** **void** main(String[] args) {

// iki tane int data type nda variable oluşturun

// birinciyi 3, ikinciyi 7 artırın

// kısa yol kullan

**int** sayi1=12;

sayi1+=3; // + ile = arasına boşluk bırakmayın.

System.***out***.println(sayi1);

**int** sayi2=18;

sayi2+=7;

System.***out***.println(sayi2);

}

}

**package** day05operators;

**public** **class** Increment03 {

**public** **static** **void** main(String[] args) {

// int data typ inde bir tane variable oluşturun.

// bu variable ları EN KISA yolu kullanarak 1 artırın

// en kısa yol sadece 1 artırmak için kullanılır.

**int** sayi1=12;

##### sayi1++;

##### Variable isminden sonra iki kere artı sembolu yazarsanız bu variablenin değerini bir artırmış olursunuz.

System.***out***.println(sayi1);

**int** sayi2=10;

sayi2++;

System.***out***.println(sayi2);

}

}

# 6. DERS ( IF STATEMENT )

**package** day06ifstatement;

**public** **class** IfStatement01 {

**public** **static** **void** main(String[] args) {

##### Parantez içindeki koşul(condition) doğru ise body içerisindeki komutu çalıştırır. değilse bir sonraki satıra geçer

##### 

##### if(3>2) {

##### System.out.println("if body çalıştı");

##### }

// bir int variable oluşturup ona bir değer atayın

// eğer atadığınız değer pozitif ise ekrana pozitif yazdırın

// eğer atadığınız değer negatif ise ekrana negatif yazdırın

// eğer atadığınız değer sıfır ise ekrana nötr yazdırın

**int** num1=12;

**if**(num1>0) {

System.***out***.println("Pozitif");

}

**if**(num1<0) {

System.***out***.println("Negatif");

}

**if**(num1==0) {

System.***out***.println("Nötr");

}

}

}

**package** day06ifstatement;

**public** **class** IfStatement02 {

**public** **static** **void** main(String[] args) {

// bir int variable oluşturun

// deger atayın

// eğer çift sayı ise ekrana çift yazdırın

// değer tek sayı ise ekrana tek yazdırın

**int** num1=13;

**if**(num1%2==0) {

System.***out***.println("ÇİFT");

}

**if**(num1%2!=0) {

System.***out***.println("TEK");

}

}

}

**package** day06ifstatement;

**public** **class** IfStatement03 {

**public** **static** **void** main(String[] args) {

// Bir Char variable oluşturup bir büyük harf değer atayın

// eğer atanan değer haftanın günlerinden herhangi birinin ilk harfi ise

// o günleri yazdırın.

// eğer atanan değer haftanın günlerinden herhangi birinin ilk harfi değil ise

// böyle bir gün yok yazsın.

**char** gun='C';

**if**(gun=='P') {

System.***out***.println("Pazar,Pazartesi,Perşembe");

}

**if**(gun=='S') {

System.***out***.println("Salı");

}

**if**(gun=='C') {

System.***out***.println("Carsamba,Cuma,Cumartesi");

}

**if** (gun!='P' || gun!='C' || gun!='S') {

System.***out***.println("Böyle bir gün yok");

// "P" olmamak VEYA "C" olmamak VEYA "S" olmamak ==> VEYA işlemi icin || ==> | -> Pipe denir

##### VEYA isleminden True sonucu alabilmek bir tanesinin True olmasi yeterlidir

##### True||True = True <-> True||False = True <-> False||False = False

##### True=1, False=0 ==> VEYA islemi toplamaya benzer

// "P" olmamak VE "C" olmamak VE "S" olmamak ==> VE islemi icin

##### && VE isleminden True sonucu alabilmek icin tum condition'lar true olamalidir

##### True && True = True <-> True && False = False <-> False && False = False

##### True=1, False=0 ==> VE islemi carpmaya benzer

}

}

}

##### & ( alt+ç) kısayol

##### | (alt+6) kısayol

**package** day06ifstatement;

**public** **class** IfStatement04 {

**public** **static** **void** main(String[] args) {

// bir tane string variable oluşturun ve bir gün ismini küçük harflerle

// değer olarak atayın

// eğer atadığınız değer hafta içi günlerinden biri ise

// ekrana hafta içi yazdırın

// değilse hafta sonu yazdırın

String gun = "pazartesi";

**if**(gun.equals("pazartesi") || gun.equals("sali") || gun.equals("carsamba") || gun.equals("perşembe") || gun.equals("cuma")) {

System.***out***.println("HAFTA İÇİ");

}

**if**(gun.equals("cumartesi") || gun.equals("pazar")) {

System.***out***.println("HAFTA SONU");

}

}

}

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfStatement05 {

**public** **static** **void** main(String[] args) {

// kullanıcıdan bir dikdörtgenin en ve boyunu alın

// en ve boy eşit ise ekrana bu bir karedir yazdırın

// en ve boy farklı ise ekrana bu bir dikdörtgendir yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("en değerini giriniz");

**double** en=scan.nextDouble();

System.***out***.println("boy değerini giriniz");

**double** boy=scan.nextDouble();

// if(en==boy) {

// System.out.println("Bu bir karedir");

// }

// if (en!=boy) {

// System.out.println("Bu bir dikdörtgendir");

// }

//

// else ==> diğer ihtilamllerin tamamı veya denilenin haricinde veya değilse

**if**(en==boy) {

System.***out***.println("bu bir karedir");

}**else** {

System.***out***.println("bu bir dikdörtgendir");

}

scan.close();

}

}

**package** day06Odev;

**import** java.util.Scanner;

**public** **class** Odev0601 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir tamsayı alın eğer tamsayı 3 ile bölünebiliyorsa ekrana “3’ün katı” yazdırın.

// 3 ile bölünemiyorsa ekrana “3’ün katı değildir” yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir tam sayı giriniz");

**int** num1=scan.nextInt();

**if**(num1%3==0) {

System.***out***.println(num1+" Sayısı üçün katıdır");

}**else** {

System.***out***.println(num1+" Sayısı üçün katı değildir");

}

scan.close();

}

}

**package** day06Odev;

**import** java.util.Scanner;

**public** **class** Odev0602 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir harf alın eğer harf “a, e, i, o, u” dan biri ise ekrana “Sesli harf” yazdırın.

// “b, c, d, e” den biri ise ekrana “Sessiz harf” yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir harf yazınız");

**char** harf=scan.next().charAt(0);

**if**(harf=='a'|| harf=='e'|| harf=='i'|| harf=='o'|| harf=='u') {

System.***out***.println(" \""+harf+ "\""+ " harfi sesli bir harftir.");

}**else** **if**(harf=='b'|| harf=='c'|| harf=='d'|| harf=='e') {

System.***out***.println("\""+harf+"\""+" harfi sessiz bir harftir.");

}**else** {

System.***out***.println("Yanlış harf girdiniz");

}

scan.close();

}

}

**package** day06Odev;

**import** java.util.Scanner;

**public** **class** OdevCharSorusu {

**public** **static** **void** main(String[] args) {

// // Kullanıcıdan bir character alın eğer character bir harf ise ekrana “Harf” yazdırın.

// Diğer durumlarda ekrana “Harf değil” yazdırın.

Scanner scan=**new** Scanner(System.***in***);

##### System.out.println("Bir character girin");

##### char harf=scan.next().charAt(0); // Kullanıcıdan char almak için next().charAt(0); kullanılır.

// ASCII TABLE a-z==> 97-122 A-B==> 65-90

**if**((harf<=122 && harf>=97)||(harf<=90 && harf>=65)) {

System.***out***.println("Bu bir harftir");

}**else** {

System.***out***.println("Bu bir harf değildir");

}

scan.close();

}

}

# IF ELSE STATEMENT

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfElseStatement01 {

**public** **static** **void** main(String[] args) {

// kullanıcıdan password girmesini isteyin

// password doğru ise ekrana "Password doğru" yazdırın

// password yanlış ise " Password yanlış tekrar deneyin" yazdırın

// doğru password = Java2020

// iki memory var Stack ve Heap

// Stack içinde primitive ve reference lar.

// biz String oluşturduğumuzda java onu Heap'e koyar.

// kullanıcının girdiği password ile java2020 in

// değerler ve referencler Stack içinde aynı olması lazım

// bu yüzden doğru şifreyi girmiş olsak bile reference adres uyuşmadığı için sonucu

// yanlış olarak verecektir.

// problemin çözümü java nın adreslere bakmasını engelleyeceğiz sadece değerlere bak

// diyeceğiz. == sembolünü kullanmıyoruz. String ismini yazdıktan sonra . koyup equals

// seçiyoruz. equals içine "Java2020" yazıyoruz.

// equals sadece değerleri kontrol eder, adresleri değil.

##### Stringlerin eşitliğini kontrol etmek istiyorsak asla!!! == sembolu kullanmıyoruz.Bunun yerine equals methodu kullanıyoruz. "==" Stringin değerini ve adresini kontrol eder. "equals()" ise sadece değerleri kontrol eder.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir password giriniz");

String pas=scan.next();

**if**(pas.equals("Java2020")) {

System.***out***.println("Password doğru");

}**else** {

System.***out***.println("Password yanlış tekrar deneyin");

}

scan.close();

}

}

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfElseStatement02 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir tam sayı isteyin ve o tam sayının mutlak değerini ekrana yazdırın.

// mutlak değer bir sayının kendi işareti ile çarpılması demektir.

// -3 un mutlak değeri

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir tam sayı giriniz");

**int** num=scan.nextInt();

**if**(num>=0) {

System.***out***.println(num);

}**else** {

System.***out***.println(num\*-1);

}

scan.close();

}

}

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfElseStatement03 {

**public** **static** **void** main(String[] args) {

// kullanıcıdan alacağı urun miktarını ve fiyatını alalım

// eğer kullanıcı 1000 den fazla ürün alırsa %20 indirim yapın ve

// kullanıcının ödeyeceği toplam fiyatı ekrana yazdırın

// 1000 den az veya 1000 ürün alırsa indirim yapmayın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Satın almak istediğiniz ürün miktarını giriniz");

**double** miktar=scan.nextDouble();

System.***out***.println("Aldığınız ürünün birim fiyatını giriniz");

**double** fiyat=scan.nextDouble();

**if**(miktar>1000) {

fiyat=fiyat\*(0.80);

}

System.***out***.println("ÜRÜN MİKTARI: "+ miktar);

System.***out***.println("Toplam Fiyat miktarı:"+ fiyat\*miktar);

scan.close();

}

}

# IF ELSE IF

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfElseIfStatement01 {

**public** **static** **void** main(String[] args) {

// kullanıcıdan yaşını alın

// yaş 13 den az ise ekrana " Calışamaz" yazdırın

// yaş 13 den 65 e kadar ise ekrana "Calışabilir" yazdırın

// yaş 65 den büyük ise ekrana "Emekli" yazdırın

// yas negatif olursa "Yas nefatif olamaz" yazsın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Yasinizi giriniz");

**int** yas=scan.nextInt();

**if**(yas<0) {

System.***out***.println("Yas negatif olamaz");

} **else** **if**(yas<13) {

System.***out***.println("Calışamaz");

}**else** **if**(yas<=65) {

System.***out***.println("Calışabilir");

}**else** {

System.***out***.println("Emekli");

}

scan.close();

}

}

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfElseIfStatement02 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan sınav notunu tam sayı olarak girmesini isteyin

// Negatif girerse ekrana yanlış değer girdiniz yazdırın

// 0 dan büyük eşit ve 50 den küçük girerse ekrana kaldınız

// 50 den büyük eşit ve 80 den küçük ise ekrana geçtiniz yazdırın

// 80 den büyük ve eşit 100 den küçük ise ekrana tebrikler yazdırın

// 100 den büyük değer girerse ekrana yanlış değer girdiniz yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println(("Notunuzu tam sayı olarak giriniz"));

**int** not=scan.nextInt();

**if**(not<0) {

System.***out***.println("Yanlış değer girdiniz");

}**else** **if**(not>100) {

System.***out***.println("Yanlış değer girdiniz");

}**else** **if** (not<50) {

System.***out***.println("Kaldınız");

}**else** **if**(not<80) {

System.***out***.println("Geçtiniz");

}**else** {

System.***out***.println("Tebrikler");

}

scan.close();

}

}

**package** day06ifstatement;

**import** java.util.Scanner;

**public** **class** IfElseIfStatement03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan altı basamaklı bir sayıisteyin. 0 ilede başlıyabilir

// kazanma ihtimali %20, amorti ihtimali(tekrar oynama) %30, kaybetme ihimali %50

// olan bir jakpot oyunu için program yazınız.

// 00000-000001-000002-.....-99998,-999999 ==> 1000000 bir milyon sayı var.

// bir milyon ihtimalden %20 sine kazandı diyeceğiz

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Altı basamaklı bir sayı giriniz. 0 başta kullanılabilir");

**int** num=scan.nextInt();

##### if(num<200000) {

##### System.out.println("Kazandınız");

##### }else if(num<500000) {

##### System.out.println("Amorti");

##### }else

##### System.out.println("Kaybettiniz");

##### scan.close();

##### 

##### }

}

# 7. DERS ( NESTED IF STATEMENT)

**package** day07nestedif;

**import** java.util.Scanner;

**public** **class** NestedIf01 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir sayı alın.

// bu sayı pozitif ise 9'dan büyük mü diye kontrol edin.

// 9'dan büyük ise ekrana "Sayı" yazdırtın.

// 9'dan küçük eşit ve 0'dan büyük eşit ise ekrana "Rakam" yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir tam sayı giriniz");

**int** num=scan.nextInt();

##### if(num>=0) {

##### if(num>9) {

##### System.out.println("Sayi");

##### }else if(num<=9 && num>=0) {

##### System.out.println("Rakam");

##### 

##### scan.close();

##### }

##### Scanner Class'ının kullandıktan sonra en altta main method'un içinde Scanner Class'ını kapatınız. Scanner Class'ını kapatmak için scan.close(); yazmak yeterlidir.

}

}

}

**package** day07nestedif;

**import** java.util.Scanner;

**public** **class** NestefIf02 {

**public** **static** **void** main(String[] args) {

// kullanıcıdan password girmesini isteyin

// password dört basamaklı bir sayı ise

// çift olup olmadığını kontrol edin

// cift sayıise "Password tamam" yazdırın

// tek sayı ise "Tekrar deneyin" yazdırın

// Password dört basamaklı değilse " Tekrar deneyin" yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Tam sayı olan bir password giriniz");

**int** pas=scan.nextInt();

**if**(pas>999 && pas<10000) {

if(pas%2==0) {

System.***out***.println("Password tamam");

}**else** **if** (pas%2!=0){

System.***out***.println("Tekrar deneyin");

}

} **else** {

System.***out***.println("Tekrar deneyin");

}

scan.close();

}

}

**package** day07nestedif;

**import** java.util.Scanner;

**public** **class** NestedIf03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan yıl girmesini isteyin, girilen yıl artık yıl ise

// ekrana artık yıl yazdırın. girilen yıl artık yıl değilse

// ekrana artık yıl değil yazdırın.

// şubat'ın 29 çektiği yıla artık yıl denir. 4 yılda bir olur.

// artık yilin iki şarıt var. yıl 100'e bölünürse 400'e de bölünmeli

// yıl 100'e bölünmezse 4'e bölünmeli

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir yıl giriniz ");

**int** yil=scan.nextInt();

**if**(yil%100==0) {

**if**(yil%400==0) {

System.***out***.println("Artik yıl");

}

}**else** **if**(yil%4==0) {

System.***out***.println("Artık yıl");

}**else** {

System.***out***.println("Artık yıl değil");

}

scan.close();

}

}

**package** day07nestedif;

**import** java.util.Scanner;

**public** **class** Odev01 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan cinsiyetini girmesini isteyin.

// Erkek ise yaşını kontrol edin. Yaşı 18 den küçük ise ekrana “Erkek çocuk” yazdırın.

// Yaşı 18 den büyük eşit ise ekrana “Adam” yazdırın.

// Kadin ise yasini kontrol edin. Yaşı 18 den küçük ise ekrana “Kız çocuk” yazdırın.

// Yaşı 18 den büyük eşit ise ekrana “Kadın” yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Cinsiyetinizi giriniz (Erkek/Bayan)");

String cns=scan.next();

System.***out***.println("Yaşınızı giriniz");

**int** yas=scan.nextInt();

**if**(cns.equalsIgnoreCase("Erkek")) {

**if**(yas<18) {

System.***out***.println("Erkek çocuk");

}**else** {

System.***out***.println("Adam");

}}

**if**(cns.equalsIgnoreCase("bayan")) {

**if**(yas<18) {

System.***out***.println("Kız Çocuk");

}**else** {

System.***out***.println("Kadın");

} scan.close();

}

}

}

**package** day07nestedif;

**import** java.util.Scanner;

**public** **class** Odev02 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir harf girmesini isteyin.

// Girdiği küçük harf ise harfin “a” olup olmadığını kontrol edin. Harf “a” ise ekrana “Ilk küçük harf” yazdırın.

// “a” değil ise ekrana “Ilk küçük harf değil” yazdırın.

// Girdiği büyük harf ise harfin “Z” olup olmadığını kontrol edin. Harf “Z” ise ekrana “Son büyük harf” yazdırın.

// “Z” değil ise ekrana “Son büyük harf değil” yazdırın.

// ASCII TABLE a-z==> 97-122 A-B==> 65-90

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir harf giriniz");

**char** harf=scan.next().charAt(0);

**if**(harf>=97 && harf<=122) {

**if**(harf==97) {

System.***out***.println("İlk küçük harf");

}**else** **if**(harf!=97){

System.***out***.println(harf+ " harfi bir küçük harftir.");

}}

**if**(harf>=65 && harf<=90) {

**if**(harf==90) {

System.***out***.println("Z harfi son büyük harftir");

}**else** **if**(harf!=90) {

System.***out***.println(harf+" harfi bir büyük harftir.");

}}

scan.close();

}

}

**package** day07nestedif;

**import** java.util.Scanner;

**public** **class** Odev03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan password girmesini isteyin.

// Girdiği password 5’e bölünüyorsa son rakamını kontrol edin.

// Son rakamı 0 ise ekrana “5’e bölünen çift sayı” yazdırın.

// Son rakamı 0 değil ise “5’e bölünen tek sayı” yazdırın.

// Girdiği password 5’e bölünmüyorsa ekrana “Tekrar deneyin” yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir Password giriniz");

**int** pas=scan.nextInt();

**if**(pas%10==0) {

System.***out***.println("5'e bölünebilen çift sayı");

}**else** **if**(pas%10==5) {

System.***out***.println("5'e bölünebilen tek sayı");

}**else** **if**(pas%5!=0) {

System.***out***.println("Tekrar deneyin");

}

scan.close();

}}

# 8.DERS ( TERNARY OPERATOR )

##### ﻿Condition ? Code 1 : Code 2

##### True False

**package** day08ternaryoperator;

**import** java.util.Scanner;

**public** **class** TernaryOperator01 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir sayı alınız

// sayı pozitif veya 0 ise ekrana Pozitif yazdırınız.

// sayı negatif ise ekrana Negatif yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir tam sayı giriniz");

**int** num=scan.nextInt();

// Ternary Operator if else

// Condition(şart)ı yazıp soru işareti koyuyoruz.

// sonra Condition doğru olduğunda ne yapacak onu yazıyoruz ve :

// sonra Condition yanlış ise

//Syntax (komut satırı)

// = condition + ?+ iş + yanlışsa yapılacak iş;

##### String sonuc= num>=0 ? "Poziftif veya 0" : "Negatif";

System.***out***.println(sonuc);

scan.close();

}

}

**package** day08ternaryoperator;

**import** java.util.Scanner;

**public** **class** TernaryOperator02 {

**public** **static** **void** main(String[] args) {

// kullanıcıdan iki sayı alın bu sayılar birbirine eşit ise

// toplamlarını ekrana yazdırın

// bu sayılar birbirinden farklı ise

// çarpımlarını ekrana yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen iki sayı giriniz");

**double** num1=scan.nextDouble();

**double** num2=scan.nextDouble();

**double** sonuc= num1==num2 ? (num1+num2) : (num1\*num2);

System.***out***.println(sonuc);

scan.close();

}

}

**package** day08ternaryoperator;

**import** java.util.Scanner;

**public** **class** TernaryOperator03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan iki sayı alınız.

// küçük sayıyı ekrana yazdırınız

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("iki sayı giriniz");

**int** num1=scan.nextInt();

**int** num2=scan.nextInt();

**int** result=num1>=num2 ? num2 : num1;

System.***out***.println(result);

scan.close();

}

}

**package** day08ternaryoperator;

**import** java.util.Scanner;

**public** **class** TernaryOperator04 {

**public** **static** **void** main(String[] args) {

// Ternary Operator (IF ELSE IF) JAVA RESMİ SERTİFİK SINAVINDA GENELDE BU ÇIKAR

// kullanıcıdan bir sayı alınız.

// sayı 0 dan büyük eşit ise

// 10 dan küçük olup olmadığını kontrol ediniz.

// 10 dan küçük ise ekrana Rakam yazdırınız değilse Pozitif Sayı yazdırınız

// sayı 0 dan küçük ise ekrana negatif sayı yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir sayı giriniz");

**int** num1=scan.nextInt();

// ? İŞARETİNDEN SONRA PARANTEZ AÇARSAK İÇİNE YAZILANLAR NE İSE ONU UYGULAR

// BUDA İKİNCİ BİR İF GİBİ DÜŞÜNÜLEBİLİR.

String result=num1>=0 ? (num1<10 ? "Bir Rakamdır" : "Bir Pozitif sayıdır"): "Bir Negatif sayıdır";

System.***out***.println(num1+" " +result);

scan.close();

}

}

**package** day08ternaryoperator;

**import** java.util.Scanner;

**public** **class** TernaryOperator05 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir karakter girmesini isteyiniz.

// karakter harf ise küçük harf olup olmadığını kontrol ediniz.

// küçük harf ise ekrana Küçük harf yazdırınız.

// büyük harf ise ekrana büyük harf yazdırınız.

// harf değilse harf değil yazdırınız.

// a-z 97-122

// A-Z 65-90

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir karakter giriniz");

**char** ch=scan.next().charAt(0);

String result= (ch>=97 && ch<=122) ? "küçük harftir":

((ch>=65 && ch<=90)? "büyük harftir": "harf değildir.");

System.***out***.println(ch +" "+result);

scan.close();

}

}

**package** day08ternaryoperator;

**import** java.util.Scanner;

**public** **class** TernaryOperator06 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir sayı alınız.

// sayı çift ise 10'a bölünüp bölünmediğini kontrol ediniz.

// Sayı 10'a bölünüyorsa "Wooow 10" yazdırınız

// 10'a bölünmüyorsa "Yazık" yazdırınız

// sayı tek ise 5 e bölünüp bölünmediğini kontrol ediniz.

// sayı 5 e bölünüyorsa "Wooow 5" yazdırınız.

// sayı 5 e bölünmüyorsa "yazık 5" yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir tam sayı giriniz");

**int** num=scan.nextInt();

//Condition soru işareti True değilse False

String result= num%2==0 ? (num%10==0 ? "Wooow 10" : "Yazık 10") : (num%5==0 ? "Wooow 5" :"Yazık 5");

System.***out***.println(result);

scan.close();

}

}

**package** day08ternaryoperator;

**public** **class** TekAndCiftAnd {

**public** **static** **void** main(String[] args) {

##### && işleminde hız kazanmak için ilk karşılaştırmaya bakar yanlış ise ikinci karşılaştırmaya bakmaya gerek duymadan direk false der. Bu yazılış yükü azaltır,hatayı azaltır.

##### & işleminde ise bütün karşılaştırmalar kullanılır.

##### & ile && aynı sonucu verir.

// System.out.println(3<2 && 5>4);

// System.out.println(3<2 && 5>4 && 6>4 && 1>4 && 5>7 && 4>4);

System.***out***.println(3<2 & 5>4);

}

}

# 9. DERS ( SWİTCH STATEMENT )

**package** day09switchoperator;

**public** **class** Switch01 {

**public** **static** **void** main(String[] args) {

// A harfini bulunca "İlk harf". B harfini bulunca "İkinci harf"

// C harfini bulunca "Ücüncü harf", D harfini bulunca "Dördüncü harf"

// yazdıran programı if else if( ) ve switch () kullanarak yazınız.

**char** harf='A';

// 1.yol==> if else if

**if**(harf=='A') {

System.***out***.println("İlk harf");

}**else** **if**(harf=='B') {

System.***out***.println("İkinci harf");

}**else** **if**(harf=='C') {

System.***out***.println("Üçüncü harf");

}**else** **if**(harf=='D') {

System.***out***.println("Dördüncü harf");

}**else** {

System.***out***.println("Bilmiyorum");

}

// 2.Yol ==> switch()

**switch**(harf) {

**case** 'A':

System.***out***.println("İlk harf");

**break**;

// üst satırın açılımı: A durumu olduğunda bunu yap dışarı çık.

// ikiden fazla else if varsa switch kullanılır.

**case** 'B':

System.***out***.println("İkinci harf");

**break**;

**case** 'C':

System.***out***.println("Üçüncü harf");

**break**;

**case** 'D':

System.***out***.println("Dördüncü harf");

**break**;

**default**:

System.***out***.println("Bilmiyorum");

}

}

}

**package** day09switchoperator;

**import** java.util.Scanner;

**public** **class** Switch02 {

**public** **static** **void** main(String[] args) {

// Kullanıcı haftanın gün numarasını girsin

// Programınız günün ismini yazsın

// yanlış gün sayısı girerse geçerli gün sayısı giriniz.

// switch kullanarak yapınız

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Haftanın gün numarasını giriniz");

**int** day=scan.nextInt();

**switch**(day) {

**case** 1:

System.***out***.println("Pazartesi");

**break**;

**case** 2:

System.***out***.println("Salı");

**break**;

**case** 3:

System.***out***.println("Çarşamba");

**break**;

**case** 4:

System.***out***.println("Perşembe");

**break**;

**case** 5:

System.***out***.println("Cuma");

**break**;

**case** 6:

System.***out***.println("Cumartesi");

**break**;

**case** 7:

System.***out***.println("Pazar");

**break**;

**default**:

System.***out***.println("geçerli gün sayısı giriniz");

}

scan.close();

}

}

**package** day09switchoperator;

**import** java.util.Scanner;

**public** **class** Switch03 {

**public** **static** **void** main(String[] args) {

// Kullanıcı ay ismini girsin

// program kaçıncı ay olduğunu ekrana yazsın

// büyük küçük harf hepsi için çalışsın

// yanlış ay ismi girerse geçerli ay ismi giriniz diye mesaf versin

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Ay ismini giriniz");

##### String ay=scan.next();

##### ay=ay.toLowerCase();

**switch** (ay) {

**case** "ocak":

System.***out***.println("1.ay");

**break**;

**case** "şubat":

System.***out***.println("2.ay");

**break**;

**case** "mart":

System.***out***.println("3.ay");

**break**;

**case** "nisan":

System.***out***.println("4.ay");

**break**;

**case** "mayıs":

System.***out***.println("5.ay");

**break**;

**case** "haziran":

System.***out***.println("6.ay");

**break**;

**case** "temmuz":

System.***out***.println("7.ay");

**break**;

**case** "ağustos":

System.***out***.println("8.ay");

**break**;

**case** "eylül":

System.***out***.println("9.ay");

**break**;

**case** "ekim":

System.***out***.println("10.ay");

**break**;

**case** "kasım":

System.***out***.println("11.ay");

**break**;

**case** "aralık":

System.***out***.println("12.ay");

**break**;

**default**:

System.***out***.println("Geçerli bir ay ismi giriniz");

}

scan.close();

}

}

**package** day09switchoperator;

**import** java.util.Scanner;

**public** **class** Switch04 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan notunu alın ve aşağıdaki kurallara göre ekrana A, B, C veya D yazdırın.

// 1. 50 den az - D

// 2. 50(dahil) ile 60 arası - C

// 3. 60(dahil) ile 80 arası - B.

// 4. 80(dahil) ustu- A

// yanlış not girilirse ekrana geçerli not giriniz yazdırın

##### switch() de long, float, double ve boolean kullanılmaz.

##### switch() int, byte, short, char,String kullanılır.

##### switch() de case den sonra sadece bir değer yazabilirsiniz, herhangi bir işlem yapamazsınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen notunuzu giriniz");

**int** not=scan.nextInt();

**if**(not>=0 && not<50) {

not=0;

}**else** **if**(not>=50 && not<60) {

not=50;

}**else** **if**(not>=60 && not<80) {

not=60;

}**else** **if**(not>=80 && not<=100) {

not=80;

}

##### switch (not) {

##### case 0:

##### System.out.println("D");

##### break;

##### case 50:

##### System.out.println("C");

##### break;

##### case 60:

##### System.out.println("B");

##### break;

##### case 80:

##### System.out.println("A");

##### break;

##### default:

##### System.out.println("Geçerli not giriniz");

##### 

##### }

scan.close();

}

}

# 10. DERS ( STRING METHODS )

**package** day10Stringmethods;

**public** **class** StringMethod01 {

**public** **static** **void** main(String[] args) {

## indexOf() method'u String içindeki karakterin indexini belirlemek için kullanılır.

##### 

##### String str1="Javva World";

##### System.out.println(str1.indexOf('v'));

##### 

##### Ekrana 2 yazdırır.index sayımı 0 dan başlar ve 2 olarak yazar.

System.***out***.println(str1.indexOf('W'));

// Ekrana 6 yazdırır. space Java için bir character dir.

System.***out***.println(str1.indexOf('w'));

// Java index bulamaz çünkü java "case sensitive" oidb

// küçük w Stringde yok. Java character bulamayınca -1 return eder.

System.***out***.println(str1.indexOf('a'));

// Ekrana 1 yazdırır. birden fazla kullanılan characterler için Java ilk characterin indexini verir.

// İNDEXOF() İKİNCİ VERSİYONU

##### System.out.println(str1.indexOf('a',2));

// String'deki ikinci a karakterinin index'ini bulsun.

// ekrana 4 yazdırmalı.

// aradığımız harfi yazıyoruz. "," 2 yazıyoruz.

##### ikinci index ten itibaren 'a'yı arayıp indexini yazdır.

System.***out***.println(str1.indexOf('a',4));

// ekrana 4 yazdırır. ikinci a'dan itibaren aramaya başlar

System.***out***.println(str1.indexOf('a',1));

// ekrana 1 yazdırır

##### System.out.println(str1.indexOf('a',5));

##### 5. indexten itibaren a olmadığı için ekrana -1 yazdırır.

// "Alamanya" Stringindeki ikinci 'a' karakterinin indexini bulunuz

String str2="Alamanya";

System.***out***.println(str2.indexOf('a'));

// birinci 'a' nın indexini bulur 2 yazar.

// idx= index için kısaltma olarak genellikle kullanılır

**int** idx=str2.indexOf('a');

System.***out***.println(str2.indexOf('a',3));

// ekrana 4 yazdırır.

System.***out***.println(str2.indexOf('a',idx+1));

// aynı karakterden bir çok defa tekrar ediliyorsa bir sonraki aynı karakterin index ini bulur.

// İNDEXOF() METHODUNUN 3. VERSİYONU

String str3="Missisippi";

// "is" Stringini arayalım.

// ilk karakterin index'ini return eder.

// ilk "is" deki 'i' harfinin index'i olan 1 i return eder.

System.***out***.println(str3.indexOf("is"));

}

}

**package** day10Stringmethods;

**public** **class** StringMethod02 {

**public** **static** **void** main(String[] args) {

## lastIndexOf() methodu aradığınız karakterin String içindeki son görünümünün indexini return eder.

// a harfininin en son nerede görüldüğünü bulur.

String str1="Sivrihisar";

System.***out***.println(str1.lastIndexOf('i'));

// ekrana 6 yazdırır.

##### System.out.println(str1.lastIndexOf('r'));

##### // ekrana 9 yazdırır.

// LASTİNDEXOF() 2. VERSİYON

##### System.out.println(str1.lastIndexOf('i',7));

##### ilk 7 indexin içinde i harfinin son görümünün indexini return eder.

System.***out***.println(str1.lastIndexOf('i',5));

// ilk 5 indexin içinde geçen en son i harfini return eder

// LASTİNDEXOF() METHODU 3. VERSİYON

String str2="rivrihisar";

System.***out***.println(str2.lastIndexOf("ri"));

// String içindeki "ri" nin son görünümünü bulacak.

// son "ri" deki 'r' harfinin indexinin return edecek.

}

}

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** StringMethod03 {

**public** **static** **void** main(String[] args) {

## charAt() methodu belli indexteki characteri return eder.

String str1="cimbombom";

// str1 in 4. indexindeki karakteri bulan kodu yazınız.

##### System.out.println(str1.charAt(4));

// index i 4 olan karakter o harfidir.

System.***out***.println(str1.charAt(0));

// index 0 ilk harfi gösterir.

// System.out.println(str1.charAt(10));

##### olmayan bir index ile işlem yapmaya çalışırsanız console da exeption alırsınız.

// buda büyük bir hata yaptınız demektir.

## length() methodu Stringin içinde kac karakter olduğunu verir.

System.***out***.println(str1.length());

// ekrana 9 yazdırır.index saymaya 0 dan başlar

##### length() saymaya 1 den başlar.

// son indexi 9 olan bir Stringin length i kaçtır.

// cevap: 9+1=10

// length =son index+1

// Kullanıcıdan ismini girmesini isteyin

// isminin son harfini ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen isminizi giriniz");

String isim=scan.nextLine();

**int** sonch=isim.length();

sonch=sonch-1;

System.***out***.println("son karakter : "+isim.charAt(sonch));

scan.close();

}

}

**package** day10Stringmethods;

**public** **class** StringMethod04 {

**public** **static** **void** main(String[] args) {

## contains() methodu bir Stringin içinde bir characterin olup olmadığını kontrol eder.

##### eğer o karakter Stringin içinde varsa true yoksa false return eder.contains methodunun içine her zaman String koyuyoruz. charı kabul etmez.

// çok kullanılan bir methoddur.

String str1="karakartal";

System.***out***.println(str1.contentEquals("m"));

// false görürürüz.

//contains

##### System.out.println(str1.contains("kar"));

// true görünür.

## trim metodu bir Stringin baş ve sonundaki boşlukları siler.

// " Ali " baş ve sondaki boşlukları siler "Ali" haline getirir.

// "Ali Can" ortadaki boşluklara dokunmaz.

##### String str2=" Ali Can ";

System.***out***.println(str2);

##### System.out.println(str2.trim());

## isEmpty() methodu bir Stringin içinde character olup olmadığına bakar.character yoksa true, varsa false return eder.

String str3= "Ali";

System.***out***.println(str3.isEmpty());

// false çünkü String boş değil

String str4=" ";

System.***out***.println(str4.isEmpty());

// false çünkü space Java için bir karakterdir.

String str5="";

System.***out***.println(str5.isEmpty());

//true verir çünkü içinde hiç bir şey yok

##### Note: isEmpty() true çıkarsa length 0 çıkar.

}

}

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** Odev1001 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan kredi kartı numarasını alın

// ve bu numaranın üçüncü, dördüncü ve sonuncu rakamlarını ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen kredi kart numaranızı giriniz");

String num1=scan.next();

System.***out***.println("Kartın üçüncü rakamı: "+num1.charAt(4));

System.***out***.println("Kartın dördüncü rakamı: "+num1.charAt(5));

System.***out***.println("Kartın son rakamı: "+num1.charAt(num1.length()-1));

scan.close();

}

}

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** Odev1002 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ismini ve soyismini alın

// isminin ve soyisminin kaç harften oluştuğunu ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

String ad, soyad;

System.***out***.println("isminizi giriniz");

ad=scan.next();

System.***out***.println("soyisminizi giriniz");

soyad=scan.next();

System.***out***.println("isminiz "+(ad.length())+" soyisminiz "+

soyad.length()+" karakterden oluşmaktadır.");

scan.close();

}

}

##### Substring(4) metodu 4.indexten itibaren sonraki kısmı keser siler. Substring(2,4) 2. ve 3. İndexi alır.

##### toLowerCase() metodu küçük harfe çevirir

##### toUpperCase() metodu büyük harfe çevirir

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** Odev1003 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ilk ve soy ismini alın

// ilk ve soy isminin ilk harflerini büyük harf olarak ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

String ad, soyad;

System.***out***.println("isminizi giriniz");

ad=scan.next();

System.***out***.println("soyisminizi giriniz");

soyad=scan.next();

System.***out***.println("Adınız: "+(ad.toUpperCase().charAt(0))+

(ad.substring(1).toLowerCase()));

System.***out***.println("Soyadınız: "+(soyad.toUpperCase().charAt(0))+

(soyad.substring(1).toLowerCase()));

scan.close();

}

}

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** Odev1004 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan yaşadığı ülkenin ismini alın

// bu ülkenin ismi Amerika ise ekrana USA,

// Ingiltere ise ekrana UK,

// Almanya ise ekrana DE yazdırın.

// Diğer ülkeler için ise NA yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Yaşadığınız ülkenin ismini yazınız");

String ulke=scan.next();

ulke=ulke.toLowerCase();

**switch**(ulke) {

**case** "amerika":

System.***out***.println("USA");

**break**;

**case**"ingiltere":

System.***out***.println("UK");

**break**;

**case**"almanya":

System.***out***.println("DE");

**break**;

**default**:

System.***out***.println("NA");

}

scan.close();

}

}

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** Odev1005 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan yaşadığı ülkenin ismini alın

// bu ülkenin ilk iki harfini büyük harf olarak ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Yaşadığınız ülkenin ismini yazınız");

String ulke=scan.next();

ulke=ulke.toUpperCase();

System.***out***.println(ulke.substring(0,2));

scan.close();

}

}

**package** day10Stringmethods;

**import** java.util.Scanner;

**public** **class** Odev1006 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan yaşadığı ülkenin ismini alın

// bu ülkenin bastan ikinci harfi ile sondan ikinci harfini

// büyük harf olarak ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Yaşadığınız ülkenin ismini yazınız");

String ulke=scan.nextLine();

// ulke=ulke.toUpperCase();

// System.out.println(ulke.substring(1,2)+ulke.substring((ulke.length()-2),ulke.length()-1));

scan.close();

// DİĞER YOL

System.***out***.println(ulke.toUpperCase().charAt(1));

System.***out***.println(ulke.toUpperCase().charAt(ulke.length()-2));

}

}

**package** day10Stringmethods;

**public** **class** Odev10den1 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki kodların ekran çıktısı alınırsa ne olur

String str="Cat, caterpillar";

System.***out***.println(str.indexOf("c"));

System.***out***.println(str.indexOf("cat"));

System.***out***.println(str.indexOf("s"));

System.***out***.println(str.indexOf("CAT"));

System.***out***.println(str.indexOf("ter"));

System.***out***.println(str.indexOf("pars"));

System.***out***.println(str.indexOf("a" , 3 ));

System.***out***.println(str.indexOf("at", 2));

System.***out***.println(str.indexOf("at", 8));

}

}

5

5

-1

-1

7

-1

6

6

-1

**package** day10Stringmethods;

**public** **class** Odev10den2 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki kodların ekran çıktısı alınırsa ne olur

String str = "Cat, caterpillar";

System.***out***.println(str.lastIndexOf("a"));

System.***out***.println(str.lastIndexOf("at"));

System.***out***.println(str.lastIndexOf("s"));

System.***out***.println(str.lastIndexOf("CAT"));

System.***out***.println(str.lastIndexOf("pars"));

System.***out***.println(str.lastIndexOf("a" , 3 ));

System.***out***.println(str.lastIndexOf("a" , 5 ));

System.***out***.println(str.lastIndexOf("at", 2));

System.***out***.println(str.lastIndexOf("at", 7));

String str1 = "";

System.***out***.println(str1.isEmpty());

String str2 = " ";

System.***out***.println(str2.isEmpty());

String str3 = "Almanya";

System.***out***.println(str3.isEmpty());

}

}

-1

1

1

1

6

true

false

false

**package** day10Stringmethods;

**public** **class** Odev10den3 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki kodların ekran çıktısı alınırsa ne olur

String str = "Cat, caterpillar";

System.***out***.println(str.toUpperCase());

System.***out***.println(str.toLowerCase());

System.***out***.println(str.substring(4).toLowerCase());

System.***out***.println(str.substring(3,8).toUpperCase());

System.***out***.println(str.toUpperCase().toLowerCase());

String strA = "Cat ";

System.***out***.println(strA.trim());

String strB = " Cat";

System.***out***.println(strB.trim());

String strC = " Cat ";

System.***out***.println(strC.trim());

String strD = "Cat";

System.***out***.println(strD.contains("a"));

String strE = " Cat";

System.***out***.println(strE.contains(" "));

String strF = "Cat";

System.***out***.println(strF.contains(""));

// String strG = "Cat";

System.***out***.println(strF.contains("d"));

}

}

CAT, CATERPİLLAR

cat, caterpillar

caterpillar

, CAT

cat, caterpillar

Cat

Cat

Cat

true

true

true

false

# 11. DERS ( STRING METHODS )

**package** day11stringsmethods;

**public** **class** StringMethod01 {

**public** **static** **void** main(String[] args) {

## startsWith() methodu bir Stringin istenen harfle başlayıp başlamadğını kontrol eder. istenen harfle başlıyorsa True, istenen harfle başlamıyorsa False return eder.

String str1="Ali Can";

// startsWith() 1. VERSİYON

System.out.println(str1.startsWith("A")); // true return etmesi lazım.

System.***out***.println(str1.startsWith("B")); // false

System.***out***.println(str1.startsWith("a")); // false

System.***out***.println(str1.startsWith("Ali")); // true

// startsWith() 2.VERSİYON

##### System.out.println(str1.startsWith("a",3));

##### 3.index ve sonrasında a varmı diye bakar varsa true yoksa false return eder.

System.***out***.println(str1.startsWith("a",5));

// 5.index ve sonrasına a varmı diye bakar varsa true der.

// true

System.***out***.println(str1.startsWith("Can",4));

## endsWith( ) methodu bir Stringin istenen harfle bitip bitmediğini kontrol eder. istenen harfle bitiyorsa true, bitmiyorsa false return eder.

String str2="Ayse Canan";

##### System.out.println(str2.endsWith("n")); // true

System.***out***.println(str2.endsWith("Canan")); // true

System.***out***.println(str2.endsWith("CANAN")); // false

System.***out***.println(str2.endsWith("")); // true verir. n den sonra içinde hiçbirşeymi var diye soruyor.

// "" ve " " tamamıyla farklıdır. ikincinin içinde boşluk karakteri var. bircinin içinde hiçbirşey yok.

String str3="Ali "; // true

System.***out***.println(str3.endsWith(" ")); // true

}

}

**package** day11stringsmethods;

**public** **class** StringMethod03 {

**public** **static** **void** main(String[] args) {

## concat() method cancatenation yapmanın yani iki stringi birleştirmenin diğer yönetmidir.str1+str2 ile aynıdır

##### concat'i istediğiniz kadar peşpeşe kullanabilirsiniz.

String str1="Ali";

String str2="Veli";

System.***out***.println(str1+str2); // AliVeli

System.***out***.println(str1.concat(str2)); // AliVeli

##### System.out.println(str1.concat(str2).concat(str1));

// AliVeliAli

System.***out***.println(str1.concat("==>").concat(str2));

// Ali==>Veli

## replace() methodu bir karakterin yerine başka bir karakter yazmaya yarar.

// örnek ata yerine ana yazmak gibi

##### String str3="ata";

System.out.println(str3.replace("t", "n")); // t gördügün yere n koy demektir.

// str3.replace(oldChar, newChar)

##### System.out.println(str3.replace("a", "ü")); // ütü yazar

##### System.out.println(str3.replace("x", "y")); // x harfi olmadığı için hiç birşey yapmaz

System.***out***.println(str3.replace("", "/")); // /a/t/a/ return eder.

##### System.out.println(str3.replace("t", "")); // aa return eder.t yi imha ediyor. bir harfi silmek için kullanılabilir.

System.***out***.println(str3.replace("at", "Mustaf")); // Mustafa

## replaceFirst() methodu değiştirmek istediğimiz karakterin sadece ilk eşleşen karakteri değiştirir.

##### String str4="karakartal";

##### System.out.println(str4.replaceFirst("k", "K")); // Karakartal

System.***out***.println(str4.replaceFirst("a", "e")); // kerekartal

System.***out***.println(str4.replaceFirst("ka", "A")); // Arakartal

System.***out***.println(str4.replaceFirst("kara", "")); // kartal

System.***out***.println(str4.replaceFirst("karak", "K")); // Kartal

System.***out***.println(str4.replaceFirst("karakartal", "cimbombom")); // cimbombom

// Soru

String text="Ali okula gitti. Ali otobüse bindi. Ali eve geldi. Ali yemek yedi. Ali uyudu.";

System.***out***.println(text.replace("Ali", "Veli"));

// Veli okula gitti. Veli otobüse bindi. Veli eve geldi. Veli yemek yedi. Veli uyudu.

}

}

**package** day11stringsmethods;

**public** **class** StringMethods02 {

**public** **static** **void** main(String[] args) {

## substring() methodu bir String'in belli bir bölümünü kesip almak için kullanılır

##### String str1="Java calışana kolaydır";

System.***out***.println(str1.substring(5));

// substring(beginIndex)

//index dahil şekilde devamını alır.

System.***out***.println(str1.substring(9)); // şana kolaydır. index 9 dahil.

##### System.out.println(str1.substring(22)); // metin 22 index ekranda hiç bir şey göremezsiniz

##### System.out.println(str1.substring(23)); // 23 olunca hata verir.

System.***out***.println(str1.substring(0)); // str1.substring=str1 olduğundan substring(0) hiç kullanılmaz

// substring() 2.VERSİYON

##### System.out.println(str1.substring(5, 13));

##### ilk index dahil ikinci index dahil değil.

// ilk indexi ekranda görürsünüz, ikinci indexi ekranda göremezsiniz. [5,13)

// çalısana return eder

System.***out***.println(str1.substring(6, 12));

// alısan

##### System.out.println(str1.substring(6, 6));// ilk "6" dahil et ikinci "6" dahil etme. ekranda birşey gözükmez.

// System.out.println(str1.substring(7, 6));// hata verir

}

}

**package** day11stringsmethods;

**public** **class** StringMethods04 {

**public** **static** **void** main(String[] args) {

// valueOf() rakamlarla oluşturulan Stringleri sayıya dönüştürür.

// böylece rakamlarla oluşturulan stringlerle matematiksel işlemler yapılabilmesini sağlar.

// Value of hem Integer wrapper classınınn içinde hemde String içinde var.

## valueOf() metodu : Integer wrapper Classının içinde olan valueof() methodu String'i integer'a çevirir.

## String'in içindeki valueof() methodu integer'i String'e çevirir.

String paraErkek="1900";

String paraKadın="2000";

// Bu ailenin toplam gelirini bulunuz

System.***out***.println(paraErkek+paraKadın); // 19002000

System.***out***.println(Integer.*valueOf*(paraErkek)+Integer.*valueOf*(paraKadın));

// maasErkek ve maasKadın sayılarını String'e dönüştürün ve concaenation yapın.

**int** maasErkek=1900;

**int** maasKadın=2000;

//3900

System.***out***.println(String.*valueOf*(maasErkek)+String.*valueOf*(maasKadın));

//19002000

}

}

**package** day11stringsmethods;

**public** **class** Day11Den02 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki kodların ekran çıktısı alınırsa ne olur diye once dusunun sonra kodları yazarak

// ekran çıktılarına bakın.

String str = "Cat, caterpillar";

System.***out***.println(str.replace('t', 'm'));

System.***out***.println(str.replace("t", "m"));

System.***out***.println(str.replace("x", "y"));

System.***out***.println(str.replace("", "|"));

System.***out***.println(str.replace("a", ""));

System.***out***.println(str.replace("at", "amel"));

System.***out***.println(str.replace("at", "e"));

System.***out***.println(str.replaceFirst("c", "C"));

System.***out***.println(str.replaceFirst("at", "amel"));

System.***out***.println(str.replaceFirst("Cat, caterpillar", "Deve"));

// Aşağıdaki kodların ekran çıktısı alınırsa ne olur diye once dusunun sonra kodları yazarak

// ekran çıktılarına bakın.

String str1 = "900";

String str2 = "1000";

**int** str3 = 900;

**int** str4 = 1000;

System.***out***.println(Integer.*valueOf*(str1) + Integer.*valueOf*(str2));

System.***out***.println(String.*valueOf*(str3) + String.*valueOf*(str4));

}

}

Cam, camerpillar

Cam, camerpillar

Cat, caterpillar

|C|a|t|,| |c|a|t|e|r|p|i|l|l|a|r|

Ct, cterpillr

Camel, camelerpillar

Ce, ceerpillar

Cat, Caterpillar

Camel, caterpillar

Deve

1900

9001000

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** Day13Odev04 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları String methodlarını kullanarak çözünüz.

// Kullanıcıdan uzunluğu çift sayı olan bir String alın

// ve bu String’in ilk yarısını ekrana yazdırın.

// Ornegin; Kullanıcı “Hollanda” girerse ekrana “Holl” yazdıracaksınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("bir kelime giriniz: ");

String kel=scan.nextLine();

**int** leng=(kel.length())/2;

System.***out***.println(kel+" kelimesinin ilk yarısı: "+kel.substring(0, leng));

scan.close();

}

}

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** Day13Odev05 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları String methodlarını kullanarak çözünüz.

// Kullanıcıdan bir String alın

// ve bu String’in ilk harfi ile son harfinin yerlerini değiştirerek

// ekrana yazdırın.

// Ornegin; Kullanıcı “Java” girerse ekrana “aavJ” yazdıracaksınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("bir kelime giriniz: ");

String kel=scan.nextLine();

**int** leng=(kel.length()-1);

**char** harfilk=kel.charAt(0);

**char** harfson=kel.charAt(leng);

System.***out***.print(kel+" kelimesinin ilk ve son harflerinin yerlerinin değişmiş"+

" şekilde yazılımı: ");

System.***out***.println(harfson+kel.substring(1, leng)+harfilk);

scan.close();

}

}

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** Day13Odev06 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları String methodlarını kullanarak çözünüz.

// Kullanıcıdan bir String alın

// ve bu String’in icinde “a” harfi varsa

// “a” harfinin ilk görünümünün indexini ekrana yazdırın.

// “a” harfi yoksa ekrana “a harfi yok” yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("bir kelime giriniz: ");

String kel=scan.next();

kel=kel.toLowerCase();

**int** index=kel.indexOf('a');

**if**(index!=-1) {

System.***out***.println("a harfinin indexi: "+index);

}**else** {

System.***out***.println("Bu kelimede a harfi bulunmamaktadır.");

}

scan.close();

}

}

# ForLoop

**package** day11stringsmethods;

**public** **class** ForLoop01 {

**public** **static** **void** main(String[] args) {

##### 6,5,4,3,2,1,0,-1 yi ekrana yazdıran bir for loop oluşturunuz.

##### 

##### for(int i=6;i>-2;i-- ) {

##### System.out.println(i);

##### }

// 3,4,5,6,7 sayılarını ekrana yazdıran for loot oluşturun

**for**(**int** i=3;i<8;i++ ) {

System.***out***.println(i);

}

// ilk 100 sayma sayısını ekrana yanyana yazdıran ve aralarına boşluk koyan programı for loop kullanarak yazınız

**for**(**int** i=1;i<101;i++) {

System.***out***.print(i+" ");

}

// ilk 100 sayma yazısını 100 den 1 e kadar yan yana bir boşluk bırakarak yazdırın

**for**(**int** i=100;i>0;i--) {

System.***out***.println("");

System.***out***.print(i+" ");

}

// ilk 50 çift sayma sayısın ekrana yazdıran for loop oluşturun

**for** (**int** i=2; i<101;i=i+2) {

System.***out***.println(i);

}

// ilk 50 tek sayma sayılarını ekrana yan yana yazdıran for loop oluşturun.

**for** (**int** i=1; i<100;i+=2) {

System.***out***.print(i+" ");

}

}

}

**package** day11stringsmethods;

**public** **class** ForLoop02 {

**public** **static** **void** main(String[] args) {

// 100 den küçük 3 ün katı olan sayma sayılarını ekrana yan yana yazdıran for loop oluşturunuz

**for**(**int** i=3;i<100;i+=3) {

System.***out***.print(i+" ");

}

System.***out***.println();

##### ilk 5 sayma sayısının toplamını veren programı for loop ile yazınız.

// 1,2,3,4,5

// int sum=0 0+1=1 ==> 1+2=3 ==> 3+3=6 ==> 6+4=10 ==> 10=5=15

// 1+2+3+4+5

##### int sum=0;

##### for(int i=1;i<6;i++) {

##### sum=sum+i;

##### }

##### System.out.println("Toplam:"+sum);

##### System.out.println();

// 10 dan büyük ilk 3 sayma sayısının toplamını veren for loopu yazınız

**int** sum1=0;

**for**(**int** i=11;i<14;i++) {

sum1=sum1+i;

}

System.***out***.println("Toplam:"+sum1);

System.***out***.println();

// 20 den büyük ilk 5 sayma sayısının toplamını veren for loop yazınız

**int** sum3=0;

**for**(**int** i=21; i<26;i++) {

sum3=sum3+i;

}

System.***out***.println("Toplam:"+sum3);

System.***out***.println();

// ilk 50 sayma sayısının toplamını veren for loop oluşturunuz

**int** sum4=0;

**for**(**int** i=1;i<51;i++) {

sum4=sum4+i;

System.***out***.println();

}

System.***out***.println("Toplam:"+sum4);

// 5 ile bölünebilen 100 den küçük sayma sayılarının toplamını veren for loop oluşturunuz

**int** sum5=0;

**for**(**int** i=5;i<100;i+=5) {

sum5=sum5+i;

}

System.***out***.println("Toplam:"+sum5);

}

}

**package** day11stringsmethods;

**public** **class** ForLoop03 {

**public** **static** **void** main(String[] args) {

// ilk 4 sayma sayısının çarpımını veren for loop oluşturunuz

**int** product=1;

**for**(**int** i=1;i<5;i++) {

product=product\*i;

}

System.***out***.println("Çarpım:"+product);

System.***out***.println();

// ilk 5 çift sayma sayısının çarpımını veren for loop oluşturunuz

**int** product2=1;

**for**(**int** i=2;i<11;i+=2) {

product2=product2\*i;

}

System.***out***.println("Çarpım:"+product2);

System.***out***.println();

// Matematikte 1 er 1 er geri sayım yapıp çarpmaya faktoriyel denir.

// 6!=6\*5\*4\*3\*2\*1

// interview sorusu

##### 9! i hesaplayan for loop yazınız

##### int product3=1;

##### for(int i=1;i<10;i++) {

##### product3=product3\*i;

##### }

##### System.out.println("9!="+product3);

##### System.out.println();

##### }

}

**package** day11stringsmethods;

**import** java.util.Scanner;

**public** **class** Day11Odev02 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden veya sonrasından başlayıp bitiş değerinde veya öncesinde biten

// tüm çift tamsayıları ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir başlangıç değeri giriniz");

**int** ilk =scan.nextInt();

System.***out***.println("Bitiş değeri giriniz");

**int** son =scan.nextInt();

**for**(**int** i=ilk; i<=son; i++) {

**if**(i%2==0) {

System.***out***.println(i);

}}

scan.close();

}

}

**package** day11stringsmethods;

**import** java.util.Scanner;

**public** **class** Day11Odev06 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş harflerini alın

// ve başlangıç harfinden başlayıp bitiş harfinde biten

// tüm harfleri büyük harf olarak ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir başlangıç harfi giriniz");

**char** harf1=scan.next().toUpperCase().charAt(0);

System.***out***.println("Bir bitiş harf yazınız");

**char** harf2=scan.next().toUpperCase().charAt(0);

System.***out***.println(harf1+" - "+ harf2);

**if**(harf1>=harf2) {

System.***out***.println("Başlangıç harfi bitiş harfinden alfabeye göre önce geliyor olması lazım");

System.***out***.println("Lütfen yeniden giriş yapınız");

}**else** **if**((harf1<='z' && harf1>='a') && (harf2>='a'&&harf2<='z')){

System.***out***.println("lütfen harf giriniz");

}

**else** {

**for**(**char** h=harf1 ; h<=harf2 ; h++) {

System.***out***.print(h+" ");

}}

scan.close();

}

}

# 12. DERS ( SCOPE KURALLARI – INSTANCE VARIABLE – LOCAL VARIABLE )

**package** day12scopewhileloop;

**public** **class** scopeKurallari {

**int** num; // Class variable veya Instance variable

**public** **static** **void** main(String[] args) {

// Scope kapsam demektir. oluşturduğunuz bir variable'i nerelerde kullanabilceğinizi

// gösterir.

// Class'ın içinde bir variable oluşturduğumuz zaman tüm methodlar o variable ulaşabilir.

// yeterki methodun içinde olmasın, Class'ın içinde olsun.

}

**public** **void** toplama() {

// int num=12;

// System.out.println("Toplama");

}

**public** **void** carpma(**int** num, String str) {

// System.out.println("Çarpma");

}

}

## 1. KURAL

##### Class'ın içinde metodların dışında oluşturulan variable'ları bütün metodlar istedikleri zaman istedikleri kadar kullanabilirler. Bu variable'lara "Class Variable" veya "Instance Variable" denir. Instance: görünür demektir. Class variable'lara değer ataması yapmazsak Java onlara default değerler verir. Java tüm sayılar için default olarak 0 değerini verir.

## 2.KURAL

##### Bir metodun içerisinde oluşturulan variable'lar sadece o metodun içinde kullanılabilir. O metodun dışındaki metodlar o variable'i kullanamazlar. Metodun içinde oluşturulan bu variable'lara "Local Variable" denir. Body içerisinde oluşturulan variable'lere değer atamak ZORUNDAYIZ. Java onlara default value vermez.

## 3.KURAL

##### Metodların parantezleri içerisinde oluşturulan variable'lar Local variable'lar gibi sadece o metodların bodyleri içinde kullanılabilir. Bu variable'lara "Local Variable" denir. Parantez içinde oluşturulan variable'lara ise değer ataması yapmayınız.

## 4.KURAL

##### Java yukarıdan aşağı çalışır. value'nin önce tipini tanımlamyıp, ardından değer ataması yapılmalıdır yoksa hata verir.

// Bir variable'i kullanmaya başlamadan önce tanımlamalıyız.

// önce num=num+2; sonra int num=12; diyemeyiz.

// int num=12; sonra num=num+2 demeliyiz.

# WHILE LOOP

**package** day12scopewhileloop;

**public** **class** WhileLoop01 {

**public** **static** **void** main(String[] args) {

##### While => iken demektir.

// While I go to school ==> Ben okula giderken.

// Whie I go to Üsküdür, it rained ==> üsküdara giderken yağmur yağdı.

##### () parantez içindeki koşula bakar true ise { } içindekini return eder döner tekrar while parantezine gider.

##### num 4ten küçük iken body'yi çalıştır.

##### int num=0;

##### while(num<4) {

##### System.out.println("Ali");

##### num++; // Bu satırı unutursanız while loop sonsuz döngüye girer.program kilitlenir.

##### }

// While loop kullanarak 1 den 10a kadar tam sayıları ekrana yazdırınız

**int** num1=1;

**while**(num1<11) {

System.***out***.println(num1);

num1++;

}

// while loop kullanarak 1 den 20 ye kadar çift tam sayıları ekrana yazdırınız

**int** num2=1;

**while**(num2<21) {

**if**(num2%2==0) {

System.***out***.print(num2+" ");}

num2++;

}

System.***out***.println(" ");

// while loop kullanarak 5 ten 120 ye kadar 3 ile bölünebilen tam sayıları yazdırınız

**int** num3=5;

**while**(num3<121) {

**if**(num3%3==0) {

System.***out***.print(num3+" ");}

num3++;

}

}

}

**package** day12scopewhileloop;

**public** **class** Day12Odev01 {

**public** **static** **void** main(String[] args) {

//While loop kullanarak 3 den 13 e kadar tum tek tamsayıları ekrana yazdırınız.

**int** i=3;

**while**(i<14) {

**if**(i%3==0) {

System.***out***.print(i+" ");

} i++;

}

}

}

**package** day12scopewhileloop;

**import** java.util.Scanner;

**public** **class** Day12Odev02 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden başlayıp bitiş değerinde

// biten tüm tamsayıları while loop kullanarak ekrana yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen bir başlangıç değeri giriniz");

**int** bas=scan.nextInt();

System.***out***.println("Lütfen bir bitiş değeri giriniz");

**int** bit=scan.nextInt();

**if**(bas<=bit) {

**while**(bas<=bit) {

System.***out***.print(bas+" ");

bas++;

}

}**else** **while**(bit<=bas) {

System.***out***.println(bas+" ");

bas--;

}

scan.close();

}

}

**package** day12scopewhileloop;

**import** java.util.Scanner;

**public** **class** Day12Odev03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden veya sonrasından başlayıp bitiş değerinde

// veya öncesinde biten tüm çift tamsayıları while loop kullanarak

// ekrana yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen bir başlangıç değeri giriniz");

**int** bas=scan.nextInt();

System.***out***.println("Lütfen bir bitiş değeri giriniz");

**int** bit=scan.nextInt();

**if**(bas<=bit) {

**while**(bas<=bit) {

**if**(bas%2==0) {

System.***out***.print(bas+" ");}

bas++;

}

}**else** **while**(bit<=bas) {

**if**(bas%2==0) {

System.***out***.println(bas+" ");}

bas--;

}

scan.close();

}

}

**package** day12scopewhileloop;

**import** java.util.Scanner;

**public** **class** Day12Odev04 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden başlayıp bitiş değerinde biten

// tüm tamsayıların toplamını ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen bir başlangıç değeri giriniz");

**int** bas=scan.nextInt();

System.***out***.println("Lütfen bir bitiş değeri giriniz");

**int** bit=scan.nextInt();

**int** sum=0;

**if**(bas<=bit){

**while**(bas<=bit) {

sum=bas+sum;

bas++;

}

}**else** **while**(bit<=bas) {

sum=bas+sum;

bas--;

}System.***out***.print("iki sayı arlalığındaki sayıların toplamı="+sum);

scan.close();

}

}

**package** day12scopewhileloop;

**import** java.util.Scanner;

**public** **class** Day12Odev05 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden başlayıp bitiş değerinde biten

// tüm tamsayıların çarpımını ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen bir başlangıç değeri giriniz");

**int** bas=scan.nextInt();

System.***out***.println("Lütfen bir bitiş değeri giriniz");

**int** bit=scan.nextInt();

**int** sum=1;

**while**(bas<=bit) {

sum=bas\*sum;

bas++;

} System.***out***.print("iki sayı arlalığındaki sayıların çarpımı="+sum);

scan.close();

}

}

**package** day12scopewhileloop;

**import** java.util.Scanner;

**public** **class** Day12Odev06 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan başlangıç ve bitiş harflerini alın

// ve başlangıç harfinden başlayıp bitiş harfinde biten

// tüm harfleri büyük harf olarak ekrana yazdırın.

// Kullanıcının hata yapmadığını farz edin.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen bir başlangıç harfi giriniz");

**char** bas=scan.next().toUpperCase().charAt(0);

System.***out***.println("Lütfen bir bitiş harfi giriniz");

**char** bit=scan.next().toUpperCase().charAt(0);

**while**(bas<=bit) {

System.***out***.print(bas+" ");

bas++;

}

scan.close();

}

}

**package** day12scopewhileloop;

**import** java.util.Scanner;

**public** **class** Day12Odev07 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir rakam alın ve bu rakam icin çarpım tablosunu ekrana yazdırın.

// Kullanıcının hata yapmadığını farz edin.

// Ornegin kullanıcı 3 girerse;

// 3x1=3 3x2=6 3x3=9 3x4=12 3x5=15 3x6=18 3x7=21 3x8=24 3x9=27 3x10=30

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Çarpım tablosunu istediğiniz rakamı giriniz");

**int** rak=scan.nextInt();

System.***out***.println("For Loop ile çözüm");

**for**(**int** i=1; i<11;i++) {

System.***out***.println(rak+"x"+i+"="+rak\*i);

}

System.***out***.println("");

System.***out***.println("While Loop ile çözüm");

**int** num=1;

**while**(num<11) {

System.***out***.println(rak+"x"+num+"="+rak\*num);

num++;

}

scan.close();

}

}

# 13. DERS ( WHILE LOOPS )

**package** day13whiledowhileloops;

**public** **class** WhielLoop01 {

**public** **static** **void** main(String[] args) {

// While loop kullanarak ilk 5 sayma sayısının toplamını

// ekrana yazdıran programı yazınız

**int** sum=0;

**int** i=1;

**while**(i<6) {

sum=sum+i;

i++;

}System.***out***.println("Toplam="+sum);

}

}

**package** day13whiledowhileloops;

**public** **class** Day13Odev01 {

**public** **static** **void** main(String[] args) {

//Aşağıdaki soruları hem for loop hem de while loop kullanarak çözünüz

// 8 den 180 e kadar 5 in kati olan tum tamsayıları ekrana yazdırınız.

System.***out***.println("For Loop ile çözüm:");

**for**(**int** i=8; i<181;i++) {

**if**(i%5==0) {

System.***out***.print(i+" ");

}

}

System.***out***.println(" ");

System.***out***.println(" ");

System.***out***.println("While Loop ile çözüm");

**int** i=8;

**while**(i<181) {

**if**(i%5==0) {

System.***out***.print(i+" ");

}

i++;

}

}

}

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** Day13Odev02 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları hem for loop hem de while loop kullanarak çözünüz.

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden başlayıp bitiş değerinde biten

// 3 ve 5 ile bölünebilen tüm tamsayıları ekrana yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen bir başlangıç değeri giriniz");

**int** bas=scan.nextInt();

System.***out***.println("Lütfen bir bitiş değeri giriniz");

**int** bit=scan.nextInt();

**int** bas1,bit1;

**if**(bas<=bit) {

bas1=bas;

bit1=bit;

}**else** {

bas1=bit;

bit1=bas;

}

System.***out***.println("For Loop ile çözüm");

System.***out***.print(bas1+" ile "+bit1+" sayı aralığında 3 ve 5'e bölünebilen sayılar: ");

**for**(**int** i=bas1;i<=bit1;i++) {

**if**(i%3==0 && i%5==0) {

System.***out***.print(i+" ");

}

}

System.***out***.println(" ");

System.***out***.println(" ");

System.***out***.println("For While ile çözüm");

System.***out***.print(bas1+" ile "+bit1+" sayı aralığında 3 ve 5'e bölünebilen sayılar: ");

**while**(bas1<=bit1) {

**if**(bas1%3==0 && bas1%5==0) {

System.***out***.print(bas1+" ");

}

bas1++;

}

scan.close();

}

}

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** Day13Odev03 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları hem for loop hem de while loop kullanarak çözünüz.

// Kullanıcıdan bir sayı alın ve bu sayının bölenlerini ekrana yazdırınız.

// Ornegin; 12'nin bölenleri 1, 2, 3, 4, 6, 12 dir.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bölenlerini bulmak istediğiniz bir tam sayı giriniz");

**int** num=scan.nextInt();

**if**(num<0) {

num=num\*(-1);

}

System.***out***.println("For Loop ile çözüm");

System.***out***.print(num+" sayısının bölenleri: ");

**for**(**int** i=1;i<=num;i++) {

**if**(num%i==0) {

System.***out***.print(i+" ");

}

}

System.***out***.println(" ");

System.***out***.println(" ");

System.***out***.println("For While ile çözüm");

System.***out***.print(num+" sayısının bölenleri: ");

**int** i=1;

**while**(i<=num) {

**if**(num%i<=0) {

System.***out***.print(i+" ");

}

i++;

}

scan.close();

}

}

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** ForLoopinterview01 {

**public** **static** **void** main(String[] args) {

## INTERVİEWLERDE ÇOK SORULAN BİR SORU : Kullanıcıdan bir string alın ve bu stringi ekrana tersten yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir cümle veya kelime giriniz");

String str=scan.nextLine();

**for**(**int** i=str.length()-1; i>=0;i--) {

System.***out***.print(str.charAt(i));

}

scan.close();

}

}

**package** day13whiledowhileloops;

**import** java.util.Scanner;

**public** **class** ForLoopinterview03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir sayı alın

// bu sayının tersini ekrana yazdırın

// bu sayının tersinin iki fazlasını ekrana yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir sayı giriniz");

## char'i String'e cevirmek icin sonuna + "" ekleyebilirsiniz

##### String num=scan.nextLine();

String ters="";

**for**(**int** i=num.length()-1; i>=0;i--) {

##### ters=ters+num.charAt(i)+"";

}

**int** tersInt=Integer.*valueOf*(ters)+2;

System.***out***.println(tersInt);

scan.close();

}

}

# 14. DERS ( DO WHILE LOOP )

**package** day14dowhileloop;

**public** **class** DoWhile01 {

**public** **static** **void** main(String[] args) {

**int** num=1;

**do** {

System.***out***.println(num);

num++;

}**while**(num<4);

// 10 ile 20 arasındaki tek tam sayıları

// ekrana do while loop ile yazdırın

##### int num1=11;

##### do {

##### System.out.println(num1);

##### num1+=2;

##### }while(num1<20);

}

}

**package** day14dowhileloop;

**public** **class** DoWhile02 {

**public** **static** **void** main(String[] args) {

// do-while ile while loop'larının farkı

##### do-while herhalükarda bir kere çalışır ama while koşulu taşımıyorsa çalışmaz. while de condition önce kontrol edilir sonra iş yapılır.do-while önce çalışır sonra kontrol edilir.

**int** num=7;

**while** (num<7) {

System.***out***.println("While: "+num);// çalışmaz

num++;

}

**int** num1=7;

**do** {

System.***out***.println("Do while: "+num1);// en az bir kere çalışır.

num1++;

}**while**(num1<7);

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** DoWhile03 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir sayı alınız.

// sayı 10 dan küçük ise ekrana kazandınız yazdırın.

// sayı 10 veya 10 dan büyük ise ekrana tekrar deneyiniz yazdırın

Scanner scan=**new** Scanner(System.***in***);

**int** num=0;

**do** {

System.***out***.println("Bir sayı giriniz");

num=scan.nextInt();

}**while**(num>=10);

##### while içindeki condition true oldukça loop devam eder.false olunca loop biter alta geçer.

System.***out***.println("Kazandınız");

scan.close();

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** DoWhile04 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan bir String alın

// bu stringin indexi tek sayı olan karakterlerini ekrana yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir kelime giriniz");

String kel=scan.next();

**int** length=kel.length();

**int** i=1;

**do** {

**if**(i%2==1) {

System.***out***.print(kel.charAt(i)+" ");

}

i++;

}**while**(i<length);

scan.close();

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** DoWhile05 {

**public** **static** **void** main(String[] args) {

// Kullanıcıdan ismini alın

// ilk harf büyük harf ise ekrana doğru giriş yaptınız

// ilk harf küçük harf ise ilk harfi büyük olarak giriniz

Scanner scan=**new** Scanner(System.***in***);

String isim=""; // sayı olunca sıfır string olunca "" kullanabiliriz.

**do** {

System.***out***.println("Lütfen ilk isminizi giriniz");

isim=scan.next();

**if**(isim.charAt(0)>='A'&&isim.charAt(0)<='Z') {

System.***out***.println("Doğru giriş yaptınız");

}**else** {

System.***out***.println("İlk harfi büyük harf olarak giriniz.");

}

}**while**(!(isim.charAt(0)>='A'&&isim.charAt(0)<='Z')); // ! değildir

scan.close();

}

}

**package** day14dowhileloop;

**public** **class** Day14Odev01 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları do-while loop kullanarak çözünüz.

// 9 den 190 e kadar 7 nin kati olan tüm tamsayıları ekrana yazdırınız.

**int** num=9;

System.***out***.print("9'dan 190'a kadar 7'nin katları olan sayılar: ");

**do** {

**if**(num%7==0) {

System.***out***.print(num+" ");

}

num++;

}**while**(num<90);

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** Day14Odev02 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları do-while loop kullanarak çözünüz.

// Kullanıcıdan başlangıç ve bitiş değerlerini alın

// ve başlangıç değerinden başlayıp bitiş değerinde biten

// 4 ve 6 ile bölünebilen tüm tamsayıları ekrana yazdırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir başlangıç değeri giriniz");

**int** bas=scan.nextInt();

System.***out***.println("Bir bitiş değeri giriniz");

**int** bit=scan.nextInt();

**if**(bas>bit) {

bit=bit+bas;

bas=bit-bas;

bit=bit-bas;

}

System.***out***.print(bas+" - "+bit+" sayı aralığında 4 ve 6 ya bölünebilen sayılar: ");

**do** {

**if**(bas%4==0 && bas%6==0) {

System.***out***.print(bas+" ");

}

bas++;

}**while**(bas<bit);

scan.close();

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** Day14Odev03 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları do-while loop kullanarak çözünüz.

// Kullanıcıya sayı girmesini söyleyin.

// Kullanıcı sıfır girdiğinde, ekrana o ana kadar girmiş olduğu

// tüm sayıların toplamını yazdırınız.

// Ornegin; 5, 4, 7, 0 ise 5+4+7=16 oldugundan ekrana 16 yazdırın

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Toplamasını istediğiniz sayıları teker teker giriniz.");

System.***out***.println("Toplam sonucu istediğinizde en son \"0\" yazınız");

**int** num;

**int** sum=0;

**do** {

System.***out***.println("Bir sayı giriniz: ");

num=scan.nextInt();

sum=sum+num;

}**while**(num!=0);

System.***out***.println("Girilen sayıların toplamı: "+sum);

scan.close();

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** Day14Odev04 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları do-while loop kullanarak çözünüz.

// Kullanıcıya iki sayı girmesini söyleyin.

// Bu sayılar eşit ise ekrana “Kare oluşturdunuz” yazdırın.

// Eşit değilse tekrar iki sayı girmesini söyleyin.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("İki tam sayı giriniz");

**int** a,b;

**do** {

a=scan.nextInt();

b=scan.nextInt();

**if**(a!=b) {

System.***out***.println("Tekrar iki tam sayı giriniz");

}

}**while**(a!=b);

System.***out***.println("Kare oluşturdunuz");

scan.close();

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** Day14Odev05 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları do-while loop kullanarak çözünüz.

// Kullanıcıya bir String girmesini söyleyin

// ve bu String’in ilk harfi ile son harfi aynı ise

// ekrana “Simetrik” yazdırın.

// Aynı değilse tekrar bir String girmesini isteyin.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir kelime yazınız");

String wort = "";

**char** ilk,son;

**do** {

wort=scan.next();

ilk=wort.charAt(0);

son=wort.charAt(wort.length()-(1));

**if**(ilk!=son) {

System.***out***.println("Yeni bir kelime daha yazınız");

}

}**while**(ilk!=son);

System.***out***.println("Simetrik");

scan.close();

}

}

**package** day14dowhileloop;

**import** java.util.Scanner;

**public** **class** Day14Odev06 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları do-while loop kullanarak çözünüz.

// Kullanıcıdan bir String alın

// ve bu String’in icinde “a” harfi varsa ekrana “a harfi var” yazdırın.

// “a” harfi yoksa tekrar bir String girmesini isteyin.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir kelime yazınız");

String wort;

**do** {

wort=scan.next();

**if**(!(wort.contains("a"))) {

System.***out***.println("Yeni bir kelime daha yazınız");

}

}**while**(!(wort.contains("a")));

System.***out***.println("Simetrik");

scan.close();

}

}

# 15. DERS ( METHOD CREATİON )

**package** day15methodcreation;

**public** **class** MethodCreation01 {

**public** **static** **void** main(String[] args) {

##### Genel usul olarak metodlar main method içine yazılmaz.main method dışında methodlar oluşturulur ve oluşturlan methodlar main method içine çağırılarak kullanılır.

// metodları main'in dışında oluşturup, mainin içinde kullanabiliriz.

// metodun ismini yazarak çağırıp main içinde kullanabiliriz.

// main metodunu sadeleştirmeye yarar.

// metodu call ediyoruz çağırıyoruz.

##### Bir metodu main içerisinden çağıracaksanız mutlaka static kelimesini kullanmalısınız.

##### void bir return type dir.hiç bir şey üretmez sadece ekrana birşey yazar.

##### Method parantezinin içinde oluşturulan variable'lara "parametre" denir.Methodu çağırırken method parantezinin içine yazılan değerlere "argüment" denir.Parametrenin data type ile argümentin data type'ı eşleşmeli. parametre int iken argüment String olamaz.

// toplama();

// carpma();

System.***out***.println(*toplama*(11,15.2));// 26.2

System.***out***.println(*carpma*(13,5)); // 65

}

// public static void toplama() {

// System.out.println(3+5);

// }

**public** **static** **double** toplama(**int** num1,**double** num2) { // data tipi ve ismi birlikte yazılan yere parametre denir.

// toplama metodu iki parametrelidir.

// double int'i kapsadığı için

// sorun çıkarmıyor. auto // eğer parametreli bir metod oluşturursak metodu çağırmak için

// içine sayı koymamız gerekiyor.

// toplama(3,5) gibi 3 ve 5'e argüment deniyor.

**return** num1+num2; // return: döndürmek, sonucu vermek

}

// public static void carpma() {

// System.out.println(3\*5);

// }

##### public static int carpma(int num1,int num2) {

##### return num1\*num2;

##### }

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** MethodCreation02 {

**public** **static** **void** main(String[] args) {

// saati saniyeye, mili kilometreye, kg'mı grama

// çeviren bir method yazınız.

// bu methodu main methodun dışında oluşturup main methodun içinden çağırınız.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Birimi giriniz");

String birim=scan.nextLine();

System.***out***.println("Miktarı giriniz");

**double** miktar=scan.nextDouble();

*donusturucu*(birim,miktar); // 10800

scan.close();

}

**public** **static** **void** donusturucu(String birim,**double** miktar) {

// switch: değişmek demek

**switch**(birim) {

**case**"saat":

System.***out***.println(miktar\*60\*60);

**break**;

**case**"mil":

System.***out***.println(miktar\*1.6);

**break**;

**case**"kilogram":

System.***out***.println(miktar\*1000);

**break**;

**default**:

System.***out***.println("saat,mil ve kilogram dışında bir birim girmeyiniz");

}

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** MethodCreation03 {

**public** **static** **void** main(String[] args) {

// Basit bir hesap makinasi

// Kullanıcıdan yapacağı işlemi işlem sembolü ile seçmesini sağlayalım.

// sonrada kullanıcıdan iki sayı girmesini isteyelim.

// Girilen iki sayı ve seçilen işleme göre doğru sonucu ekrana yazdıralım.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("\" + \" \" - \" \" \* \"\" : \" işlemlerinden birini seçiniz");

**char** islem=scan.next().charAt(0);

System.***out***.println("işlem yapmak için iki sayı giriniz");

**double** num1=scan.nextDouble();

**double** num2=scan.nextDouble();

*hesapMakinasi*(islem,num1,num2);

scan.close();

}

**public** **static** **void** hesapMakinasi(**char** islem,**double** num1,**double** num2) {

// case: durum 4 farklı durum var switch kullanmak makul

**switch**(islem) {

**case**'+':

System.***out***.println(num1 +"+"+num2+"="+ (num1+num2));

**break**;

**case**'-':

System.***out***.println(num1 +"-"+num2+"="+ (num1-num2));

**break**;

**case**'\*':

System.***out***.println(num1 +"\*"+num2+"="+ (num1\*num2));

**break**;

**case**':':

System.***out***.println(num1 +":"+num2+"="+ (num1/num2));

**break**;

**default**:

System.***out***.println("\"+\" \"-\" \"\*\"\":\" : dışında işlem girmeyiniz");

}

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** Day15Odev01 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları çözerken metodları main methodun dışında oluşturup main methodun icinden çağırınız.

// Kullanıcıdan isim ve soyismini alın.

// Kullanıcının isim ve soyisminin ilk harflerini büyük

// diğer harflerini küçük harf olarak ekrana yazdıran programı yazınız.

String soyisim = **null**;

String isim = **null**;

*isimYazdirma*(isim,soyisim);

}

**public** **static** **void** isimYazdirma(String isim, String soyisim) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Lütfen isminizi giriniz");

isim=scan.nextLine();

isim=isim.toLowerCase();

**char** ilk=isim.toUpperCase().charAt(0);

System.***out***.println("Lütfen soyisiminizi giriniz");

soyisim=scan.next();

soyisim=soyisim.toLowerCase();

**char** soyilk=soyisim.toUpperCase().charAt(0);

System.***out***.print("İsim soyisim: ");

**if**(isim.contains(" ")) {

String ilkİsim=ilk+isim.substring(1,(isim.indexOf(" ")));

String ikinciİsim=isim.substring((isim.indexOf(" "))+1).toLowerCase();

**char** ilk2=ikinciİsim.toUpperCase().charAt(0);

isim=ilkİsim+" "+ilk2+ikinciİsim.substring(1);

System.***out***.print(isim);

}**else** {

System.***out***.print(ilk+isim.substring(1));

}

System.***out***.println(" "+soyilk+soyisim.substring(1));

scan.close();

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** Day15Odev02 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları çözerken metodları main methodun dışında oluşturup main methodun icinden çağırınız.

// Kullanıcıdan iki sayı alın

// bu sayılar eşit ise ekrana “Eşit”,

// birinci sayı büyük ise ekrana “Birinci sayı büyük”,

// ikinci sayı büyük ise ekrana “Ikinci sayı büyük” yazdırınız.

**int** num1=0;

**int** num2=0;

*sayiEsitligi*(num1,num2);

}

**public** **static** **void** sayiEsitligi(**int** num1,**int** num2) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("İki sayı giriniz");

num1=scan.nextInt();

num2=scan.nextInt();

String result= num1==num2 ? "Bu iki sayı eşittir" :((num1>num2)? "Birinci sayı ikinci sayıdan büyüktür" :

"ikinci sayı birinci sayıdan büyüktür");

System.***out***.println(result);

scan.close();

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** Day15Odev03 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları çözerken metodları main methodun dışında oluşturup main methodun icinden çağırınız.

// Kullanıcıya sayı girmesini söyleyin.

// Kullanıcının girdiği sayının rakamları toplamını ekrana yazdıran bir program yazın.

**int** num=0;

*rakamToplami*(num);

}

**public** **static** **void** rakamToplami(**int** num) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Bir tam sayı giriniz");

num=scan.nextInt();

**int** sum=0;

**while**(num>0) {

**int** bas=num%10;

sum=sum+bas;

num=num/10;

}

System.***out***.println("Sayının rakamları toplamı="+sum);

scan.close();

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** Day15Odev04 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları çözerken metodları main methodun dışında oluşturup main methodun icinden çağırınız.

// Kullanıcıya paralelkenar, dikdörtgen ve üçgen kelimelerinden birini ve iki sayı seçmesini söyleyin.

// Hangi şekli seçerse, o şeklin alanını ve çevresini ekrana yazdıran programı yazınız.

String sekil="";

*alanCevre*(sekil);

}

**public** **static** **void** alanCevre(String sekil) {

**double** a,b,c,h,sAlan,sCevre;

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Alan ve çevresini hesaplamak istediğiniz şekli seçiniz.");

System.***out***.println("Paralel kenar için: P");

System.***out***.println("Dikdörtgen için: D");

System.***out***.println("Üçgen için: U yazınız.");

sekil=scan.next().toLowerCase();

**switch** (sekil) {

**case** "p":

System.***out***.println("Paralelkenarın taban uzunluğunu (a) giriniz :");

a=scan.nextDouble();

System.***out***.println("Paralelkenarın kısa kenar uzunluğunu(b) giriniz : ");

b=scan.nextDouble();

System.***out***.println("Paralelkenarın yüksekliğini(c) giriniz :");

c=scan.nextDouble();

sAlan=b\*c;

sCevre=2\*a+2\*b;

System.***out***.println("Paralelkenarın alanı :"+sAlan+" cm2");

System.***out***.println("Paralelkenarın çevresi :"+sCevre+ " cm");

**break**;

**case** "d":

System.***out***.println("Dikdörtgenin a kenar uzunluğunu giriniz :");

a=scan.nextDouble();

System.***out***.println("Dikdörtgenin b kenar uzunluğunu giriniz : ");

b=scan.nextDouble();

sAlan=a\*b;

sCevre=2\*a+2\*b;

System.***out***.println("Dikdörtgenin alanı :"+sAlan+" cm2");

System.***out***.println("Dikdörtgenin çevresi :"+sCevre+ " cm");

**break**;

**case** "u":

System.***out***.println("Üçgenin a taban uzunluğunu giriniz :");

a=scan.nextDouble();

System.***out***.println("Üçgenin b kenarının uzunluğunu girin : ");

b=scan.nextDouble();

System.***out***.println("Üçgenin c kenarının uzunluğunu girin :");

c=scan.nextDouble();

System.***out***.println("Üçgenin h yüksekliğini giriniz :");

h=scan.nextDouble();

sAlan=a\*h/2;

sCevre=a+b+c;

System.***out***.println("Üçgenin alanı :"+sAlan+" cm2");

System.***out***.println("Üçgenin çevresi :"+sCevre+ " cm");

**break**;

**default**:

System.***out***.println("Yanlış karakter girdiniz tekrar deneyiniz");

scan.close();

}

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** Day15Odev05 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki soruları çözerken metodları main methodun dışında oluşturup main methodun icinden çağırınız.

// Kullanıcıya bir String girmesini söyleyin

// ve bu String’i yukarıdan aşağıya doğru yazdıran Program yazınız.

// Ornegin; CAN ==> C

// A

// N

String klm="";

*yukAsagi*(klm);

}

**public** **static** **void** yukAsagi(String klm) {

**int** leng;

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Yukarıdan aşağıya alt alta yazılmasını istediğiniz bir kelime giriniz");

klm=scan.next();

leng=klm.length()-(1);

**for**(**int** i=0; i<=leng;i++ ) {

System.***out***.println(klm.charAt(i));

}

scan.close();

}

}

**package** day15methodcreation;

**import** java.util.Scanner;

**public** **class** day15Odev06 {

**public** **static** **void** main(String[] args) {

// 1'den 9'a kadar rastgele atanan bir sayının kaç girişte tahmin edildiğini gösteren bir oyun yazınız.

Scanner scan=**new** Scanner(System.***in***);

**int** ran1=(**int**) (Math.*random*()\*10);

**int** num1=0;

**int** i=1;

System.***out***.println("1'den 9'a kadar Sayı tahmini oyunu");

System.***out***.println("Sayımızı kaç girişte tahmin edebileceksiniz hadi deneyelim!!");

System.***out***.println("1-9 arası bir sayı giriniz");

**do**{

num1=scan.nextInt();

**if**(num1>9) {

System.***out***.println("Yanlış sayı girdiniz tekrar deniyiniz");

}

**if** (num1==ran1) {

System.***out***.println("Tebrikler!!! "+num1+" sayısını "+i+ ". denemede bildiniz.");

i=9;

}

**else** **if**(i==9) {

System.***out***.println("Maalesef "+ ran1+" sayısını bilemediniz");

}

**else** **if**(i<9) {

System.***out***.println("1-9 arası bir sayı giriniz");

}

i++;

}**while**(i!=10);

scan.close();

}

}

# 16. DERS ( CONSTRUCTOR, OVERLOADİNG )

# CONSTRUCTOR

**package** day16overloading;

**public** **class** Constructor01 {

Objeckt nasıl üretilir:

// Consructor: inşaatçı, usta, yapıcı, kalıp demektir.

##### Consructor Classtan object oluşturmaya yarar.Bir method değildir, variable değildir,bir kod parçasıdır.Constructor, Class'ın body'si içinde olmalıdır.Constructor'ın ismi Class'ın ismi ile aynı olmalıdır.Constructor'larda return type(int,String) olmaz.Consructori hangi Classta ürettiğimi unutursam cmd tuşana basıp constructor ismine tıklarsam üretildiği class'a götürüyor.

// aşağıda objekt oluşturup default Constructor kullanıyoruz.

**int** price = 20000;

**int** year= 2020;

String make="Honda";

String type="Civic";

**public** **static** **void** main(String[] args) {

##### Constructor01 hondaAraba=new Constructor01();

// bir tane objeckt üretildi.

##### İsimden önce data type yazılır. int, String gibi. bu primitive data typlerdir, bunları java üretmiştir.yukarıdaki Constructor01 de bir data typedir. non primitive dir. biz üretmiş olduk.Bu şekilde istenildiği kadar data type üretilebilir.

##### new objekct üretmeye yarar. ne zaman objeckt üretmek istersek new yazmamız gerekiyor. Bir yerda new görüyorsak orada objekt üretilmiş demektir.

##### Constructor01() en sondaki bu kelime javaya bu kalıpla bir objekt üret demektir.hondaAraba yazıp nokta koyduğumuzda seçenekleri görürüz.bir insan Class oluşturuyorsa Objeckt üretmek içindır.

##### Java Classla beraber bir constructor oluşturur.Buna "default constructor" denir. default Constructor'larda parametre olmaz.Scanner scan=new Scanner(System.in); içinde System.in şeklinde bir parametre olduğundan default Constructor değildir.Default Constructor Class oluşturulduğu anda Java tarafından oluşturulur. Biz kendimiz bir Constructor oluşturduğumuzda Java default Constructor'ı imha eder.

##### System.out.println(hondaAraba.price);

System.***out***.println(hondaAraba.year);

System.***out***.println(hondaAraba.make);

System.***out***.println(hondaAraba.type);

}

}

**package** day16overloading;

**public** **class** Constructor02 {

**public** **static** **void** main(String[] args) {

// Constructor02 Class'ı içinden Constructor01 Classından bir Constructor alıyoruz.

##### Her objeckt'e oluşturulduğu classtan isminin yanına koyulan bir nokta ile çağırıp içerdiği seçenekleri bulabiliriz, başka classta bunlara ulaşılmaz ama objekt ismi ile constructorı çağırabiliyoruz.

##### 

##### Başka bir Class'tan ürettiğimiz Objeckt'i object ürettiğimiz class'da kullanabiliriz.Objeckt(in özelliklerini değiştirebiliriz, fakat bu değişim sadece o object ile sınırlı kalır. yeni üretilecek olan objektler bu değişimden etkilenmez.Eğer üretilecek her Objekt'in değişime uğramış olmasını istiyorsanız class'daki özelliklerini değiştirmelisiniz.

Constructor01 hondaAraba01=**new** Constructor01();

System.***out***.println("Zamlı fiyat: "+ (hondaAraba01.price+1000)); // Arabanın fiyatına zam yaptık.

System.***out***.println("Year: "+ (hondaAraba01.year-1));

System.***out***.println(hondaAraba01.make.replace("Honda", "Toyota"));

System.***out***.println("Type: "+ hondaAraba01.type.replace("Civic", "Camry"));

// üstte yapılan değişikliler eski kalıbı bozmaz. bu değişiklikler sadece yukarıya münhasırdır.

Constructor01 hondaAraba02=**new** Constructor01();

System.***out***.println("Zamlı Fiyat: "+(hondaAraba02.price));

}

}

**package** day16overloading;

**public** **class** Constructor03 {

##### Variableları bütün Classta kullanacağımız için main Class dışına yazıyoruz(instance variable).this bu Class'ın içindekiler demektir.Constructor'ı kendimiz tanımlayınca default Constructor iptal olur artık bizim tanımlamalarımızı kullan demektir. Constructor oluşturma main dışında, objeckt oluşturma main içinde yapılır.

String isim="Ali Can";

**int** yas=33;

**int** kilo=85;

String meslek="Automation Tester";

**boolean** emekli=**false**;

Constructor03(){

}

##### Constructor03(String isim, int yas,boolean emekli){

##### 

##### this.isim=isim;

##### this.yas=yas;

##### this.emekli=emekli;

##### 

##### }

Constructor03(String isim){

**this**.isim=isim;

}

##### public static void main(String[] args) {

##### 

##### Constructor03 insan01=new Constructor03();

##### System.out.println(insan01.isim);

Constructor03 insan02=**new** Constructor03("Ayhan yıldız",56,**true**);

System.***out***.println(insan02.isim);

System.***out***.println(insan02.yas);

System.***out***.println(insan02.emekli);

System.***out***.println(insan02.meslek);

System.***out***.println(insan02.kilo);

Constructor03 insan03=**new** Constructor03("Ayse yıldız");

System.***out***.println(insan03.isim);

System.***out***.println(insan03.kilo);

System.***out***.println(insan03.emekli);

}

}

# OVERLOADİNG

**package** day16overloading;

**public** **class** Overloading01 { // Overloading aşırı yükleme demektir.

**public** **static** **void** main(String[] args) {

##### 1. Method ismini değiştirmeden parametreleri değiştirerek aynı isimli methodlar oluşturabilirsiniz.

##### 2. Main method içerisine çağrılan method'larda parametrelere bakar hangi parametre hangi methoda uyuyorsa onu kullanır.

##### 3. Parametre sayısı değiştirerek aynı isimli ayrı method oluşturulabilir.

##### 4. Parametre sayısını değiştirmeden herhangi bir parametrenin type'ini değiştirerek isim değiştirmeden ayrı bir method oluşturabiliriz.

##### 5. öncelik olarak data typlerden birebir olarak ölçüşene bakar onu çalıştırır, birebir örtüşen yoksa Auto wideninge göre kullanabileceği methoda bakar yoksa hata verir.

##### 6. farklı data type'indeki parametrelerin yerlerini değiştirerek te aynı isimli farklı methodlar oluşturabilirsiniz.

// iki sayının toplamını ekrana yazdıran bir method oluşturunuz

*toplama*(7.2,5.3); // Method 4

*toplama*(3,2); // Method 1

*toplama*(7); // Method 2

*toplama*("Ali"); // Method 3

}

##### // Method 1

##### 

##### public static void toplama (int num1,int num2) {

##### 

##### System.out.println(num1+num2);

##### 

##### }// Method 2

##### 

##### public static void toplama (int num1) {

##### 

##### System.out.println(num1+num1);

}// Method 3

**public** **static** **void** toplama (String str) {

System.***out***.println(str);

}// Method 4

**public** **static** **void** toplama (**double** num1,**double** num2) {

System.***out***.println(num1+num2);

}

}

**package** day16overloading;

**public** **class** Overlaoding02 {

**public** **static** **void** main(String[] args) {

##### Note 1: Return type'i değiştirmek overloading için yeterli değildir. Java methodların farklı olup olmadıklarını anlamak için iki şeye bakar.a) method ismine b) parametrelere bakar.

##### Method ismi ve parametreler Java için methodların imzası gibidir.Java da method imzası yerine " method signature" denir.

##### Note 2: Access Modifier'ları değiştirmek de methodları Java açısından farklı hale getirmez(public,privat vs. = access modifier)çünkü access modifier'lar method signature'a dahil değildir.

##### Note 3: Method'ların body'sini değiştirmek de methodları java açısında farklı hale getirmez çünkü body'lar method signature'a dahil değildir.

##### Note 4: Siz daha kodu yazarken yani kodu çalıştırmadan önce Java sizi yaptığınız hatalardan dolayı uyarırsa bu tip hatalara "Compile Time Error" denir. (altta kırmızı çizgi ile yazması)

##### Note 5: kodu yazarken herhangi bir hata yok. fakat kodu çalıştırdıktan sonra console'da kırmızı hata mesajlar alırsanız bu hatalara "Run Time Error" denir.

// public static void carpma(int num1, int num2) {

// System.out.println(num1\* num2);

// }

// public static int carpma(int num1, int num2) {

// return num1/num2;

// }

// private static int carpma(int num1, int num2) {

// return num1\* num2;

//

// }

}

}

# 17. DERS ( CONSTRUCTOR )

**package** day17constructors;

**public** **class** Constructors01 {

String isim="Ali Can";

**int** yas=33;

**public** **static** **void** main(String[] args) {

*buyume*(33);

##### method ile main method aynı Classta ise direk method ismi yazılarak kullanılabilir,

}

**public** **static** **void** buyume(**int** yas) {

yas++;

System.***out***.println(yas);

}

**public** **static** **void** isimDegistirme(String isim) {

System.***out***.println(isim);

}

}

**package** day17constructors;

**public** **class** Constructors02 {

**int** a = 5;

// buyume methodu bu class'da değil Constructors01 class'ında.

// O yüzden Java kullanmama müsade etmiyor.

// ama bazen bu methodu kullanmaya ihtiyacımız olabilir böyle bir durumda

// eğer siz başka bir class'daki methodu kullanmak istiyorsanız

// O class'tan bir tane objeckt oluşturup o objekt sayesinde istediğiniz methodu kullanabilirsiniz.

// static yazılmadığı zaman methodlar objelere bağlanıyor.

// static yazılırsa methodlar class"a ait olmuş oluyor.

// buyume(33); BU ŞEKİL YAZARSAM HATA VERİR.

**public** **static** **void** main(String[] args) {

Constructors01 obj1=**new** Constructors01();

obj1.*buyume*(33);

obj1.*isimDegistirme*("Kemal Can");

// büyüme methodu static olduğundan Java object kullanarak büyüme() methodunu

// çağırmamızı istemez ve kodun altını sarı renkle çizer.

// sarı renk beğenmedim ama çalıştırıyorum demektir. - mekruh-

System.***out***.println(obj1.isim);

System.***out***.println(obj1.yas);

##### Class ismini yazıp "." koyduğumuzda o classtaki Constructor variable'larına ulaşabiliriz. Eğer bir Variable veya method static olarak oluşturulmuşsa, o variable yada method'a objekt oluşturmadan sadece class ismini kullanarak ulaşabilirsiniz.

##### static methodları objekt üzerinden çağırmaya gerek yok class ismi üzerinden çağırmak daha pratiktir.

##### System.out.println(Constructors03.ad); // Ali Can

System.***out***.println(Constructors03.*kilo*); // 33

Constructors03.*artirma*(45); //

Constructors03.*degistirme*("Ayhan Işık"); // Ayhan Işık

}

}

**package** day17constructors;

**public** **class** Constructors03 {

**static** String *ad*="Ali Can";

**static** **int** *kilo*=33;

**public** **static** **void** main(String[] args) {

}

**public** **static** **void** artirma(**int** yas) {

yas++;

System.***out***.println(yas);

}

**public** **static** **void** degistirme(String isim) {

System.***out***.println(isim);

}

}

**package** day17constructors;

**public** **class** Constructors04 {

##### int num1=12;

##### static int num2=22;

**public** **static** **void** main(String[] args) {

##### Static methodların içinde sadece static elemanlar kullanılabilir. num1 static olmadığından main method'da static olduğundan main method içerisinde kullanılamaz.

##### num2 static olduğu için main metod içinde kullanmamız problem olmuyor.num1++; nın önündeki commenti kaldırırsanız Java num1++; nın altını kırmızı çizer.

// num1++;

*num2*++;

}

}

**package** day17constructors;

**public** **class** Constructors05 {

##### instance variable aynı classta object oluşturulmadan çağrılabilir.

##### Static olmayanlar static içinde kullanılamazlar. bunun için object oluşturmanız lazım.

##### Static bir variable'i object ile çağırırsak çalışır ama tavsiye edilmez,calışır ama compile error gösterir altını sarı olarak çizer.

##### Return type yoksa isim class ismi ile aynı ise bu Constructor dır.

##### metod static ise class ismi ile başka bir classtan bu metod çağırılabilir.

##### Static method içerisinde static ile tanımlama yapmıyoruz.

##### local variable'lara değer ataması yapılmak zorunda.

##### Constructor da access modifier olabilir ama return type olamaz(public Constructor05() {} gibi)

##### Static variable oluşturmanın diğer bir sebebi; ne kadar az variable kullanılırsa hafızada o kadar az yer kaplar static olunca daha az variable kullanılmış olur.

**public** **static** **void** main(String[] args) {

Constructors04 obj1=**new** Constructors04();

System.***out***.println(obj1.num1);

System.***out***.println(obj1.*num2*); // num2 static olduğundan object kullanarak

// Java istemiyor.

System.***out***.println(Constructors04.*num2*); // Class ismi ile çağırınca sarı çizgi kayboldu.

}

}

# 18. DERS ( THİS, STATİC KEYWORD )

# THİS

**package** day18statickeyword;

**public** **class** ThisKeyword {

**int** x =12;

**static** **int** *y* = 13;

ThisKeyword(**int** x){

##### this kullandığım zaman içinde bulunduğum class'taki instance veya static variable'lara ulaşmak istiyorum demektir.this constructorlar arasında geçiş yapmanın yoludur.

##### 

##### this(); // bu classın içinde parametresiz constructor'ı çağır demektir.this() kullanılacaksa mutlaka Constructor'ın ilk satıra yazılmalıdır aksi takdirde compile Time Error verir.

**this**.x=x;

System.***out***.println("Parametreli constructor");

}

ThisKeyword(){

System.***out***.println("Parametresiz constructor");

}

ThisKeyword(String str){

System.***out***.println("String Parametreli constructor");

}

**public** **static** **void** main(String[] args) {

ThisKeyword obj1=**new** ThisKeyword(15);

System.***out***.println(obj1.x); // 15

}

}

**package** day18statickeyword;

**public** **class** Test1 {

// for loop'un içerisinde Test1.i static i'yi kullanır.

// 2 şer olarak artırılan i ise local variable i’yi kullanır.

// static i 9 defa çalışıp 9 sayı yazdırır.

**static** **int** *i*=1;

**public** **static** **void** main(String[] args) {

**int** i=1;

**for** (Test1.*i*=1; Test1.*i*<10; Test1.*i*++) {

i=i+2;

System.***out***.print(i+" ");

}

}

}

**package** day18statickeyword;

**public** **class** MyClass01 {

**int** x = 3;

**int** y = 5;

MyClass01 (){

x += 1;

System.***out***.print("-x" + x);

}

MyClass01 (**int** i ){

**this**();

**this**.y = i;

x += y;

System.***out***.print("-x" + x);

}

MyClass01 (**int** i, **int** i2 ){

**this**(3);

**this**.x -= 4;

System.***out***.print("-x" + x);

}

**public** **static** **void** main(String[]args) {

MyClass01 mc1= **new** MyClass01(4,3);

System.***out***.println(mc1.x);

System.***out***.println(mc1.y);

}

}

**package** day18statickeyword;

**public** **class** Counter {

**int** count=0;

Counter(){

count++;

System.***out***.println(count);

}

**public** **static** **void** main(String[] args) {

Counter c1=**new** Counter();

Counter c2=**new** Counter();

Counter c3=**new** Counter();

System.***out***.println();

System.***out***.println(c1.count);

System.***out***.println(c2.count);

System.***out***.println(c3.count);

}

}

# STATIC BLOCK

**package** day18statickeyword;

**public** **class** Static01 {

**static** **int** *en*=12;

**static** **int** *boy*=10;

**static** **int** *yaricap*=10;

**static** **double** *pi*;

**static** **int** *alan*;

// int alan=en\*boy;

##### Eğer bir Variable oluştururken hesap yapmamız gerekirse "static block" oluşturmak iyidir. Yani; int alan=en\*boy demek yerine "static block" oluşturup bu işlemleri static block'in içine yazmak daha iyidir.

##### "static block" variable'lara değer atamak içinde kullanılabilir. "static block" Class'ın içinde bütün methodların(main dahil) ve Constructorların dışında oluşturulmalıdır."static block" Class oluşturulduğu zaman oluşturulur."static block" tüm methodlardan önce çalıştırılır.Birden fazla static block varsa üstteki önce çalışır.Static class oluşturulabilir ama alt classlar(inner class) static Class olabilir, üst class'lar static olamazlar. Static class main method dışında olması gerekir.

##### static {

##### pi=3.14;

##### }

##### 

##### static {

##### alan=en\*boy;

##### }

**public** **static** **void** main(String[] args) {

}

**static** **class** AltClass{ // alt Class

}

}

# 19.DERS (ARRAYS )

**package** day19arrays;

**public** **class** Arrays01 {

**public** **static** **void** main(String[] args) {

// Arrays: tepsi demektir.

##### Birden fazla varriable'i aynı anda kullanmak istediğimiz zaman arrays kullanıyoruz.

##### Array'ler object'tir bu yüzden Heap Memory'de depolanır.

##### Array'ler object'tir. bu yüzden valueların yanında methodlar da vardır.

##### Array'ler object'tir bu yüzden runtime'a oluşturulurlar.

##### Primitiveleri Java oluşturur. non primitive lardan ise istediğimiz kadar üretebiliriz.Bu non primitivelerden biriside arraysdir.Arraysler heapte depolanır.Heap büyük bir alan olduğu için bulunabilmesi için adreslenmesi gerekir.

##### Arrays içinde tüm variable'lar aynı data type'ınde olmalıdır.

##### Arrays sadece Primitive type (byte, short...8 tane) ve objeckt referans konulabilir.

##### Arrays'ın içinde Stringin adresi olur, kendisi olmaz.Array'in içine stackdekiler konulabilir.

##### Bir array oluşturmadan önce o Array'in içine kaç variable koyacağımıza karar vermeliyiz.Bir Array içine koyabileceğimiz variable sayısına o Array'in "length"i denir.

// bir Array'in length=2 gibi

##### O Array içine length'den fazla variable koyamayız.

##### int x; declare denir.

##### Array oluşturmak için iki yol vardır:

##### 1) int myArray[ ]; daha cok kullanılır

##### 2) int [ ]myArray; Dat Type isim [ ];

##### [ ] Array sembolü. Bu yazılınca Stack Memory'de bir adres oluşturulur.Eclipse de run butonuna basılınca Array oluşturulur.

## BİR ARRAY NASIL OLUŞTURULUR?

##### Array bir objeckt olduğu için new kullanılarak oluşturulur.

##### int myArray [ ] = new int[7];

##### 7 lenght uzunluğudur, değer yazılmazsa compile error verir.

##### int olarak öncesinde bir değer atanmamışsa default değer olarak 7 tane 0 atar.7 değer atanacak siz 3 değer atarsanız gerisini yazmazsanız 2 3 4 0 0 0 0 yapar.

##### boolean b[]=new boolean[3]; false false false gibi çünkü boolean default değeri falsedir.

##### array'ler index kullanır. bu nedenle array içindeki elamanlardan birini görmek istiyorsak index ini yazarak print yapabiliriz, index olmadan sadece array ismini yazarsak reference'ını (adresini) görürürüz.

##### 

##### Integer Array oluşturalım

##### 

##### int arr1[]=new int[5];

##### System.out.println(arr1); // [I@1540e19d yazar bu adrestir.

##### // array in ismini yazarak yazdırmaya çalışırsak adresi alırız.

System.***out***.println(arr1[0]); // ekranda 0 görürüz.

arr1[0]=12; // indexi 0 olana 12 ata demektir.

System.***out***.println(arr1[0]); // 12 görürüz

arr1[1]=13;

System.***out***.println(arr1[1]);

arr1[2]=14;

System.***out***.println(arr1[2]);

arr1[3]=15;

System.***out***.println(arr1[3]);

arr1[4]=22;

System.***out***.println(arr1[4]); // 0

// 12

// 13

// 14

// 15

// 22

for loop kullanarak array'deki elemanları ekrana yazdırın

**for** (**int** i=0; i<5;i++) {

System.***out***.print(arr1[i]+" ");

}

##### Array'de olmayan indexe değer ataması yapılırsa Run Time Error alınır.

##### Array'de olmayan indexi kullanmaya çalışırsak ArrayIndexOutOfBoundsException hatası alırız.

arr1[5]=23;

System.***out***.println(arr1[5]);

}

}

**package** day19arrays;

**public** **class** Arrays02 {

**public** **static** **void** main(String[] args) {

// 4 elemanlı char bir array oluşturalım.

// Tüm elemanları for döngüsü kullanarak ekrana yazdırın

**char** arr3 []=**new** **char**[4];

arr3[0]='A';

arr3[1]='D';

arr3[2]='E';

arr3[3]='M';

**for**(**int** i=0;i<4;i++) {

System.***out***.print(arr3[i]);

System.***out***.println();

System.***out***.println(arr3[3]);

## Array'in length'ini bulmak için "ArrayIsmi.length" yazmak yeterlidir.

Stringlerde de length methodu kullanılır ama Stringlerde kullanılan length methodu parantezlidir.

##### Stringlerde a.length() arraylerde ise sadece a.length şeklinde yazılıyor.

System.***out***.println(arr3.length);

##### Arraydeki son elemanı yazdırmak için length kullanınız.

##### System.out.println(arr3[arr3.length-1]);

// M görmemiz lazım.

}

}

}

**package** day19arrays;

**public** **class** Arrays03 {

**public** **static** **void** main(String[] args) {

## FLAG / COUNT

##### elemanları 1,2,3,4,5 bir int array oluşturunuz.Bu arrayde 3 elemanının olup olmadığını kontrol ediniz.3 varsa ekrana "3 var" yazdırın.3 yoksa "3 yok " yazdırın.

// Array oluşturup eleman eklemek için 1.YOL

// int arr[]=new int[5];

// arr[0]=1;

// arr[1]=2;

// arr[2]=3;

// arr[3]=4;

// arr[4]=5;

## Array oluşturup eleman eklemek

2. YOL

##### int arr[]= {1,2,3,4,5};

##### 

##### int count=0; // flag(bayrak) denir.

##### 

##### for (int i=0; i<5;i++) {

##### if(arr[i]==3) {

##### count++;

##### }

##### }

##### 

##### if(count>0) {

##### System.out.println("3 var");

##### }else {

##### System.out.println("3 yok");

##### }

// int arr1[]= {1,2,3};

// int arr2[]= {1,2,3};

//

// if(arr1==arr2) {

// System.out.println("same");

// }else {

// System.out.println("Not same"); // not same yazacaktır çünkü index karşılaştırması olmazsa

// // adres karşılaştırması yapar buda aynı sonucu vermez

}

}

**package** day19arrays;

**public** **class** Odev1901 {

**public** **static** **void** main(String[] args) {

// Bir array’de herhangi iki elemanın aynı olup olmadığını kontroleden bir program yazınız.

// Aynı eleman varsa “Aynı eleman var”

// yoksa “Aynı eleman yok” yazdırınız.

**int** arr[]= {11,12,13,14,15,11,18};

**int** count=0;

**for**(**int** i=0;i<arr.length;i++) {

**for** (**int** a=0; a<arr.length-1; a++){

**if**(i<a || i>a ? arr[i]==arr[a]:**false**) {

count++;

}

}

}**if**(count>0) {

System.***out***.println("Aynı eleman var");

}**else** {

System.***out***.println("Aynı eleman yok");

}

}

}

package day20arrays;

import java.util.Arrays;

import java.util.Scanner;

public class Arrays01 {

public static void main(String[] args) {

// Kullanıcıdan kaç elemanlı bir array girmesini isteyin.

// Bu array’in tum elemanlarını ekrana yazdırın.

// Bu arayın ilk elemanını son eleman yapın ve tum elemanlarını ekrana yazdırın.

// Mesela; array {1, 2, 3} ise ekrana {2, 3, 1} seklinde yazdırın.

Scanner scan=new Scanner(System.in);

System.out.println("Kaç elemanlı bir integer array oluşturmak istersiniz");

int length=scan.nextInt();

int arr[]=new int[length];

System.out.println("Array elemanlarını giriniz");

for(int i=0; i<length;i++) {

arr[i]=scan.nextInt();

}

System.out.println(Arrays.toString(arr));

## Arrays.toString() methoduna parametre olarak array'in ismini yazarsanız o array'in tüm elemanlarını ekranda görürsünüz.

##### System.out.println(Arrays.toString(arr));

int arrSon[]=new int[length];

for(int i=1; i<length;i++) {

arrSon[i-1]=arr[i];

}

arrSon[length-1]=arr[0];

System.out.println(Arrays.toString(arrSon));

scan.close();

}

}

**package** day20arrays;

**import** java.util.Arrays;

**public** **class** Arrays02 {

**public** **static** **void** main(String[] args) {

// Bir integer Array'in elemanlarını küçükten büyüğe diziniz

// ve ekrana iki durumuda yazdırınız

// String arr[]= {"Ali","Veli","Deniz"};

**int** arr[]= {2,11,0,23,7};

System.***out***.println("Sıralamadan önce: "+ Arrays.*toString*(arr));

##### [ ] işareti olmadan elemanları yazdırmak istiyorsak for loop kullanıyoruz.

##### for(int i=0; i<arr.length; i++) {

##### System.out.print(arr[i]+ ",");

##### }

System.***out***.println();

## Arrays.sort() methoduna parametre olarak array'in ismi yazılırsa o Array'in elemanlarını küçükten büyüğe sıralanmış olur.

##### 

##### Arrays.sort(arr);

System.***out***.println("Sıralamadan sonra: "+ Arrays.*toString*(arr));

// char'lardan oluşan bir array oluşturunuz ve "ascending order" yapınız.

**char** arrCh[]= {'a','c','x','j'};

System.***out***.println("Sıralamadan önce: "+ Arrays.*toString*(arrCh));

Arrays.*sort*(arrCh);

System.***out***.println("Sıralamadan sonra: "+ Arrays.*toString*(arrCh));

##### charlar sıralanırken java ASCII kodlarını kullanır.Büyük harflerin ASCII kodları küçük harflerden küçüktür,bu nedenle büyük harfler öne geçer.

// java rakamların ASCII kodunu o rakamın değeri gibi düşünür ve bu harflerden küçüktür,

// Soru: String'lerden oluşan bir array oluşturun ve ascending order yapın

String arrSt[]= {"can","Ali","Veli","Deniz","ahmet","9"};

System.***out***.println("Sıralamadan önce: "+ Arrays.*toString*(arrSt));

Arrays.*sort*(arrSt);

System.***out***.println("Sıralamadan önce: "+ Arrays.*toString*(arrSt));

// Soru: boolean bir array oluşturun. ascending order(küçükten büyüğe sıralayın)oluşturun.

##### booleanlar için sort methodu kullanılamaz.

**boolean** arrBl[]= {**true**,**false**,**false**};

System.***out***.println("Sıralamadan önce"+ Arrays.*toString*(arrBl));

}

}

**package** day20arrays;

**import** java.util.Arrays;

**public** **class** Arrays03 {

**public** **static** **void** main(String[] args) {

// iki array oluşturup bu array'lerin birbirine eşit

// olup olmadığını kontrol ediniz.

// Not: ik arrayin eşit olabilmesi için

// 1) Elemanları aynı olmalı

// 2) eşit elemanların index'leri de eşit olmalı

// {1,2,3} eşittir. ama {1,2,3} ile {1,3,2} eşit değildir.

**int** arr1[]= {1,2,3};

**int** arr2[]= {1,2,3};

##### System.out.println(arr1==arr2);

##### // false verir çünkü referenc'ları karşılaştırır.

// "==" adresleride kontrol eder.

// iki array farklı iki object olduğundan adresleride farklıdır.

##### System.out.println(Arrays.equals(arr1, arr2));

##### // true verir çünkü Arrays.equal() methodu sadece değerleri ve indexleri kontrol eder adreslere bakmaz.

}

}

**package** day20arrays;

**import** java.util.Arrays;

**public** **class** Arrays04 {

**public** **static** **void** main(String[] args) {

Bir array'in içinde belli bir elemanın var olup olmadığını kontrol ediniz.

// binarySearch (ikili araştırma demek) bu iş için kullanılır.

## binarySearch() methodu aranan eleman var olduğunda o elemanın indexini return eder.

##### Önemi Not: binarySearch methodunu kullanmadan önce sort() methodunu kullanmak zorundasınız aksi takdirde binarySearch methodu manalı bir sonuç vermez,verdiği sonuc güvenilir olmaz.

##### int arr[]= {1,12,12,8,2,4,3};

##### 

##### // yukarıda verilen array'de 3 sayısı eleman olarak var mı?

##### 

##### Arrays.sort(arr);

##### Arrays.binarySearch(arr, 3);

##### 

##### System.out.println(Arrays.binarySearch(arr, 3));

// 2

##### sonuc 2 yazar. önce sıralama yapar sonra bize indexi döndürür çünkü binary yapıyor bize sayılar döndürecek.

System.***out***.println(Arrays.*binarySearch*(arr, 12)); // 5

##### aynı elemandan birden fazla olduğunda ilkinin indexini verir. elemanı bulduktan sonra break yapar.

##### System.out.println(Arrays.binarySearch(arr, 5)); //-5

##### binarySearch() methodu olmayan elemanlar için negatif sayılar return eder.

##### negatif olması o elemanın arrayde olmadığı anlamına gelir

##### -5 o eleman olsaydı kaçıncı eleman olduğu anlamındadir.

##### {1,2,3,4,18,12,12} var olsaydı 4 ten hemen sonra olurdu buda 5.eleman demektir.

}

}

**package** day20arrays;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** Odev2002 {

**public** **static** **void** main(String[] args) {

// Soru:

// Kullanıcıya kaç elemanlı bir array gireceğini sorun.

// Kullanıcıdan array’in elemanlarını girmesini isteyin.

// a) Bu array’in icinde herhangi bir elemanın var olup olmadigini kontrol edin.

// b) Bu arayın tum elemanlarını tersten yazdırın.

// Mesela; array {1, 2, 3, 4} ise ekrana {4, 3, 2, 1} seklinde yazdırın.

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Kaç elemanlı bir Array oluşturmak istiyorsunuz");

**int** length=scan.nextInt();

**int** arr[]=**new** **int**[length];

System.***out***.println("Lütfen array elemanlarını yazınız");

**for** (**int** i = 0; i < length; i++) {

arr[i]=scan.nextInt();

}

Arrays.*sort*(arr);

System.***out***.println("Array içinde aramak istediğiniz karakteri giriniz");

**int** j=scan.nextInt();

**if**(Arrays.*binarySearch*(arr, j)>=0) {

System.***out***.println("istediğiniz karakter Array içinde mevcut");

}**else** {

System.***out***.println("istediğiniz karakter Array içinde mevcut değil");

}

**int** arr2[]=**new** **int**[length];

**for** (**int** i = length-1; i >= 0; i--) {

arr2[(length-i)-1]=arr[i];

}

System.***out***.println("ilk girdiğiniz Array elemanları :"+ Arrays.*toString*(arr));

System.***out***.println("Array elemanlarının tersten yazılımı: "+ Arrays.*toString*(arr2));

scan.close();

}

}

# 21. DERS ( MULTİDİMENSİONALARRAY )

**package** day21multidimensionalarray;

**import** java.util.Arrays;

**public** **class** SplitMethod {

**public** **static** **void** main(String[] args) {

## split bir string methodudur. str.split( ); methodunu seçip parametre içine hangi karakteri yazarsak boşluklara göre Stringi parçalayıp kaç parça olduğunu bize yazar.

##### Girilen karakteri çıkarır aradaki kelimeleri sayar. " " koyduğumuzda kelimeler ortaya çıkar. aşağıdak bu şekilde 4 tane String oluşmuş olur. Split methodu kullandığınızda mutlaka bir Array oluşturmanız gereklidir.

// Kelimeleri sayma

String str="Ali Can okula basladi";

String kelime[]=str.split(" ");

System.***out***.println("Cümlemiz: "+str);

System.***out***.println("Kelimelerim: "+ Arrays.*toString*(kelime));

System.***out***.println("kelime sayıs: "+ kelime.length);

// harfleri sayma

String str2="Kahramanmaraş";

String harf[]=str2.split("");

System.***out***.println();

System.***out***.println("Kelimem: "+str2);

System.***out***.println(Arrays.*deepToString*(harf));

System.***out***.println("Harf sayısı: "+harf.length);

}

}

**package** day21multidimensionalarray;

**public** **class** SplitMethodSoru {

**public** **static** **void** main(String[] args) {

// Verilen bir cümledeki boşluklar hariç karakter sayısını bulunuz

// her zaman boşluk sayısı kelime sayısından bir eksiktir.

// 1.YOL

String str1="karakter sayısını bulma";

str1=str1.replace(" ", "");

String harf[]=str1.split("");

System.***out***.println(harf.length);

// 2.YOL

//Bosluk sayisi kelime sayisindan bir azdir

//Bosluk sayisi = Kelime sayisi - 1

String str2 = "Verilen bir cumledeki bosluklar haric character sayisini bulunuz.";

String kelime[] = str2.split(" ");

**int** boslukSayisi = kelime.length - 1;

System.***out***.println();

System.***out***.println(boslukSayisi);

String character[] = str2.split("");

System.***out***.println(character.length - boslukSayisi);

}

}

# MULTİDİMENSİONAL ARRAY

**package** day21multidimensionalarray;

**import** java.util.Arrays;

**public** **class** MultiDimensionalArray01 {

**public** **static** **void** main(String[] args) {

// Multi dimensional Array oluşturma (2 boyutlu oluşturma)

// asıl arrayin içinde 3 eleman onun içindede 4 eleman var

##### int arr[][]=new int[3][4];

##### System.out.println(Arrays.toString(arr)); // [[I@1540e19d, [I@677327b6, [I@14ae5a5]

##### 

##### System.out.println(Arrays.deepToString(arr)); // .deepToString sonuna kadar git tüm elemanları yazdır demektir.

##### // [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]] yazdırır.

// DEĞER ATAMA 1.YOL

arr[0][0]=1;

arr[0][1]=2;

arr[0][2]=3;

arr[0][3]=4;

arr[1][0]=5;

arr[1][1]=6;

arr[1][2]=7;

arr[1][3]=8;

arr[2][0]=9;

arr[2][1]=10;

arr[2][2]=11;

arr[2][3]=12;

System.***out***.println(Arrays.*deepToString*(arr));

System.***out***.println(arr[1][2]);

**int** top=arr[0][3]+arr[2][1];

System.***out***.println(top);

}

}

**package** day21multidimensionalarray;

**import** java.util.Arrays;

**public** **class** MultiDimensionalArray02 {

**public** **static** **void** main(String[] args) {

// Multi Dimensional Array Oluşturup Değer Atama 2.Yöntemi

**int** arr[][]= {{1,2},{3}, {4,5,6} } ;

System.***out***.println(Arrays.*deepToString*(arr));

System.***out***.println(arr[0][1]+arr[1][0]+arr[2][2]); // 2+3+6=11

// arr arrayindeki tüm elemanların toplamını veren programı yazınız.

##### arr.length bize iç içe kaç tane array olduğunun sayısını veriyor.(ilk index parantez )

##### System.out.println(arr.length); // kaç ayrı array grubu varsa onun sayısını verir(ilk index)

##### // arr[i].length ikinci index parantezinin uzunluğu için

##### System.out.println(arr[1].length);// ikinci array grubu içerisindeki eleman sayısını verir.

**int** sum=0;

**for**(**int** i=0; i<arr.length;i++) {

**for** (**int** j = 0; j < arr[i].length; j++) {

sum=sum + arr[i][j];

}

}

System.***out***.println("Tüm elemanların toplamı: "+sum);

}

}

**package** day21multidimensionalarray;

**public** **class** MultiDimensionalArray03 {

**public** **static** **void** main(String[] args) {

// arr1 = { {1,2}, {3,4,5}, {6} } ve

// arr2 = { {7,8,9}, {10,11}, {12} } veriliyor.

// Bu iki array’in tum elemanlarinin toplamini bulan programi yaziniz.

**int** arr1[][] = { {1,2}, {3,4,5}, {6} };

**int** arr2 [][] = { {7,8,9}, {10,11}, {12} };

**int** sum=0;

**for** (**int** i = 0; i < arr1.length; i++) {

**for** (**int** j = 0; j < arr1[i].length; j++) {

sum=sum + arr1[i][j];

}

}

**int** sum2=0;

**for** (**int** i = 0; i < arr2.length; i++) {

**for** (**int** j = 0; j < arr2[i].length; j++) {

sum2=arr2[i][j]+ sum2;

}

}

System.***out***.println(sum+sum2);

}

}

**package** day21multidimensionalarray;

**import** java.util.Arrays;

**public** **class** Odev2101 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki multi dimensional array’in tum elemanlarının

// çarpımını ekrana yazdıran bir program yazınız.

// { {1,2,3}, {4,5,6} }

**int** arr[][]={ {1,2,3}, {4,5,6} };

**int** mult=1;

**for** (**int** i = 0; i < arr.length; i++) {

**for** (**int** j = 0; j < arr[i].length; j++) {

mult=mult\*arr[i][j];

}

}

System.***out***.println(Arrays.*deepToString*(arr)+" Sayılarının çarpımı= "+mult);

}

}

**package** day21multidimensionalarray;

**import** java.util.Arrays;

**public** **class** Odev2102 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki multi dimensional array’in

// iç array’lerindeki son elemanların çarpımını ekrana yazdıran

// bir program yazınız { {1,2,3}, {4,5}, {6} }

**int** arr[][]= { {1,2,3}, {4,5}, {6} };

**int** mult=1;

**for** (**int** i = 0; i < arr.length; i++) {

mult=mult\*arr[i][(arr[i].length)-1];

}

System.***out***.println(Arrays.*deepToString*(arr)+" Array'inin son elemanları çarpımı= "+mult);

}

}

**package** day21multidimensionalarray;

**public** **class** Odev2103 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki multi dimensional array’lerin iç array’lerinde

// aynı index’e sahip elemanların toplamını ekrana yazdıran bir program yazınız

// arr1 = { {1,2}, {3,4,5}, {6} } ve arr2 = { {7,8,9}, {10,11}, {12} }

**int** arr1[][]= { {1,2}, {3,4,5}, {6} };

**int** arr2 [][]= { {7,8,9}, {10,11}, {12} };

**int** leng1=arr1.length;

**int** leng2=arr2.length;

**int** lengi;

**if** (leng1<=leng2) {

lengi=leng1;

}**else** {

lengi=leng2;

}

**for** (**int** i = 0; i < lengi; i++) {

**int** lengj;

**if** (arr1[i].length<=arr2[i].length) {

lengj=arr1[i].length;

}**else** {

lengj=arr2[i].length;

}

**for** (**int** j = 0; j < lengj; j++) {

System.***out***.print("arr1"+"["+i+"]"+"["+j+"]"+"+"+"arr2"+"["+i+"]"+"["+j+"]"+ "="+" ");

System.***out***.println(arr1[i][j]+arr2[i][j]);

}

}

}

}

**package** day21multidimensionalarray;

**import** java.util.Arrays;

**public** **class** Odev2104 {

**public** **static** **void** main(String[] args) {

// Aşağıdaki multi dimensional array’in iç array’lerindeki

// tum elemanların toplamını birer birer bulan

// ve herbir sonucu yeni bir array’in elemanı yapan

// ve yeni array’i ekrana yazdıran bir program yazınız

// { {1,2,3}, {4,5}, {6,7} }

// Ornegin; { {1,2,3}, {4,5}, {6,7} } ==> 1+2+3=6 4+5=9 6+7=13 ==> output: {6, 9, 13}

**int** arr[][] = { { 1, 2, 3 }, { 4, 5 }, { 6, 7 } };

**int** arr1[] = **new** **int**[arr.length];

**int** sum=0;

**for** (**int** i = 0; i < arr.length; i++) {

**for** (**int** j = 0; j < arr[i].length; j++) {

sum = sum + arr[i][j];

}

arr1[i] = sum;

sum=0;

}

System.***out***.println(Arrays.*toString*(arr1));

}

}

# ARRAY LİST

**package** day21multidimensionalarray;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ArrayList01 {

**public** **static** **void** main(String[] args) {

##### Array'lerde lenght sabittir, değiştirilemez. ArrayListler'de (List) length esnektir. Siz eleman ekledikçe List Length'ini artırır. Siz eleman sildikçe List length'ini azaltır. Hem List hem de ArrayList üzerine gelip java.util olarak import yapıyoruz.

// List<String> list = new List<>(); OLMAZ

##### ArrayList<String> list = new List<>(); OLMAZ

##### ArrayList<String> list = new ArrayList<>(); OLABİLİR

//List<String> list = new ArrayList<String>(); OLABİLİR

##### List<String> list = new ArrayList<>(); GENEL KULLANIM ŞEKLİ

##### List'e eleman eklemek için add() methodu kullanılır.

List<String> list01 = **new** ArrayList<>();

System.***out***.println(list01); // []

##### list01.add("Ali");

System.***out***.println(list01); // []

//[Ali]

// Veliyi araya ekleme

list01.add("Can");

list01.add(1, "Veli");

System.***out***.println(list01); // [Ali, Veli, Can]

// list01'i [Han, Ali, Kemal, Veli, Can, Aysel] şekline getirin.

list01.add(0, "Han");

list01.add(2, "Kemal");

list01.add(5, "Aysel");

System.***out***.println(list01);

System.***out***.println("Eleman sayısı: "+ list01.size()); // 6

}

}

**package** day21multidimensionalarray;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ArrayList02 {

**public** **static** **void** main(String[] args) {

// List oluşturun isim list01 olsun elemanlar integer olsun

##### List'ler data type olarak primitive'leri kabul etmez. Primitive'lerde non-Primitive yapmak için Wrapper class'ları kullanırız.

##### 

##### List<Integer> list01=new ArrayList<>();

##### Bir list'in içinde elaman olup olmadığını kontrol etmek için isEmpty methodu kullanılır. Boş ise true , dolu ise false return eder.

##### System.out.println(list01.isEmpty());

##### Bu liste eleman ekleyin. list01.add(123);

System.***out***.println(list01.isEmpty());

##### List'den eleman silmek için remove() methodu kullanılır. iki remove methodu var biri indexle çalışıyor

list01.add(124);

list01.add(125);

list01.add(124);

System.***out***.println(list01);

// list01.remove(index);

list01.remove(1); // indexi 1 olan elemanı siler

// son elemanı siliniz.

System.***out***.println();

System.***out***.println(list01); // [123, 124, 125, 124]

//123 ü silin

list01.remove(0);

System.***out***.println(list01); // [125, 124]

// listteki son elemanı silin

list01.remove(list01.size()-1);

System.***out***.println(list01); // 125

}

}

**package** day21multidimensionalarray;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ArrayList03 {

**public** **static** **void** main(String[] args) {

// ["Ali", "Can", "Ayse", "Ali"] Stringleri ile bir List olusturun.

List<String> list = **new** ArrayList<>();

list.add("Ali");

list.add("Can");

list.add("Ayse");

list.add("Ali");

System.***out***.println(list);

##### list.remove("Ali"); //Birden fazla ayni eleman varsa ilkini siler

System.***out***.println(list);//[Can, Ayse, Ali]

##### list.remove("Kemal");//Listede olmayan elemanlar icin error vermez

System.***out***.println(list);//[Can, Ayse, Ali]

##### System.out.println(list.remove("Ali"));//true

##### System.out.println(list.remove("Kemal"));//false

System.***out***.println(list.remove(0));//Bu, Can elemanini siler ve ekrana silinen elemani yazar

System.***out***.println(list);

##### "Ayse" elemanini "Aysegul" yapin. set() methodu kullanilir.

##### System.out.println(list.set(0, "Aysegul"));//Ayse yani degisimden onceki hali ekrana yazdirir.

##### System.out.println(list);//[Aysegul]

//List'e "Kenan" ve "Zeynep" ekleyin

list.add("Kenan");

list.add("Zeynep");

System.***out***.println(list);//[Aysegul, Kenan, Zeynep]

list.clear();

System.***out***.println(list.isEmpty());//true

System.***out***.println(list);//[]

}

}

**package** day21multidimensionalarray;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** ArrayList04 {

**public** **static** **void** main(String[] args) {

// ["A", "C", "B", "C"] listini oluşturun

List<String> list=**new** ArrayList<>();

list.add("A");

list.add("C");

list.add("B");

list.add("C");

System.***out***.println(list);

// list de eleman olarak B var mı?

// contains methodu kullanılır.

## list.contains("B"); list B elemanını içeriyor mu demektir. varsa true yoksa false return eder.

##### 

##### list.contains("B");

##### System.out.println(list.contains("B")); // true verir.

Bir listteki elemanları alfabetik sıraya(Natural Order) dizmek için Collections.sort(list); methodu kullanılır. Parametre olarak listin adı girilir.

// list deki elemanları alfabetik sıraya koyunuz.

##### Collection ==> bir araya getirilmiş parçalar. ayakkabılık, kalemlik, koleksiyon gibi. Collection bir class, içerisinde methodlar var.

##### 

##### Collections.sort(list);

System.***out***.println(list);

}

}

**package** day21multidimensionalarray;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** Odev2105 {

**public** **static** **void** main(String[] args) {

// Elemanları A, C, E, ve F olan bir String ArrayList oluşturup ekrana yazdırınız.

List<String> list=**new** ArrayList<>();

list.add("A");

list.add("C");

list.add("E");

list.add("F");

System.***out***.println(list);

// İndexsiz add() methodunu kullanarak, B’yi ekleyiniz.

// İndex’li add() methodunu kullanarak, L’yi 1 numaralı index’e ekleyiniz.

// ArrayList’i ekrana yazdırınız, list şöyle olmalı; A, L, C, E, F, B.

list.add("B");

list.add(1, "L");

System.***out***.println(list);

// set() methodu kullanarak, E’yi D yapınız.

// ArrayList’i ekrana yazdırınız, list şöyle olmalı; A, L, C, D, F, B.

list.set(3, "D");

System.***out***.println(list);

// remove() methodu kullanarak, F’yi siliniz.

// ArrayList’i ekrana yazdırınız, list şöyle olmalı; A, L, C, D, B.

list.remove(4);

System.***out***.println(list);

// sort() methodu kullanarak, elemanları alfabetik sıraya diziniz.

// ArrayList’i ekrana yazdırınız, list şöyle olmalı; A, B, C, D, L.

Collections.*sort*(list);

System.***out***.println(list);

// contains() methodu kullanarak, L’nin list’de var oldugunu ve M’nin list’de

// var olmadığını doğrulayınız.

System.***out***.println(list.contains("L"));

System.***out***.println(list.contains("M"));

// size() methodu kullanarak, list’in kaç eleman olduğunu ekrana yazdırınız.

System.***out***.println(list.size());

## clear() methodu kullanarak, list’deki tum elemanları siliniz.

##### 

##### list.clear();

System.***out***.println(list);

// isEmpty() methodu kullanarak, list’deki tum elemanların silindiğini doğrulayınız.

System.***out***.println(list.isEmpty());

}

}

# 22. DERS ( ARRAY LIST)

**package** day22arrayList;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ArrayList01 {

**public** **static** **void** main(String[] args) {

## equals() methodu iki listin birbirnie eşit olup olmadığını kontrol eder, List'ler eşit ise "true", değilse "false" return eder.

##### equals() methodu aynı index'de aynı eleman olup olmadığını kontrol eder, equals() methodu objelerin reference (adreslerine) larına bakmaz.

List<String> list1 = **new** ArrayList<>();

List<String> list2 = **new** ArrayList<>();

##### System.out.println(list1.equals(list2)); // true

list1.add("A");

System.***out***.println(list1.equals(list2)); // false

list2.add("A");

System.***out***.println(list1.equals(list2)); // true

list1.add("A");

list1.add("B");

list2.add("B");

list2.add("A");

System.***out***.println(list1.equals(list2)); // false

}

}

**package** day22arrayList;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** ArrayList02 {

**public** **static** **void** main(String[] args) {

## toArray() methodu listi Array'e çevirmek için kullanılır.

List<String> list = **new** ArrayList<>();

list.add("Ali");

list.add("Can");

System.***out***.println(list); // [Ali, Can]

##### I. YOL: to Array() method'unun içinde parametre olarak new String[0] kullanınız.

##### 

##### String arr[]=list.toArray(new String[0]);

System.***out***.println(Arrays.*toString*(arr));

##### II. YOL: Oluşturduğunuz array'in data type'ini Object olarak seçin.

##### Object class, Java'da bütün class'ların parent'idir. Yani Object class orta atadır.Object class, Java'da parenti olmayan tek class'dır.String, Object class'ın child'i olduğundan data type olarak bazı durumlarda String yerine Object kullanırız.

##### 

##### Object[] arr1=list.toArray();

System.***out***.println(Arrays.*toString*(arr1));

## asList( ) methodu array'leri liste çevirmek için kullanılır.

##### asList( ) methodu parametre olarak array'in ismini kabul eder.

String arr2[]= {"Aliye","Can"};

##### List<String> list1 = Arrays.asList(arr2);

System.***out***.println(list1);

// list1.add("Emine");

// System.out.println(list1); // Run Time Error hata verir.boyut değişemiyor.

##### Array'den liste çevirme yaptığınızda, elde ettiğiniz list uzunluk olarak esnek değildir. Yani array'den oluşturduğunuz liste ekleme ve çıkarma yapamazsınız.

// list1.remove("Aliye"); // boyut değişeceği için remove() yapamazsınız.

list1.set(0, "Kemal");

System.***out***.println(list1); // [Kemal, Can]

// List<String> list=new ArrayList<>();

// list.add("one");

// list.add("two");

// list.add(7); // list lere sadece tek tip data eleman eklenebiliyor hata verir.

}

}

# FOR EACH LOOP

**package** day22arrayList;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ForEachLoop01 {

**public** **static** **void** main(String[] args) {

For Each Loop, for loop'un üst versiyonudur.Diğer adı "Enhanced(Zenginleştirilmiş) For Loop"

//mümkün olduğunca For Loop yerine bundan sonra bunu kullanacağız.

##### int arr[]= {12, 21, 13, 43};

##### for loop kullanarak elemanları ekrana yazdır.

##### 

##### for (int i = 0; i < arr.length; i++) {

##### System.out.print(arr[i]);

##### }

System.***out***.println();

System.***out***.println("=======");

##### for each loop kullanarak elemanları ekrana yazdır.

##### 

##### for(int w:arr) {

##### // w 0.indexten başlayıp son indexe kadar uygular.

##### System.out.print(w);

##### }

System.***out***.println();

System.***out***.println("=======");

// String içeren bir array oluşturun. Elemanlarını yanyana aralarına boşluk koyarak

// For Each Loop ile ekrana yazdırınız.

String arr1[]= {"for","each","kullan"};

**for**(String w:arr1) {

System.***out***.print(w+" ");

}

System.***out***.println();

System.***out***.println("=======");

// Integer elemanlar içeren bir list oluşturun

// for each loop kullanarak bu elemanların toplamını ekrana yazdırın.

List<Integer> list =**new** ArrayList<>();

list.add(11);

list.add(12);

list.add(13);

**int** sum=0;

**for**(**int** w:list) {

sum=sum+w;

}

System.***out***.println("elemanlar toplamı= "+sum);

System.***out***.println();

System.***out***.println("=======");

##### // {{1, 2}, {5}, {6, 7, 8}} arrayindeki tüm elemanların toplamını bulunuz.

##### 

##### int arr2[ ][ ]={{1, 2}, {5}, {6, 7, 8}};

##### int sum1=0;

##### 

##### for(int[ ] w:arr2) {

##### for(int z: w) {

##### sum1=sum1+z;

##### }

##### }

System.***out***.println(sum1);

}

}

**package** day22arrayList;

**public** **class** Odev2201 {

**public** **static** **void** main(String[] args) {

// For-each loop Soru 1:

// Bir integer array oluşturunuz

// ve bu array’deki tum sayıların çarpımını

// For-each loop kullanarak bulunuz. Sonucu ekrana yazdırınız.

**int** arr[]= {1,2,3,4,5};

**int** mult=1;

**for**(**int** w:arr) {

mult=mult\*w;

}

System.***out***.println("arr!= "+mult);

}

}

**package** day22arrayList;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Odev2202 {

**public** **static** **void** main(String[] args) {

// For-each loop Soru 2:

// Bir integer list oluşturunuz

// ve bu list’deki tum sayıların karesinin toplamını

// For-each loop kullanarak bulunuz. Sonucu ekrana yazdırınız

List<Integer> list=**new** ArrayList<>();

list.add(2);

list.add(4);

list.add(6);

**int** sum=0;

**int** mult=1;

**for**(**int** w: list) {

mult=w\*w;

sum=sum+mult;

}

System.***out***.println(sum);

}

}

**package** day22arrayList;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Odev2203 {

**public** **static** **void** main(String[] args) {

// For-each loop Soru 3:

// İki String array oluşturunuz

// ve bu array’lerdeki ortak elemanları For-each loop kullanarak bulunuz.

// Sonucu ekrana yazdırınız.

// Ortak eleman yoksa ekrana “Ortak eleman yok” yazdırınız

String arr1[]= {"Ali", "Veli","Can"};

String arr2[]= {"Deniz","Veli","Ahmet"};

List<String>sonuc=**new** ArrayList<>();

**for**(String w:arr1) {

**for**(String y:arr2) {

**if**(w.equals(y)) {

sonuc.add(w);

}

}

}**if**(sonuc.isEmpty()) {

System.***out***.println("Ortak eleman yok");

}**else** {

System.***out***.println(sonuc);

}

}

}

**package** day22arrayList;

**public** **class** Odev2204 {

**public** **static** **void** main(String[] args) {

// For-each loop Soru 4:

// Bir String oluşturunuz,

// bu String’deki character’leri yukarıdan aşağıya for-each loop kullanarak yazdırınız.

// İpucu: split()

String kelime= "Elveda";

String ch[]=kelime.split("");

**for**(String w:ch) {

System.***out***.println(w);

}

}

}

**package** day23date;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ListAllMethod {

**public** **static** **void** main(String[] args) {

## listaddAll() methodu iki listi birleştirmek için kullanılır.

##### index kullanmadan addAll()yazım sırasına göre bir ekleme yaparsınız.list'i istediğiniz index'e eklemek istiyorsanız addAll(index,element) seçmeniz lazım.

List<Integer>list1=**new** ArrayList<>();

list1.add(11);

list1.add(12);

System.***out***.println(list1); // [11, 12]

List<Integer>list2= **new** ArrayList<>();

list2.add(13);

list2.add(14);

list2.add(15);

System.***out***.println(list2); // [13, 14, 15]

##### list1.addAll(list2);

System.***out***.println(list1); // [11, 12, 13, 14, 15]

list2.addAll(list1);

System.***out***.println(list2); // [13, 14, 15, 11, 12, 13, 14, 15]

List<Integer> list3=**new** ArrayList<>();

list3.add(1);

System.***out***.println(list3); // [1]

List<Integer> list4=**new** ArrayList<>();

list4.add(2);

list4.add(3);

System.***out***.println(list4); // [2, 3]

##### list4.addAll(1, list3);; // [2, 1, 3]

System.***out***.println(list4);

}

}

# 23.DERS ( LocalDate )

**package** day23date;

**import** java.time.LocalDate;

**import** java.time.LocalDateTime;

**import** java.time.LocalTime;

**public** **class** Date01 {

**public** **static** **void** main(String[] args) {

## Bugünün tarihini ekrana yazdıralım.

##### System.out.println(LocalDate.now());

System.***out***.println(LocalTime.*now*());

##### System.out.println(LocalDateTime.now());

##### 2020-05-20T17:03:01.519 buradaki "T" tarih ile zamanı ayırmak için konuluyor.

}

}

**package** day23date;

**import** java.time.LocalDate;

**public** **class** Date02 {

**public** **static** **void** main(String[] args) {

// manipule değiştirmek demektir.

## Date Manipulation (Date üzerinde değişiklikler yapmak)

##### LocalDate date=LocalDate.now();

System.***out***.println(date);

##### System.out.println(date.plusDays(1));

System.***out***.println(date.plusDays(13));

System.***out***.println(date.plusDays(-1)); // çok kullanılmaz bunun için başka bir method var.

##### System.out.println(date.plusMonths(2));

##### System.out.println(date.plusYears(3));

##### System.out.println(date.plusWeeks(4));

##### System.out.println(date.minusDays(3));

System.***out***.println(date.minusMonths(2));

}

}

**package** day23date;

**import** java.time.LocalTime;

**public** **class** Date03 {

**public** **static** **void** main(String[] args) {

## Time Manipilation (Zamanı değiştirme)

##### LocalTime time = LocalTime.now();

##### System.out.println(time);

##### 

##### System.out.println(time.plusSeconds(45));

System.***out***.println(time.plusMinutes(20));

System.***out***.println(time.plusHours(10));

System.***out***.println(time.minusSeconds(33));

System.***out***.println(time.minusMinutes(12));

System.***out***.println(time.minusHours(12));

LocalTime time1 = LocalTime.*now*();

System.***out***.println(time1);

}

}

**package** day23date;

**import** java.time.LocalDate;

**import** java.time.LocalTime;

**import** java.time.ZoneId;

**import** java.time.format.DateTimeFormatter;

**public** **class** Date04 {

**public** **static** **void** main(String[] args) {

## Date formatını değiştirmek

##### Java minute için "m" kullanır, ay için "M" kullanır.

##### LocalDate date=LocalDate.now();

##### 

##### DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd.MMM.yyyy");

##### System.out.println(dtf.format(date)); // 08.Nis.2021

DateTimeFormatter dtf1 = DateTimeFormatter.*ofPattern*("dd.MM.yyyy");

System.***out***.println(dtf1.format(date.plusMonths(7))); // 08.11.2021

DateTimeFormatter dtf2 = DateTimeFormatter.*ofPattern*("yyyy/M/dd");

System.***out***.println(dtf2.format(date)); // 2021/4/08

## Time Formatını değiştirme

##### HH kullanırsanız 24 saatlik dilimi kullanır. hh olursa pm am kullanır.

##### LocalTime time=LocalTime.now();

##### 

##### DateTimeFormatter dtf3=DateTimeFormatter.ofPattern("hh:mm");

##### System.out.println(dtf3.format(time.plusHours(3))); // 12:39

## Başka ülkenin local zamanını almak

##### time.now(zone); bir ülkenin zone değerini GMT-3

##### System.out.println(LocalTime.now(ZoneId.of("America/Sao\_Paulo"))); // 04:42:18.784

}

}

# 24.DERS ( VARIABLE ARGUMENTS )

**package** day24varargsaccessmodifiers;

**public** **class** Varargs01 {

**public** **static** **void** main(String[] args) {

## Varargs: Variable Arguments

##### Bir method oluşturduğumuzda argüment olarak istediğimiz kadar argüment girebilmemizi sağlar. int den sonra ... koyunca java bunun Varargs olduğunu anlar.sonu açık olan bir Array gibi düşünebiliriz.

*countNum*(5); // length uzunluğu 1

##### countNum(5,7,9); // length uzunluğu 3

*countNum*(5,7,8,11,13,15,17,19,21); // length uzunluğu 9

##### sumNums(1,2,3,4,5,6,7,8,9,10);

*printName*("Ali","Kahraman");

##### printName("Ayşe Yılmaz Kahraman"); // Ayşe Yılmaz Kahraman

}

##### public static void countNum(int... a) { // Vararg'ın ismi a

##### System.out.println(a.length);

##### }

##### public static void sumNums(double... b) { // Varargs'ın içine koyduğumuz tüm sayıları toplayan prog.

##### double sum=0;

##### for(double w:b) { // w 0.indexten başlayıp son indexe kadar uygular.

##### sum=sum+w;

##### }

##### System.out.println(sum);

##### }

##### public static void printName(String...n) {

##### for(String w:n) {

##### System.out.println(w+" ");

##### }}}

**package** day24varargsaccessmodifiers;

**public** **class** Varargs02 {

**public** **static** **void** main(String[] args) {

## VARARGS'IN YANLIŞ KULLANIMLARI

##### Kural 1: Varargs her zaman son parametre olmalıdır. çünkü varargs sonsuz elemana sahip for loop ta varargs bitmiyor ki diğer parametreye geçebilsin çünkü ilk parametreyi varargs yaparsanız girdinğiniz her değer varargs'ın içine düşer, siz asla ikinci parametreye değer verememiş olursunuz bu nedenle hata verir.

##### 

##### Kural 2: parametre olarak 1'den fazla varargs kullanamazsınız çünkü varargs en sonda olmalıdır, en sonda olmak tek olmayı gerektirir.

*valueChar*("Character",'a');

*valueChar*("Character",'b','c');

// product(1);

// product(3);

}

##### public static void valueChar(String s, char... c) {

##### for(char w: c) {

##### System.out.print(s);

##### System.out.println(w);

##### }

##### }

##### // public static void product(int... x, int y) { // yanlış

// int product=1;

// for(int w: x) {

// product=product\*w;

// System.out.println(product);

// }

##### // public static void product1(int... x, int... y){// yanlış

// int product1=1;

// for(int w: x) {

// product1=product1\*w;

// System.out.println(product1);

// }

}

# ACCESS MODİFİER

**package** day24varargsaccessmodifiers;

**public** **class** AccessModifier {

**public** **static** **void** main(String[] args) {

##### Bir class, constructor, variable, ve method’a erişimi kısıtlamak veya genişletmek icin Access Modifier’lar kullanılır.

## 4 tane Access Modifier vardır.

##### 1) private: Class'lar private olamaz. Class içinde Constructor, method ve variable için kullanılabilir.sadece class içindekiler tarafından kullanabilir.

##### 2) default (Package Private) herhangi bir Access Modifier yazmazsak,Java Access Modifier’i default olarak kabul eder. Sadece aynı package içindeki classlar tarafından görülüp ulaşılabilir.

##### 3) protect: Class'lar protected olamaz. Aynı package içindeki classlar ve diğer package'lardan child Classlar tarafından görülüp ulaşılabilir.

##### 4) public (herkese açık):class’a, method’a, variable’a veya constructor’a her yerden ulaşılabilir,herhangi bir kısıtlama koymaz.

}

}

# 25.DERS ( StringBuilder )

**package** day25stringbuilder;

**public** **class** StringBuilder01 {

**public** **static** **void** main(String[] args) {

## StringBuilder class'ı String üretmeye ve üretilen Stringleri manipüle etmeye yarar.

##### String class'ı kullanarak bir String üretirseniz eğer yaptığınız değişiklikler ilk ürettiğiniz Stringi etkilemez.Ama illaki ilk ürettiğim string değişsin isterseniz atama yapmak zorundasınız. String class'ını kullanarak ürettiğiniz stringler atamasız değiştirilemedeği için String class'ına "immutable" denir.

##### String ile uretilen Stringlerde yapilan degisiklikler ilk uretilen (orijinal olan) Stringi degistirmezken StringBuilder ile uretilen Stringlerde yapilan degisiklikler ilk uretilen orijinal Stringi degistirir. Yani Stringlerde degisim ilk Stringi etkilemez ama StringBuilderde etkilenir.StringBuilder Class'ını kullanarak String üretirseniz yaptığınız değişiklikler ilk ürettiğiniz String'i etkiler. Atama yapmaya gerek yoktur. StringBuilder Class'ı değişimi yansıttığı için ona "mutable" denir.

##### String str="Ali";

##### str.substring(1); // ==>li

##### System.out.println(str);// ==> Ali yazar çünkü atama yapılmadı.

str=str.substring(1);

System.***out***.println(str);// li yazar. çünkü atama yapıldı.

##### atama yapmadan değişiklik uygulansın istiyorsanız StringBuilder Class'ı kullanılmalıdır.

##### StringBuilder strBld=new StringBuilder("Ayşe");

## strBld.append("Can"); // append () methodu String'e yeni bir String eklemek için kullanılır. Concatenation yapar.

##### System.out.println(strBld);// AyşeCan

// 2.YOL

StringBuilder strBld2=**new** StringBuilder();

strBld2.append("Ali Can"); // boş String'e "Ali Can" ekledim.

System.***out***.println(strBld2); // Ali Can

// 3.YOL

##### StringBuilder strBld3=new StringBuilder(7); // uzunluğu 7 chracter olan bir String ürettim.

##### strBld3.append("Ali");

##### strBld3.append("Kahraman");

##### System.out.println(strBld3); // AliKahraman

##### }

}

**package** day25stringbuilder;

**public** **class** StringBuilder02 {

**public** **static** **void** main(String[] args) {

##### substring() methodu atama yapmadan Stringi değiştirmez.

##### StringBuilder str1=new StringBuilder("animals");

##### str1.substring(3);

##### System.out.println(str1); // ekrana animals yazar

##### str1=str1.substring(3); altını çizer hata verir çünkü substring methodu String class'ından gelir ve bir String return eder. halbuki str1 String değil, StringBuilder dır. java data typeleri uyuşmadığından dolayı atamayı kabul etmez.Bu şekilde atama yapmaya izin vermez ama ekrana yazdırmaya izin verir.

// Bu hatadan kurtulmak için iki yol vardır.

// I.YOL : String classından yeni bir String üretip atama yapınız.

// II.YOL :str1.substring(3) ifadesini systemoutprintln(str1.substring(3)) //içerisine yazarak yazdırılabilir

String str2=str1.substring(3);

System.***out***.println(str2);

System.***out***.println(str1.substring(3));

System.***out***.println(str1.substring(1,4));

// indexOf methodu beLli bir character'in index'ini return eder

System.***out***.println(str1.indexOf("m")); // 3

// length() methodu String Class'ından gelir ve String'in uzunluğunu return eder.

System.***out***.println(str1.length()); // 7

// charAt() methodu belli bir indexteki characteri return eder.

System.***out***.println(str1.charAt(5)); // animals da index i 5 olan l harfidir . l return eder.

## insert () methodu istenen indexe istenen characteri ekmeleye yarar.insert () methodu append() gibi StringBuilder'i direkt değiştirir.

##### 

##### str1.insert(0, "X");

##### System.out.println(str1); // Xanimals

## delete() methodu istenen index aralığındaki characteri siler.

##### str1.delete(0,1); // StringBuilder classından geldiği için atama ihtiyacı duymadan direk değiştirir.

## deleteCharAt(); istenen indexteki characteri siler.

##### str1.deleteCharAt(6);// s harfini silmeli

##### System.out.println(str1);// animal

## reverse() Stringi tersten yazdırır.

##### str1.reverse();

##### System.out.println(str1);// lamina

## toString() methodu StringBuilder'i String'e çevirmek için kullanılır.

##### str1.toString();

##### System.out.println(str1);// str1 artık StringBuilder değil String'dir.

//StringBuilder Java'nın 5.versiyonunda oluşturuldu.

//StringBuffer'lar StringBuilder'ların eski versiyonudur.

##### Stringbuilder'lar daha hızlı çalışır.Bu yüzden StringBuilder kullanmayı tercih edelim.

}

}

**package** day25stringbuilder;

**public** **class** Sorular {

**public** **static** **void** main(String[] args) {

// String s1="java";

// StringBuilder s2=new StringBuilder("java");

//

// if(s1==s2) System.out.println("1"); // farklı data typleri eşit mi diye sorulamaz

// if(s1.equals(s2));

// System.out.println("s2");

// the code does not compile

}

}

# 26.DERS ( ENCAPSULATION, INHERİTANCE )

# ENCAPSULATION

**package** day26encapsulation;

**public** **class** Encapsulation01 {

**private** String kimlikNo="34298567399";

**private** **int** krediKartNo= 12345678;

**private** **boolean** soguk = **true**;

**public** **static** **void** main(String[] args) {

##### Programcılıkta herşeyi herkesin görmesi iyi değildir.Program kodları herkese açık olunca başkaları içine girip değişiklik yapıp programı bozabilir.Program yazarken bazı şeyleri mesela variable'lerı saklamak isteriz.Bu saklama işlemi başkalarına kapalı iken bizim için açık, ulaşılabilir olmalı.

## Encapsulation data saklama(data-hiding) yöntemidir. Encapsulation iki stepte yapılır.

##### 1) Data'yı(Variable, method) private yapmalısınız.Bu şekilde başkaları ulaşamaz.

##### 2) bu sakladığım dataya kendim ulaşabilmem için public olan getter() ve setter() methodlar üretmeliyim. getter()methodu data'yı sadece okumamıza yarar, getter() methodu data'da değişiklik yapamaz. setter() methodu data'yı değiştirmemize yarar.

## GETTER() METHODU ÜRETMEK İÇİN;

##### 1) Access Modifier public olmalı.

##### 2) Return type'i variable'in return type'i ile aynı olmalı.

##### 3) Method ismi "get+variable ismi" şeklinde olmalı return type boolean ise getter() method ismi "is" ile başlar "get" kullanılmaz.

##### setter() methodu; Datayı değiştirmek için kullanılır. void iki yerde kullanılır. konsola yazdırma ve action'larda kullanılır. void kullanılınca return kullanılmaz. setter() methodun data type'i her zaman void'dir. action için. parametre yerine mutlaka birşey yazılmalıdır.

## SETTER() ÜRETMEK İÇİN;

##### 1) Access modifier public olmalı

##### 2) Return type void olmalı.

##### 3) Method ismi "set+variable ismi" şeklinde olmalı

##### 4) Parametre kullanılmalı()

##### 5) setter() methodların ismi boolean'lar içinde set ile başlar.

##### public String getKimlikNo() {

##### return kimlikNo;

##### }

**public** **int** getKrediKartNo() {

**return** krediKartNo;

}

##### public boolean isSoguk() {

##### return soguk;

##### }

##### public void setKimlikNo(String kimlikNo) {

##### this.kimlikNo =kimlikNo;

##### }

**public** **void** setKredikartNo(**int** krediKartNo) {

**this**.krediKartNo = krediKartNo;

}

##### public void setSoguk(boolean soguk) {

##### this.soguk = soguk;

##### }

}

**package** day26encapsulation;

**public** **class** Encapsulation02 {

**private** **char** ilkHarf = 'S';

**private** **double** para = 23.75;

**private** **boolean** emekli = **false**;

## GETTER() VE SETTER() METHODLARINI KISA YOLDAN OLUŞTURMAK

##### variable "ilkHarf" üzerine sağ tıkla Source==>Generate Getter and Setter... tıkla

##### Variable'a atanan değerin okunmasını istemiyorsanız o variable'la alakalı getter() oluşturmamalısınız. Variable'a atanan değerin değiştirilmesini istemiyorsanız o variable ile alakalı setter() oluşturmamalısınız.

##### 

##### mesela kendi email adresimizi girdik ve başkasının okumasını görmesini istiyor, değiştirmesini istemiyorsak sadece getter() kullanır hiç Setter() kullanmazsanız setter() işaretlenmezse variable değerleri değiştirilemez demektir. Bu tarz Class'lara "Immutable Class" denir. Sadece Setter() kullanır Getter() kullanmazsanız okunamaz ama değiştirilebilir demektir.

##### 

##### başka bir class'taki variable'a object oluşturarak ulaşabilirim.

**public** **static** **void** main(String[] args) {

##### Encapsulation01 obj = new Encapsulation01();

##### System.out.println(obj.getKimlikNo()); // başka klasstaki variable okuduk.

##### obj.setKimlikNo("10000000"); // başka bir class'taki variable değiştirdik.Bu değişiklik sadece bu Class'a özgüdür. asıl Classta değişiklik olmaz.

System.***out***.println(obj.getKimlikNo());

}

**public** **boolean** isEmekli() {

**return** emekli;

}

**public** **void** setEmekli(**boolean** emekli) {

**this**.emekli = emekli;

}

**public** **char** getIlkHarf() {

**return** ilkHarf;

}

**public** **void** setIlkHarf(**char** ilkHarf) {

**this**.ilkHarf = ilkHarf;

}

**public** **double** getPara() {

**return** para;

}

**public** **void** setPara(**double** para) {

**this**.para = para;

}

}

# INHERITANCE

**package** day26encapsulation;

**public** **class** Inheritance01 {

**public** **static** **void** main(String[] args) {

##### Inheritance kelime manası olarak Miras demek. İki Class arasında Parent - Child ilişkisi varsa inheritance'dan bahsedilebilir. Childlar tarafından parent Class'taki variable'lar kullanılabilir. Aynı package'da ise default Access modifier lar parent Class'lardan variable kullanabilir.

## Inheritance Kuralları:

##### 1) Inheritance Public ve Protected data'lar için problemsiz çalışır.

##### 2) Inheritance default data'lar(variable'lar) için child ile parent aynı package'da ise çalışır.

##### 3) Inheritance Private data'lar için çalışmaz.

## Child ve Parent ilişkisi nasıl kurulur?

##### Class oluştururken "extends" keyword kullanırız.

}

}

**package** day26encapsulation;

##### Inheritance01, Inheritance02'nin parent'ı.

// Normalde alttaki satır bu şekilde idi " public class Inheritance02 {"

// onu bu şekle getirdik "public class Inheritance02 extends Inheritance01 {"

##### extends'den sonraki class parent, önceki class child olur.

##### Bir parentin bir çok childi olabilir (hierarchical) Inheritance bir childin iki parenti olamaz.java multiple inharitance'a müsade etmiyor (single inheritance)

##### Multilevel Inheretance: child==>parent Class==>parent Class çocuk dede ilişkisi gibi cocuğu babaya extends yapıyoruz sonra babayı dedeye extends yapıyoruz.

##### public class Inheritance02 extends Inheritance01 {

**public** **static** **void** main(String[] args) {

//

}

}

**package** day26encapsulation;

**public** **class** Animal {

**public** **int** age;

**protected** String name;

**public** **static** **void** main(String[] args) {

}

**public** **void** move() {

System.***out***.println("Hareket edebiliyor");

}

**public** **void** eat() {

System.***out***.println("Yemek yer");

}

}

**package** day26encapsulation;

**public** **class** Mammal **extends** Animal {

**public** **boolean** dogum = **true**;

**public** **static** **void** main(String[] args) {

}

**public** **void** feed() {

System.***out***.println("Çocuklarını besler");

}

}

**package** day26encapsulation;

**public** **class** Dog **extends** Mammal {

// Dog ==> Mammal ==> Animal

// Dog için bark, feed, move, eat methodları kullanılabilir hale geldi.

**public** **boolean** sadik = **true**;

**public** **static** **void** main(String[] args) {

Dog dog = **new** Dog();

System.***out***.println("Animal Class'dan geldi"+dog.age);

System.***out***.println("Animal Class'dan geldi"+dog.name); // Stringler için default değer null

System.***out***.print("Animal Class'dan geldi: ");

dog.eat();

System.***out***.print("Animal Class'dan geldi: ");

dog.move();

System.***out***.println("Mammal Class'dan geldi: "+ dog.dogum); // true

dog.feed(); // çocuklarını besler

System.***out***.println("Dog Class'dan geldi: "+dog.sadik);// true

dog.bark();

}

**public** **void** bark() {

System.***out***.println("Köpek havlar");

}

}

**package** day26encapsulation;

**public** **class** Cat **extends** Mammal {

// Cat Class'ının kullanabileceği methodlar: meow, feed, move, eat

// Dog Class'ının kullanabileceği methodlar: bark, feed, move, eat

// kardeşler arasında Inheritance olmaz. Dog Class'ında meow methodu kullanılamaz.

**public** **boolean** clean = **true**;

**public** **static** **void** main(String[] args) {

Cat cat=**new** Cat();

System.***out***.println("Animal Class'dan geldi: "+cat.age);

System.***out***.println("Animal Class'dan geldi: "+cat.name);

cat.move();

cat.eat();

System.***out***.println("Animal Class'dan geldi: "+cat.dogum);

cat.feed();

System.***out***.println(cat.dogum);

cat.meow();

}

**public** **void** meow() {

System.***out***.println("Kediler miyavlar");

}

}

# 27.DERS ( CONSTRUCTOR CALL )

##### inheritenceda main method içerisinde object oluşturulunca önce grandparent sonra parent'ın sırasıyla constructorlarını kullanır ve çalıştırır. kaç tane üst parent varsa hepsini return eder. altta constructor oluşturulunca "Dog();" parentlerda "()" parametresiz ne kadar constructor varsa hepsini çağırıyor. en büyük atasından aşağıya doğru çalışır.

##### dog: object ismi Dog():Constructor ismi

##### main methodu içine yazılan methodlar çalışır, yazılmayanlar çalışmaz.

**package** day27constructorCall01;

##### public class Dog extends Mammal {

##### 

##### public boolean sadik = true;

##### public static void main(String[] args) {

##### 

##### Dog dog = new Dog();

##### 

##### }

##### public void bark() {

##### System.out.println("Köpekler havlar");

##### }

##### 

##### Dog(){ // Parent'daki parametresiz constructor'ı çağır demektir.

##### System.out.println("Dog parametresiz constructor");

##### }

##### 

##### }

**package** day27constructorCall01;

**public** **class** Mammal **extends** Animal {

**public** **boolean** dogum = **true**;

### 

**public** **static** **void** main(String[] args) {

Mammal mammal = **new** Mammal();

Mammal mammal01 = **new** Mammal(15);

}

**public** **void** feed() {

System.***out***.println("Çocuklarını besler");

}

Mammal(){ // Parent'daki parametresiz constructor'ı çağır demektir.

## super();

##### // super() keyword'unu kullanırsanız tekrar parenta git demektir.super() kullanılacaksa mutlaka ilk satırda olmalıdır. super() parenta git demektir.super() ve this() aynı constructor içinde aynı anda kullanılamaz. çünkü ikisininde ilk satırda olma zorunluluğu var. aynı class içindeki parametresiz constructorı çağırmak için kullanıyoruz.

System.***out***.println("Mammal parametresiz constructor");

}

##### Mammal(int age){ // inheritance olduğu için önce Animal Class'a gider Animal'da parametreli constructor olmadığı için aşağı satıra iner

##### super(); // tekrar parenta yani Animal Class'ına gider orada parametresiz constructor araştırır varsa parametresiz constructor'ı çalıştırır.

##### System.out.println("Mammal parametreli constructor");

##### }

}

**package** day27constructorCall01;

**public** **class** Animal {

**public** **int** age;

**protected** String name;

**public** **static** **void** main(String[] args) {

Animal animal = **new** Animal();

}

**public** **void** move() {

System.***out***.println("Hareket edebiliyor");

}

**public** **void** eat() {

System.***out***.println("Yemek yer");

}

Animal(){

System.***out***.println("Animal parametresiz constructor");

}

}

# 28.DERS ( OVERRİDİNG )

**package** day28overriding;

**public** **class** Overriding {

**public** **static** **void** main(String[] args) {

##### 1)Parent'taki method'u method signature'ini degistirmeden method body'sini degistirerek Child class'da kullanmaya "Method Overriding" denir.

##### 2)Parent class'daki private method'lar override edilemezler

##### 3)final method'lar override edilemezler cunku; final methodlarin body'leri olabilecekleri son haldedir yani method body'leri degistirilemez. Halbuki overriding'in amaci method body'i degistirerek methodu kullanmaktir. Bu celiskiden dolayi Java final method'larin override edilmesine musaade etmez.

##### 4)static method'lar override edilemezler.Cunku; static methodlar ortak kullanima aciktir onun body'sini degistirmek herkesi etkiler bu yuzden Java static methodlarin override edilmesine musaade etmez.

##### 5)Child class'daki override edilmis method'un (Overriding Method) access modifier'i Parent class'daki override edilen method'un(Overridden Method) access modifier'indan daha dar olamaz.

##### 6)Child class'daki override edilmis methodun return type'i ya parent class'daki override edilmis methodun return type'i ile ayni olur veya onun child'larindan biri olabilir

##### 

##### 

##### Overriding: uyarlamak demektir.

##### Parent'taki bir methodu Child Class'ında kendimize uyarlayarak kullanmamıza Overriding deniyor.

// overriding yaparken method ismi ve parametreler değişmemeli.

// asla bu signature değiştirilmez.

// Overriding'de method içerisindeki bodyler { } değiştirilebilir.

##### Parent içindeki method (ses) . bu methoda Method Overridden denir. Child içerisindeki method ses( ). bu methoda Method Overridding Method.

// 4 Access Modifier var. private, default, protected, public.

##### Overriding yaptığımızda child in access modifier ile aynı olmalı veya daha geniş olmalıdır.Parent Access modifier default olduğunda, Child Access modifier default, protected, public olabilir.Child Parenta göre durum alır.Parent public olduğunda child default olamaz. Child parent'ı daraltamaz, sınırlandıramaz.

## IS-A VE HAS-A

##### Child dan parenta giderken İs-a ilişkisi, parent'dan child'a gidersek has-a ilişkisi vardır.

##### Dog IS-A Mammal == True Mammal HAS-A Dog ==> True

##### Return typeları kullanırken bu is-a ve has-a ilişkisi olması lazım. Overriding'de is-a ilişkisi olması lazım. Child daki return type child olması lazım.

## OVERRİDİNG KURALLARI

##### 1) Method signature'ı(name ve parameters) değiştirmeyin.

##### 2) 4 Access Modifier var. private => default => protected => public. Overriding yaptığımızda child in access modifier ile aynı olmalı veya daha geniş olmalıdır.

##### 3) Overriding COVARİANT return type olmalıdır. yani İS-A olmaldırı. Child'dakinin return type parent'takinin return type'nın child'i olmalıdır.

##### 4) private,static and final methodları overriding yapılamazlar. private olan birşey o Class'a özel bir şeydir,Class'a özel olan başka Class'ta kullanılamaz.private overloading yapılabilir.static bütün objelere açıktır, herkese hitap eden bir özelliği var bu özelliği değiştiremezsiniz.final method artık son haline getirilmiş, tamamlanmış, değişime açık değil demektir. overriding olmaz.

##### methodun başına final yazılarak kullanılır.

## OVERRİDİNG KONTROLÜ

##### 1) parent child ilişkisini kontrol et. extends yazıyor mu?

##### 2) method ismi aynımı

##### 3) parametreler aynı mı

##### 4) Accesse Modifier aynı olmalı veya kapsamalı

##### 5) return type'lar int is-a void ? değil false compilation error verir.

##### 

##### Parametresiz Constructor kullanılmışsa parent varsa önce parent'a gider oradaki parametresiz constructor'ı çalıştırır.

//

// public void ses() { )// Parent içindeki method. bu methoda Overriding denir.

// System.out.println("Hayvanlar ses Çıkarır");

// }

//

// public void ses() {

// System.out.println("Kediler miyavlar");

// }

//

// public void ses() {

// System.out.println("Köpekler havlar");

// }

}

}

**package** day28overriding;

**public** **class** Animal {

**public** **int** age=4; // instance variable

**public** String name="Karabas"; // instance variable

**public** **static** **void** main(String[] args) {

}

**public** **void** move() {

System.***out***.println("Hayvanlar hareket eder");

}

}

**package** day28overriding;

**public** **class** Mammal **extends** Animal {

**public** **boolean** birth = **true**;

**public** String name = "Kedi";

**public** **static** **void** main(String[] args) {

##### Mammal mammal = new Mammal();

##### önce parente gider parametresiz constructor arar orada birşey olmadığından işlem yapamaz döner Mammal() Constructora gider this.birth ü yazdırır.

// sonra feed methoduna gider, Yavrularını besler yazdırır.

// sonra Animal'daki age variable'ına gider onu yazdırır.

// . yazıp methodlar çıktığında yanında parantez işareti varsa

// bu methoddur. yoksa bu variable dır. ayrıca yan tarafında

// bu method veya variable'ın hangi Class'ta olduğunu görebiliriz.

// sonra move methodunu yazdırır.

}

**public** **void** feed() {

System.***out***.println("Yavrularını besler");

}

## THİS VE SUPER KULLANIMI

parent Class Animal'dada variable name var, bu Class'dada name var.this kullanınca normalde ikisinide göstermesi lazım

ama iki taraftada aynı isimli olunca this öncelikli olarak içinde bulunduğu class'dakini gösterir.

this hem içinde bulunduğunuz class hem parent class'larraki method ve variable'lara ulaşmamızı sağlar.

##### public Mammal() {

##### System.out.println(this.birth); // true

##### this.feed();

##### System.out.println(this.age);

**this**.move();

System.***out***.println(**this**.name);

##### System.out.println(super.age); // 4

##### System.out.println(super.name); // Karabas

##### super.move(); // Hayvanlar hareket eder.

##### Parantezsiz super ise sadece parent'lardaki variable ve methodlara ulaşmanızı sağlar.

##### Parentteki variable veya methodlara super kullanarak ulaşmak daha güvenlidir. çünkü child ve parent'lardaki aynı isimli variable veya methodlarda problem yaşayabilirsiniz.bu nedenle genel kullanımdak this class içindekiler için, super ise parent içindekiler için kullanılır.

}

}

# 29.DERS ( POLYMORPHISM)

**package** day29polymorphismexception;

**public** **class** Polymorphism01 {

**public** **static** **void** main(String[] args) {

// Poly: çoklu demek poliklinik gibi

##### Polymorphism : çoklu yapı demek.

##### Bir metodu çoklu yapı haline getiriyorsanız polymorphism yapıyorsunuz demektir.Bunun için Overloading ve Overriding kullanılır.Polymorphism'i Overloading ile yaparsanız "Compile Time Polymorphism" denir.Diğer adı Static Polymorphism'dir.

##### Polymorphism'i Overriding ile yaparsanız "Run Time Polymorphysm" denir.Diğer adı Dinamic polymorphism'dir.

}

##### public void eat() {

##### System.out.println("Ye!");

##### }

##### 

##### Parametreyi değiştirerek Overlaoding yapıp Polymorphism yaptık.

##### public void eat(String name) {

##### System.out.println(name+ "Ye!");

##### }

**class** Yeni **extends** Polymorphism01{

##### Body'yi değiştirerek Overriding yapıp polymorphism yaptık.

##### public void eat() {

##### System.out.println("Lütfen Ye!");

##### }

}

}

# EXEPTİONS

**package** day29polymorphismexception;

**public** **class** Exception01 {

**public** **static** **void** main(String[] args) {

##### ﻿Java code çalıştırılırken, farklı error’lar oluşabilir:

##### 1) Code’u yazan hatalı yazmış olabilir

##### 2) Kullanıcı hatalı data girişi yapıyor olabilir

##### 3) Öngörülemeyen hatalar oluşabilir

##### Hata oluştuğunda Java çalısmayı durdurur ve console’a bir hata mesajı yazdırır.

##### Buna exception atma manasına gelen “throw an exception” denir.

##### Exception: istisnai problemler demektir.mesela sayılar birbirine bölünebilir ama sadece 0'a bölümünde hata verir. Bu problemli duruma exception diyoruz. Bazı problemler vardır, halledilebilir, handling exeption, bazılarıda halledilemez throwing exeption. yardım istemeye throw denir.

## Temelde iki türlü exception var.

##### 1)Exeption'ların bir kısmı Compile Time Error (Checked Exeption)

##### ﻿Checked Exception’lar kesinlikle code yazılırken halledilmelidirler. Yanlış code yazıldığında altı kırmızı olarak çizilir. Compile Time Error'lara Checked Exeption'da denir.

##### 2) Bir kısmıda Run Time Error'dur.

## Compile Time Error (Checked exeption)'a örnek:

##### 1) FileNotFoundExceptions Programıma bir dosyayı almak istediğimde dosya bulunamadığı zaman verilen hatadır.

##### 2) IOException: input output exception yazdığım programıma dışardan bir şey almak veya dışarıya bir şey yollamak istediğimde bu hatayı verir.

##### Not: FileNotFoundException, IOException'ın Child'ıdıdır.

}

}

**package** day29polymorphismexception;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**public** **class** Exception02 {

# Compile Time Error(Checked Exception)'lar nasıl halledilir?

## 1) Method isminden sonra "throws" keyword kullanılır.

##### "throws" keywordunu kullanırsanız konsolda teknik mesajlar görürsünüz. Bu yüzden çok tercih edilmez.

##### public static void main(String[] args)throws FileNotFoundException {

## Yazdığımız bir programın içerisine bir dosya eklemek istediğimizde FileInputStream Class'ını kullanıyoruz.

##### FileInputStream file = new FileInputStream("C:/ogrenci.txt");

##### 

##### // e.printStackTrace(); OTOMATİK OLUŞAN BU SATIRI SİLİYORUZ. ALTTAKİNİ YAZIYORUZ

##### }

##### FileInputStream() parametresini boş bırakırsanız FileInputStream Class'ının parametresiz Constructor'ı olmadığından altını çizer exception verir içine "C:/ogrenci.txt" yazınca altını yine çizer. altına gidip "add throws" linkini tıklıyoruz. main methoda "throws FileNotFoundException" yazısını ekleyecektir.çalıştırınca altta bir sürü hata mesajları gösterecek. bunlar yabancı birileri için readable olmadığı için bu 1. yolu çok tercih etmiyoruz.

## 2) try-catch block kullanılır.

##### ﻿1)“try block” exception üretme ihtimali olan code içerir.

##### 2) “try block”dan sonra her zaman “catch block” yazılır, “catch block” exception’i handle eder.

##### 3) “try block”dan sonra 1’den fazla “catch block” kullanilabilir, istenirse “finally block” da kullanilabilir. “finally block” exception oluşsa da oluşmasa da çalışır.

##### ﻿4) Program yazarken bazı code’ların hata üretebileceğini öngörüyorsanız bu code’u try block’un icine koyup hatayı catch block kullanarak yakalayabilirsiniz.

##### ﻿5) Bir try block 1’den fazla catch block içerebilir. Her catch block’da farklı bir exception yakalayabilirsiniz.

##### ﻿a) Eğer exception ilk catch block’da yakalanırsa sonraki catch block’lar çalışmaz.

##### b) Sonraki catch block’lar kesinlikle öncekilerin parent’i olmalıdır. Aksi takdirde Compile Time Error alırsınız.

##### try-catch block kullanmak daha iyidir, çünkü readable'dır. Bunu kullanırsanız konsolda teknik mesajlar yerine kendi mesajlarımızı görebiliriz. Bu yüzden try-catch kullanmak tercih edilir.

##### throws : yardım iste (denemeden yardım iste) throw yapılırsa java mesajarı kendi yazar, bu çok readable olmuyor.

##### try-catch blocks: (try:)dene eğer hata çıkarsa (catch:) onu yakala ve konsola yazdır.lastiği değiştirmeyi dene yapamazsan bana yaz.

##### try kullanılırsa ekrana çıkacak hatayı siz yazarsınız daha anlaşılır olur.

##### FileInputStream() parametresini boş bırakırsanız FileInputStream Class'ının parametresiz Constructor'ı olmadığından altını çizer exception verir içine "C:/ogrenci.txt" yazınca altını yine çizer. altına gidip "add try" linkini tıklıyoruz.

##### // try {

##### // FileInputStream file = new FileInputStream("C:/ogrenci.txt");

##### // } catch (FileNotFoundException e) {

##### yazacaktır.

##### // e.printStackTrace(); OTOMATİK OLUŞAN BU SATIRI SİLİYORUZ. ALTTAKİNİ YAZIYORUZ

##### // System.out.println("Dosya bulunamadı, Allah bilir kim sildi");

##### Çalıştırınca bu yazı çıkacaktır. biz bir kod yazdığımızda muhtemel hatayı tahmin ederiz. }

// yeni bir Class oluşturuyoruz.

##### class ExceptionTryCatch{

##### public static void main(String[] args) {

##### try {

##### FileInputStream file = new FileInputStream ("C:/ogrenci.txt");

##### } catch (FileNotFoundException e) {

##### System.out.println("Dosya'nın path'i yanlış veya dosya silinmiş olabilir");

##### // e.printStackTrace(); BU SATIRI SİLİYORUM

##### class ismi ExceptionTryCatch üstüne gidip sağ tıkla run as git ==> java yı seç altta try yazanı seç ve tamam de.

##### üstteki mesaj değilde "Dosya'nın path'i yanlış veya dosya silinmiş olabilir"); mesajı altta mesaj olarak paylaşılır.

}

}

}

**package** day29polymorphismexception;

**public** **class** Exception03 {

**public** **static** **void** main(String[] args) {

##### ArithmeticException Run Time Exception'dir. Matematik kurallarina uygun olmayan bir islem yapildiginda bu exception alinir.

##### 

##### try'dan sonra 1'den fazla catch kullanilabilir.Ama ustteki catch alttaki catch'in child'i olmalidir. catch'leri Specific'den Genel'e dogru dizmelisiniz yoksa Compile time Error alırsınız.

//Bolme islemi yapan bir program yazalim

##### int num1 = 6;

##### int num2 = 0;

##### try { // try:dene

##### System.out.println(num1/num2);

##### }catch(ArithmeticException e) { // catch: yakala

##### System.out.println("Sifir ile bolme yapilmaz");

##### }catch(Exception e) { // Exception: parametrenin data type'i e: ismi

##### System.out.println("Her exception'i yakala");

}

}

}

# 30.DERS( EXEPTİONS )

**package** day30exceptions;

**public** **class** CommonRunTimeException {

**public** **static** **void** main(String[] args) {

# COMMEN RUN TİME EXCEPTİONS

## ArithmeticException:

##### System.out.println(12/0);

## ArrayIndexOutOfBoundsException :

##### illegal index durumunda, olmayan index istendiğinde

##### // int arr[] = {a,b,c} System.out.println{arr[12]); olmayan 12.indexi yazdır diyor.

## ClassCastException :

##### String'i integer'a çevirme.

##### // String name = "Ali";

##### // Integer number=(Integer) name;

##### Run Time Exception verir.

## IllegalArgumentException:

##### bir insanın yaşından bahsediyorsunuz -3 diyorsunuz.

##### // if(yumurtasayısı<0){

##### // throw new IllegalArgumentException("yumurta sayısı negatif olmamalı")

##### normalde java bunun hatalı olduğunu anlamaz biz throw new ile hata mesajı oluşturuyoruz.

## NullPointerException:

##### String name = null;

##### // public void printLength(){

##### system.out.println(name.length()); }

##### String'e null atandığı veya String instance variable'a değer atanmadığı zaman length() methodu çalışmaz ve NullPointerException alırsınız.

## NumberFormatException:

##### int num = Integer.parsenlnt("abc");

##### Bir String'i integera çevirmeye çalışırken ortaya çıkan exception'dır.

}

}

**package** day30exceptions;

**public** **class** ThrowSorusu01 {

**public** **static** **void** main(String[] args) {

**try** {

*hop*(); // RuntimeException yakalar çünkü onun parent'ıdır.

}**catch** (Exception e) {

System.***out***.println(e); // burada e exception adıdır. e yazdığımız için

// altta java.lang.RuntimeException yazar.

}

}

**private** **static** **void** hop() {

**throw** **new** RuntimeException("cannot hop");

// cannot hop cevap

}

}

## getMessage: eException'ı yakala sadece mesajı yazdır.

**package** day30exceptions;

**public** **class** ThrowSorusu02 {

##### public static void main(String[] args) {

##### 

##### try {

##### hop();

##### }catch (Exception e) {

##### System.out.println(e.getMessage()); // getMessage eException'ı yakala sadece mesajı yazdır.teknik hata mesajını yazma demektir. sadece e yazılsaydı java.lang.RuntimeException yazacaktı.

##### }

##### }

##### 

##### private static void hop() {

##### throw new RuntimeException ("cannot hop");

##### }

}

## e.printStackTrace(); hangi satırlarda Exception aldındığını gösterir.

**package** day30exceptions;

**public** **class** ThrowSorusu03 {

##### public static void main(String[] args) {

##### 

##### try {

##### hop();

##### } catch ( Exception e) {

##### e.printStackTrace(); // bunu kullanırsak hangi satırlarda Exception aldığımızı gösterir. Exception hataları oluşum sırasına göre consolda gösterir.

##### // at day30exceptions.ThrowSorusu03.hop(ThrowSorusu03.java:19)

##### // at day30exceptions.ThrowSorusu03.main(ThrowSorusu03.java:8)

##### 

##### }

##### }

##### private static void hop() {

##### throw new RuntimeException ("cannot hop");

##### }

}

**package** day30exceptions;

**public** **class** TryCatch01 {

##### e yazmazsak altını çizer. ArithmeticException olası aritmatik tüm hataları yakalar.istediğimiz kadar catch kullanabiliriz tek şartla yukarıdaki alttakinin child'i olmak şartıyla.

##### catch'den sonra istersek finally yazabiliriz.finally{} block Exception olsa da olmasa da her zaman çalışır.finally{} block mutlaka try-catch block'dan sonra kullanılır.

##### public static void main(String[] args) {

##### 

##### int num1 = 12;

##### int num2 = 0;

##### 

##### try {

##### System.out.println(num1/num2);

##### }catch(ArithmeticException e) {

##### 

##### 

##### 

##### System.out.println("Sıfır ile bölme yapılamaz");

##### }finally {

##### System.out.println("İşleminiz bitmiştir");

##### }

## INTERVEİW SORUSU:

## final, finally, finalize keyword'larının farkı nedir?

##### 1) finally{} (block) ==> try-catch block'dan sonra kullanılır. Exception olsa da olmasada çalışır.

##### 2) final; (keyword) ==> Variable'larda final kullanılırsa, o variable'ın değeri değiştirilemez.mesela pi sayısı final yapıldığı zaman değeri değiştirilemez.

##### // final String code = "POS";

##### Method'larda final kullanılırsa, o methodun{} body kısmı değiştirilemez.yani Overriding yapamam demektir. Inheritance mümkün değildir. Class'larda final kullanılırsa o Class extends edilemez.(parent olamaz)

##### // public class Animal{} ==> public class Dog extends Animal{};

##### // hatalı==> public final class Animal{}==> public class Dog extends Animal{} olmaz.

##### 3) finalize() (method) ==> finalize() methodu, Garbage Collector'ın imha edeceği dataları imha edilecek hale getirir. çöpleri topla sonra atarım.

## Garbage Collection

Garbage Collection erişilemeyen nesneleri silerek belleğin verimli bir şekilde kullanılmasını sağlamaktadır. Konuyu örnekler üzerinden açıklamaya devam edelim;

Ogrenci ogr = new Ogrenci();

Yukarıdaki kod satırında “*new Ogrenci()*” komutu ile Heap bellek bölgesinde Ogrenci sınıfından bir nesne oluşturduk. Bu nesneye erişebilmek için adresini, belleğin Stack bölgesinde tutulan “*ogr*” ismindeki işaretçiye aktardık. Herhangi bir şekilde “*ogr*” isimli işaretçiyi kaybedersek, Heap bellek bölgesinde oluşturduğumuz nesneye erişmemiz mümkün olmayacaktır. Erişilemeyen nesnelerde Garbage Collection tarafından silinerek bellekte gereksiz yer kaplaması önlenecektir.

##### Ogrenci ogr1 = new Ogrenci();

##### Ogrenci ogr2 = new Ogrenci();

##### ogr1 = ogr2;

##### Bu örneğimizde Ogrenci sınıfından iki adet nesne oluşturduk. 1. ve 2. satırlarda ilk oluşturduğumuz nesnenin adresini “ogr1” işaretçisine, ikinci oluşturduğumuz nesnenin adresini “ogr2“ işaretçisine aktardık. 3. satırda ise ikinci nesnenin adresini, ilk nesnenin adresini tutan “ogr1” işaretçisine aktardık. Yani artık ogr1 işaretçisi, ilk nesnenin adresini değil ikinci nesnenin adresini tutmaktadır. İlk oluşturulan nesnenin adresi kaybedildiği için artık o nesneye ulaşmak mümkün olmayacaktır. Nesneye erişimin mümkün olmadığı durumlarda nesnenin bellekte kalmasının bir anlamı yoktur. Bu yüzden Garbage Collection tarafından silinecektir.

**package** day30exceptions;

**public** **class** TryCatch02 {

**static** String *o*;

##### public static void main(String[] args) {

##### 

##### String s = "";

##### System.out.println(s.length());

##### 

##### try { // dene bunu çalıştır her yerde kullanılabilir ama genel kullanım exception ihtimalinde kullanılıyor.

##### s+="t";

##### }catch(Exception e) {

##### s+="c";

##### }finally {

##### s+="f";

##### }

##### s+="a";

##### 

##### System.out.println(s);

##### 

##### String n = null;

##### String m;

##### 

##### // String'e null atandığı veya String instance variable'a (o) değer atanmadığı zaman length() methodu çalışmaz ve NullPointerException alırsınız.NullPointerException Run Time Exception'dır. n bir bir objedir, obje için uzunluk ölçülemez. String m bir methodun içinde olduğu için local variable'dir mutlaka değer atanması lazım yoksa hata verir.

##### 

##### // System.out.println(n.length());

##### // System.out.println(m.length());

##### // System.out.println(o.length());

}

}

**package** day30exceptions;

**public** **class** TryCatchl03 {

## INTERVİEW SORUSU:

## throw ==>

##### java bir satıra geldiğinde normalde hata vermese bile burada hata yap diyebiliriz, bunun için throw kullanırız. mesela girilmemesi gereken arazilere mayın döşenir, aynı şekilde bir satıra girilmemesi isteniyorsa orada hata versin isteyebiliriz, bunun için throw kullanılır.

## throw , throws keywordlerinin farkı :

##### 1) throw, method body'si içine yazılır, throws method parentezi ile curly brace'in {} arasına yazılır. throw'u method body'si içinde istediğimiz yerde kullanabiliriz.

##### 2) throw'u method body'si içinde exception almak istediğimiz yerde kullanabiliriz. throws ise mecburen method isminden sonra bir kere kullanılabilir.

##### 3) bir throw sadece bir tane exception üretir. bir throws ile birden fazla exception üretebilirsiniz yani "throws NullPointerException, ArithmeticException" mümkün.

##### 4) throw için syntax ==> throw new NullPointerException;

##### throws için syntax ==> throws NullPointerException

##### public static void main(String[] args) {

##### 

##### System.out.println(exceptions());

##### }

##### 

##### @SuppressWarnings("finally") // 48,49,50. satırların altını sarı renk çiziyordu bunu yazarak kaldırdık.

##### public static String exceptions() {

##### String result = "";

##### String v = null;

##### 

##### try {

##### try {

##### result += "before";

##### v.length(); // null uzunluğu istenince NullPointException hatası alınır ve alt satıra geçmeden satır 29'a geçer.

##### result += "after";

##### }catch( NullPointerException e) {

##### result += "catch";

##### throw new RuntimeException();

##### // burada Exception oluştursun diyor ama altta onu yakalayacak catch yok bu nedenle bir alt satıra geçer.

##### }finally {

##### // alttaki üç satırın altını sarı renk çiziyordu üstte @SuppressWarnings("finally") yazarak bunu kaldırdık.

##### result +="finally";

##### throw new Exception(); // yeni bir Exception oluşturur ve bunu alttaki catch yakalar.

##### }

##### }catch(Exception e) { // Bu parent Exception dır. satır 31'u kapatırsak

##### // bu sefer bu

##### result += "done";

##### }

##### return result; // result'u çalıştırır, return'u ekrana yazdırır.

##### // method'da void kullanılmadığı için return

##### }

##### }

##### Konsolda “beforecatchfinallydone” yazar

# Throwable

**package** day31abstraction;

**public** **class** Throwable {

##### Throwable: atılabilir demektir. bütün exception'ların ortak faaliyetlerinin bulunduğu classtır. java bazen exception bazen de error diye consola hata yazar.

## Out of Memory Error:

##### Javada kullandığımız iki memory var Stack ve Heap bunlandan birisi dolarsa Java memory'ye bir şey koyamaz ve Out of Memory Error verir. hafıza dolmuş demektir

## Error'lar handle edilemez, çözülemezler. Error ile Exception arasındaki fark budur.

##### Error'lar Run Time Error'dur.

## StackOverflowError :

##### Stack memory'den cok fazla data çağırımı yapılırsa memory kilitlenir ve bu hatayı verir. bu nedenle kodun kısalığı Stack ve heap memorydeki yükü azaltır.

## System Crash Error:

##### kullanılan sistemin çökmesinde ortaya çıkan Error'dur.Bu Errorlar kodumuzun dışında ortaya çıkan arızalardır.

##### birden fazla catch görüyorsanız child parent ilişkisini soruyordur muhtemelen

}

# 

# 31.DERS( ABSTRACTION )

##### Abstract: soyut Concrete: somut-beton

##### 

##### abstract class'lar obje üretemeyen class'lardır. Bir Abstract Class'dan obje üretilemez, Constructor'ları yoktur.

##### public abstract class Animal(){} ==> Animal animal = new Animal();

##### // new Animal altını çizer hata gösterir. obje üretilemez çünkü abstract method {} body si olmayan methoddur.

##### body si olmayan methodlara abstract method denir.bizim şu ana kadar bildiğimiz method lar body si olan Concrete methodlardı ikisi arasındaki fark birisinde body si var ne yapılacağı tarif ediliyor. diğerinde body si yok ne yapılacağı tarif edilmiyor. yapılacak işin ismi söylenir nasıl yapılacağından bahsedilmez.

##### Abstract class concrete ve abstract child'i olur ama child'in içinden parent'daki abstract methodlar mutlaka override edilmelidir.

##### override: bir methodun body si değiştirilerek yapılır.

##### Child abstract ise herhangi bir override zorunluluğu yok. araba için abstract class oluşturdunuz, motorun çalışması gibi zorunlu methodlar vardır. bütün arabalar motoru çalışır olması lazım, bunun için Child Class'lar override ederek mecburen motor methodunu kullanıp bunu çalıştırması istenir yoksa child olarak kabul etmez.

## abstract olunca bu method mutlaka bulunmak zorunluluğu koyulmuş oluyor.

##### concrete class ın icine abstract method konmaz,sadece concrete konur. Abstract Class'lar concrete(body'li) ve abstract(body'siz) methodlar içerebilir.

##### abstarct methoda body koymaya calısırsam CompiletimeError alırım.

##### concrete methoda body koymazsam CompiletimeError alırım.

## Abstract Class'ların hem concrete class hem de abstract Class olan child'leri olabilir.

## Abstract Class'ın child'i concrete class ise Abstract class'daki abstract methodalrı mutlaka override etmeli ve onlara body eklemelidir.

## // ==> bunun faydası şudur; child Class'larda mecburen olması gereken methodları bu şekilde belirlemiş oluruz.

##### Abstract Class'ı child'i abstract Class ise Abstract Class'daki herhangi bir şeyi override etmek zorunda değildir.

##### Java’da kalıtım(inheritence) bir sınıfın kendine ait özellikleri başka bir sınıfa aynen aktarması ya da bazı özelliklerin kullanılmasına diğer sınıflarda izin vermesidir.

##### package day31abstraction;

##### public abstract class Abstraction01 {

##### public static void main(String[] args) {

##### 

##### }

##### public void concreteMethod() {

##### System.out.println("Ben concrete'im");

##### }

##### public abstract void abstractMethod();

##### }

## Bir methodun bir class'ta mutlak surette çalışmasını istiyorsak bu yolu izleriz.

##### Abstraction01 abstract class oldugundan herhangi bir override zorunlulugu yok. Compile Time Error almiyorum. ama Abstraction02 concrete Class olduğundan ve bu Class’ı abstract bir Class olan Abstraction01’in Classı yapmak için yanına extends Abstraction01 yazarsam altını çizer ve hata verir. Abstract bir Class içinde abstract bir method var bunu inherit etmediğimizden hata veriyor. bu methodun inherit edilmesi lazım.(abstract methodların body si olmuyor, bu methoda body ekleyerek methodu concrete methoda dönüştürme)

##### bunun için yazının üzerine gidip Add unimplemented methods

##### tıklayınca aşağıdaki methodu body si ile birlikte kendisi oluşturuyor.dikkat ederseniz bu methodu oluşturmadan, kullanmadan bu Class'ı child olarak kullanmamıza izin vermiyor. Honda parent'i bir child class'ına motor methodu olmadan çalışmana izin vermem demek gibi birşey.

**package** day31abstraction;

##### public class Abstraction02 extends Abstraction01{

##### 

##### public static void main(String[] args) {

##### Abstraction02 obj = new Abstraction02();

##### obj.abstractMethod();

##### obj.concreteMethod();

##### }

##### @Override

##### public void abstractMethod() {

##### 

##### }

##### }

**package** day31abstraction;

**public** **abstract** **class** Abstraction03 **extends** Abstraction01 {

**public** **static** **void** main(String[] args) {

// Abstraction03 abstract Class olduğundan herhangi bir override zorunluluğu yoktur.

// Compile Time Error almıyorum.

// Bu Clas abstract olduğu için obje oluşturamam

}

}

# 32.DERS ( ABSTRACTİON )

**package** day32abstraction;

**public** **abstract** **class** Otel {

// Abstract Class'ın her child'inde oluşturulan abstract method mutlaka kullanılması gerektiğinden

// ve bu method Class'a göre uyarlanabilir olduğundan body konulmayıp bu body içeriği Class'larda

// oluşturalacak inheritence metodlara bırakılmıştır.

// herhangi bir parent tarafından override edilen abstract method alttaki Concrete child'lar

// tarafından override edilmesede olur.

##### bir Class'ı final yaptığımız zaman daha onun child'i olamaz.

##### Bizim abstract Class oluşturmamızın sebebi içine methodlar koyalım o metodlarıda child Class'lar kullansın amaçlı. bu nedenle abstract Class'lar final olamazlar. aynı şekilde abstract methodlar final edilemezler. abstract method private da olamaz, private methodlar override edilemezler.

**public** **static** **void** main(String[] args) {

}

**public** **abstract** **void** kahvalti();

**public** **int** wifi(**int** kişiSayisi) {

**return** kişiSayisi\*2;

}

}

**package** day32abstraction;

**public** **class** AileOdasi **extends** Otel {

**public** **static** **void** main(String[] args) {

AileOdasi musteri2 = **new** AileOdasi();

musteri2.kahvalti();

System.***out***.println("Wifi ücreti: "+ musteri2.wifi(6));

}

@Override

**public** **void** kahvalti() {

System.***out***.println("Kahvaltınız Cafeteria'da olacaktır.");

}

}

## ﻿Abstract Class Kurallari:

##### 1. Bir Abstract Class’dan object oluşturulamaz.

##### 2. Abstract class’lar abstract ve concrete method’lar icerebilir.

##### 3. Abstract class private ve final olamazlar.

##### 4. Bir abstract class başka bir another abstract class’a extends ettiginde abstract

##### method’lari override etmek zorunda degildir.

##### 5. Abstract class’a extends eden ilk concrete class abstract class’daki tum abstract

##### method’lari override etmek zorundadir.

## Abstract Method Kurallari:

##### 1. Abstract method’lar sadece abstract classlar icinde olusturulabilir.

##### 2. Abstract methodlar private ve final olamazlar.

##### 3. Abstract method’larin method body’si olamaz.

**package** day32abstraction;

**public** **class** KralDairesi **extends** Otel {

**public** **static** **void** main(String[] args) {

KralDairesi musteri3 = **new** KralDairesi();

musteri3.kahvalti();

System.***out***.println("TV Ücreti: "+musteri3.tv("international", 5));

System.***out***.println("Wifi ücreti: "+musteri3.wifi(3));

}

**public** **void** kahvalti() {

System.***out***.println("Kahvaltınız teras kat Kral Restaurant'dadır.");

}

**public** **int** tv(String international, **int** saat) {

**if**(international.equals("international")) {

**return** saat\*5;

}**else** {

**return** 0;

}

}

}

**package** day32abstraction;

**public** **class** TekKisilikOda **extends** Otel {

**public** **static** **void** main(String[] args) {

TekKisilikOda musteri1 = **new** TekKisilikOda();

musteri1.kahvalti();

System.***out***.println("TV Ücriti: "+musteri1.tv(5));

System.***out***.println("Wifi Ücreti: "+musteri1.wifi(1));

}

@Override

**public** **void** kahvalti() {

System.***out***.println("Kahvaltınız sadvictir");

}

**public** **int** tv(**int** saat) {

**return** saat\*2;

}

}

# INTERFACE

##### Interface Class'a benzer ama Class değildir. Abstract Classların içine hem abstract hem constract method koyabiliyorduk, Interface'e sadece abstract method konulabilir.

## INTERFACE METHOD

##### interface'de method oluştururken kesinlikle body yazmayın interface'deki methodlar kesinlikle public ve abstract olmalıdır.interface'deki abstract methodları üretirken abstract keyword kullansanızda olur kullanmasanızda olur.

##### çünkü interface kendi içine yazılan methodları otomatik olarak abstract olarak kabul eder.

##### public void add(); veya public abstract void add();

## INTERFACE VARIABLE

##### Interface içerisine variable koyabaliriz ama bu

##### variable'lar mutlaka public static ve final olmalıdır. bu kuralı dinlemez variable'ı private veya protected yaparsanız Compile Time Error alınır.

##### interface içindeki variable'ları mutlaka initialize etmek(değer atamak) zorundasınız, aksi takdire Compile Time Error alırsınız.

##### int a=12; gibi yapılmalı . interface'de variable'lar initialize, public, static, final olmalı ve mutlaka initialize edilmelidir. initialize : değer atanmalıdır. Variable oluştururken public,static,final keywordları kullanılsada olur kullanılmasada olur. kullanılmasada kendisi default olarak public, static, final olarak atar.

##### package day32abstraction;

##### public interface Interface01 {

##### public void add();

##### public abstract void subtract();

##### 

##### public static final int a = 12;

##### int b = 11;

##### public int c = 13;

##### final int d = 14;

##### static int e = 15;

##### }

# 33.DERS ( INTERFACE )

##### Interface: arayüz demektir.

##### Interface Access modifier sadece public ve default olabilir.

##### multipleInheritence mümkün değil yani Java'da bir child'in birden fazla parenti olamaz ama bazen bir child'in birden fazla parenti olması gerekebilir bu sorun Java'da Interface ile çözülmüş. bir çok parent'e ihtiyac duyduğumuzda Interface Animal, Interface Mammal yapıyoruz.böylelikle concrete dog child extends olarak Animal ve Mammal'e ulaşabiliyor.

##### body {} si olan methodlara concrete method, body'si olmayan methodlara abstract method denir.

##### Interface kullanımının diğer bir sebebi yapılması gereken zorunlu methodlar Interface içerisine yazılır, Interface içerisine isteğe bağlı methodları koyamıyoruz, mecburi olan methodlar yazılır.

##### bir child'i birden fazla Inheritence bağlamak için implements yazıp Inheritence isimleri virgül ekleyerek yazıyoruz.

##### public class LuxHonda extends Araba implements IcAraba, DisAraba {

##### Class ==> Interface : implements Class'ları Interface'e bağlamak için implements

##### Class ==> Class : extends Class'ları Class'a bağlamak için extends

##### Interface ==> Interface : extends Interface'leri Interface'lere bağlamak için extends

##### Farklı Interface'lerde aynı isimde farklı Return type'lerdeki metodlar yazılırsa Compile Time Error verir.

##### Farklı Interface'lerde aynı isim ve aynı Return type'lerdeki metodlar olursa implements durumunda sorun olmaz.

##### Farklı Interface'lerde Variable'lar için data type ve isimler aynı ama değerler farklı ise implements durumunda bu variable'lara Interface isimleri ile ulaşabiliriz.

##### Bir variable final ise java onu mavi olarak yazar, onu değişteremeyeceğimi anlarım.

##### Normalde Inheritence içinde sadece abstracat method olabilir demiştik ancak java 8 den sonra bir özellik eklenmiş default ve static ekleyerek {} body li concrete method üretebilme imkanı tanımıştır.

##### public default void döseme() {

##### System.out.println("Interface'de default keyword ile döseme yaptım");

##### interface interface'e child olacaksa extends kullanılır

##### interface'lerin Access modifier'i public ve default olabilir, protected ve private olamaz.

**package** day33interface;

**public** **abstract** **class** Araba {

**public** **abstract** **void** move();

**public** **void** diesel() {

System.***out***.println("Diesel 2.0 Turbo");

}

**public** **void** benzin() {

System.***out***.println("Benzin 3.5 Turbo");

}

}

**package** day33interface;

**public** **interface** DisAraba {

**int** ***price*** = 3000;

**void** kapi();

}

**package** day33interface;

**public** **interface** IcAraba {

// 3 şekildede yazılabilir. methodlar public ve abstract olmak zorunda

**int** ***price*** =2000; //Bir variable final ise java onu mavi olarak yazar, onu değişteremeyeceğimi anlarım.

**boolean** ***old*** = **true**;

**public** **abstract** **void** direksiyon();

**public** **void** koltuk();

**void** klima();

## public default void döseme() { // obje kullanarak child'de bu methodu çağırabilirim.

## System.*out*.println("Interface'de default keyword ile döseme yaptım");

## }

## 

## public static void isitma() { // direk Class ismi ile herhangi bir Class'tan bu methodu çağırabilirim.

## System.*out*.println("Static ile yan ayna isitma özelliği koyduk");

## }

}

**package** day33interface;

**public** **class** BasicHonda **extends** Araba **implements** DisAraba,IcAraba {

**public** **static** **void** main(String[] args) {

BasicHonda basicHonda = **new** BasicHonda();

basicHonda.diesel();

basicHonda.direksiyon();

basicHonda.kapi();

basicHonda.klima();

basicHonda.koltuk();

basicHonda.move();

}

@Override

**public** **void** direksiyon() {

System.***out***.println("Hidrolik direksiyon");

}

@Override

**public** **void** koltuk() {

System.***out***.println("Kumaş koltuk");

}

@Override

**public** **void** klima() {

System.***out***.println("Manuel Klima");

}

@Override

**public** **void** kapi() {

System.***out***.println("Dört kapılı");

}

@Override

**public** **void** move() {

System.***out***.println("Önden çekişli");

} }

**package** day33interface;

**public** **class** LuxHonda **extends** Araba **implements** IcAraba, DisAraba {

**public** **static** **void** main(String[] args) {

LuxHonda luxHonda = **new** LuxHonda();

luxHonda.benzin();

luxHonda.direksiyon();

luxHonda.kapi();

luxHonda.klima();

luxHonda.koltuk();

luxHonda.move();

## System.*out*.println("Ic araba Interface'ınden"+IcAraba.*price*); // iki ayrı interfacede price isminde variable olduğundan interface ismi ile variable’ı çağırdık.

## System.*out*.println("Dış araba Interface'ınden"+ DisAraba.*price*);// 3000

## System.*out*.println(*old*); // iki tane old isimli variable olmadığından interface adını kullanmadan variable’ı çağırdık.

## 

## luxHonda.döseme(); // Inheritence'daki bodyli default methodu obje oluşturarak çağırdık

## IcAraba.*isitma*(); // Inheritence'daki bodyli static methodu obje oluşturmadan class ismi ile çağırdık

}

@Override

**public** **void** move() {

System.***out***.println("3 saniyed 100 km'lik hıza ulaşır");

}

@Override

**public** **void** kapi() {

System.***out***.println("Parmak izi ayarlı kapı");

}

@Override

**public** **void** direksiyon() {

System.***out***.println("Gerçek deri direksiyon");

}

@Override

**public** **void** koltuk() {

System.***out***.println("Isıtmalı deri koltuk");

}

@Override

**public** **void** klima() {

System.***out***.println("Digital Klima");

}

}

# 34.DERS ( COLLECTIONS(Interface) )

##### Collections: yığın, birikim demektir.

##### Javada temelde üç tane list var. List, Queue, Set.

##### Collection bu üçünün parent'ı durumunda. Java her zaman ortak özellikleri parent'a koyar.

##### Iterable Collection Interface'ının parentıdır. değiştirilebilir demektir ve bunların hepsi Interface'dır.

##### Child Interface ise parent Class olamaz.

##### Queue sıra demek.

##### Iterator bir collection'ın elemanları arasında gezer ve o elemanları değiştirir. for each loop Collection elemanları arasında gezer ama değiştiremez.

##### Collection olarak biz sadece List'i öğrenmiştik

##### List<String> list1 = new ArrayList<>(); yazdığımızda hem List hem de ArryList'in altını cizer

##### List ayrı bir Interface ArrayList ayrı bir Class, Iterator Javada bir Interface'dir.

## .iterator methodu başka bir objeyi interface'e uygun hale getirmektir.

##### interface'de normalde Constructor olmaz, obje oluşturulamaz ama iterator methodu sayesinde başka yerde oluşturulmuş objeyi interface'de kullanılabilir haline getiriyor.

## .hasNext methodu bir sonraki elaman var mı yok mu, kontrol eder, eleman varsa true atar.

## System.out.print(iterator.next());// yazdırmak için next methodunu kullanmamız gerekiyor.

## .next methodu iterator'a git pointer’i bir sonraki elemanın öncesine götür demektir.

##### Eğer siz bir Iterator'da değişimin kalıcı olmasını istiyorsanız Iterator kullanmak durumundasınız, for-each loop elamanlar arasında gezer ama Collection'ı değiştiremez. iterator ise elemanlar arasında gezer ve istenirse Collection'ı değiştirir.

##### Data type ya kendisi gibi yada parenti gibi olmalı, obje oluşturabilmesi içinde class olmalı

##### SortedSet obj2 = new TreeSet(); obj2 objesi TreeSet Class'ında oluşturuluyor, SortedSet onun parenti

**package** day34collections;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**public** **class** IteratorVeForEachLoop {

##### public static void main(String[] args) {

##### 

##### List<String> list1 = new ArrayList<>();

##### list1.add("X");

##### list1.add("Y");

##### list1.add("Z");

##### System.out.println(list1);// [X, Y, Z]

##### 

##### Iterator<String> iterator = list1.iterator();

##### // .iterator methodu başka bir objeyi interface'e uygun hale getirmektir.

##### // Tipi Iterator interface olan iterator isimli bir obje oluşturduk.

// bu objeyi list1 kullanarak üretmiş olduk.

// while(iterator.hasNext()) { //.hasNext methodu bir sonraki elaman var mı yok mu, kontrol eder.

// yukarıdaki while de kendisinden sonra eleman var olana kadar

// true eder eleman bitince false der çıkar.

// System.out.print(iterator.next());// yazdırmak için next methodunu kullanmamız gerekiyor.

// }

System.***out***.println();

// for-each loop kullanarak list1'in tüm elemanlarını ekrana yazdırınız.

**for** (String w: list1) {

System.***out***.print(w); //XYZ

}

System.***out***.println();

System.***out***.println();

// for-each loop ile değişim yapmaya çalışalım

**for**(String w: list1) {

w = w+"A";

}

System.***out***.println(list1);// [X, Y, Z]

// for-each loop elamanlar arasında gezer ama Collection'ı değiştiremez

// iterator ise elemanlar arasında gezer ve istenirse Collection'ı değiştirir.

System.***out***.println();

// Iterator ile for-each loop farkı:

**while**(iterator.hasNext()) {

iterator.next(); // .next methodu iterator'a git bir sonraki elemanı yazdır demektir.

iterator.remove(); // remove methodu eleman siler.

}

System.***out***.println(list1); // []

}

}

# 35.DERS ( COLLECTİONS )

# LISTITERATOR

##### ListIterator Iterator'a göre daha zengin daha çok method içeriyor.

## iterator'larda elemanlar arasında gezmek için her zaman While'ı kullanıyoruz.

## .next()bir sonraki elemana geç return et demektir.

## .hasPrevious pointer'ı en sondan başlayıp geriye doğru bir önceki elemana bak varsa true ver demektir.

##### öncelikle .hasNext'i kullanıp pointi sona getiriyoruz.en sona gelen point .hasPrevious ile kaldığı yerden devam edip geriye doğru gider. .hasNext kullanmadan .hasPrevious kullanırsak en baştan başlar geride hiçbirşey olmadığından devam etmez.

##### .hasPrevious() ve previous() methodlarını kullanarak bir list'in elamanlarını tersten yazmak isterseniz önce hasNext() ve next() kullanmalısınız.

## list oluşturmadan ListIterator oluşturup ona eleman ekleyemiyorum.

**package** day35collections;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.ListIterator;

**public** **class** ListIteratorMethods01 {

**public** **static** **void** main(String[] args) {

// elemanları A,B ve C Stringleri olan bir list oluşturunuz

##### List<String> list = new ArrayList<>();

##### 

##### list.add("A");

list.add("B");

list.add("C");

System.***out***.println(list);

## list iterator oluşturma

##### ListIterator<String> listIterator = list.listIterator();

##### while(listIterator.hasNext()) {

##### Object element = listIterator.next();

## // data type herşeye uygun olsun diye object yaptık.

##### System.out.print(element + " ");

##### }

System.***out***.println();

##### while(listIterator.hasPrevious()) {

##### Object element = listIterator.previous();

##### System.out.print(element + " ");

##### }

}

}

**package** day35collections;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.ListIterator;

**public** **class** ListIteratorMethods02 {

**public** **static** **void** main(String[] args) {

// Elemanları A,B ve C olan bir list oluşturunuz

// list iterator kullanarak bu elemanları ve AW, BW ve CW ye dönüştürünüz.

List<String> list = **new** ArrayList<>();

list.add("A");

list.add("B");

list.add("C");

System.***out***.println(list);

ListIterator<String> listIterator = list.listIterator();

// list oluşturmadan ListIterator oluşturamıyorum.

## set methodu bu indextekinin yerine bunu ata,uygula demektir.

##### while(listIterator.hasNext()) {

##### Object harf = listIterator.next();

##### listIterator.set(harf + "W");

##### }

// iterator'da yapılan değişiklikler kalıcı oluyor.

System.***out***.println(list);

list.add("Kemal");

list.add("Can");

System.***out***.println(list);

}

}

# LINKEDLIST(Class)

##### LinkedList bir Class'tır. LinkedList'te koyduğumuz her elemanın sağ tarafında bir pointer varmış gibi düşünüyoruz bu pointer onu sağ tarafındaki elemana bağlar.linked zaten bağlanmak demektir.bu şekilde tren vagonları gibi her eleman bir sonraki elemana bağlıdır.

## Çok fazla eleman silmemiz veya eleman eklememiz gereken durumlarda LinkedList kullanılır. Eleman ekleme ve çıkarma durumlarında LinkedList başarılıdır.

##### LinkedList'te head hariç her eleman için bir data bir de pointer var.Head ilk eleman, head'de sadece pointer vardır.

##### Her eleman pointer'i bir sonraki elemanı gösterir, son elemanın pointer'ı(tail) pointer'i null(boş) gösterir. . A . B . C . . Z . null ilk pointe head diyoruz,A'yı gösterir. son noktaya tail diyoruz null gösterir.

##### Data ve pointer içeren her elemanın diğer adı "node" dur. " . A " birlikte bir node'dur.

// LinkedList objesi oluşturunuz

**package** day35collections;

**import** java.util.LinkedList;

**public** **class** LinkedList01 {

##### public static void main(String[] args) {

##### 

##### LinkedList<String> linklist = new LinkedList<>();

##### linklist.add("Mark");

##### linklist.add("Amanda");

##### linklist.add("John");

##### linklist.add("Ann");

##### linklist.add("Pamela");

##### System.out.println(linklist); // [Mark, Amanda, John, Ann, Pamela]

##### 

## linklist.remove(2);

System.***out***.println(linklist);

## linklist.removeFirst();

System.***out***.println(linklist);

## linklist.removeLast();

System.***out***.println(linklist);

## linklist.add(1, "Ali");

System.***out***.println(linklist);

## linklist.addFirst("Kemal");

System.***out***.println(linklist);

## linklist.addLast("Zeynep");

System.***out***.println(linklist);

## linklist.set(1, "Ajanda");

System.***out***.println(linklist);

LinkedList<String> linklist2 = **new** LinkedList<>();

linklist2.add("X");

linklist2.add("Y");

## linklist2.addAll(linklist); // iki listi birbirine ekliyoruz

System.***out***.println(linklist2);

## linklist.addAll(3, linklist2);

System.***out***.println(linklist);

## System.*out*.println(linklist2.contains("Can"));// True veya False return eder.

## linklist2.clear(); // clear methodu listin elemanlarını siler, sizde boş bir list kalır.

System.***out***.println(linklist2);

}

}

# SET(Interface)

**package** day35collections;

**public** **class** Set01 {

##### Set interface'i duplication'a(tekrarlı elemanlara) müsaade etmez. Set içinde aynı elemandan iki tane olamaz.

## Set'i tekrarlı eleman bulunmasını istemediğimiz zaman kullanırız.

##### HashSet, LinkedHashSet ve TreeSet Class'ları Set interface'inin childleridir ve Set'te tekrarlı elemanlar kullanılamadığından dolayı bu Class'lardada hiç bir surette bu Class'larda tekrarlı eleman olmaz. Set gördüğümüzde no dublication diye düşünübeliriz.

##### TreeSet aynı zamanda SortedSet interface'inin child'didir.

## SortedSet interface'inin özelliği elemanlarının sıralanmış olmasıdır.

##### TreeSet bu interface'in child'i olduğundan inheritence'dan(miras) dolayı aynı özellik TreeSet Class'ında da mevcuttur.

## Hem tekrarlı olmasın, hemde sıralı olsun istiyorsak TreeSet Class'ını kullanırız.

## 3 tane bilmemiz gereken Set var.

##### 1) HashSet(No dublication)

##### 2) LinkedHashSet(No dublication)

##### 3) TreeSet(No dublication+Sorted)

}

# 36.DERS ( COLLECTIONS )

# HashSet(Class)

##### 1) Tekrarlı eleman kullanımına(Duplication) izin vermez.

##### 2) Key - Value yapısını kullanır. Key'ler ve Value'lar HashSet için unique'dir(benzersiz). normalde Key'ler her zaman unique'dir. Value'lar kullanılan Class'a bağlı olarak unique’de olabilir, olmayadabilir.

## Kullanılan Class HashSet ise Value'larda unique olur ama ArrayList ise unique olma zorunluluğu yoktur.

##### Key'i java kendi üretir, tekrarı bir benzeri yoktur, aynı şekilde vaule'larında benzeri olmamalı. Key-Value'lara en güzel örnek sözlüklerdir. Bir kelimenin manası bir kere yazılır başka yazılmaz. kelimeler Key'dir, Value'ler ise tekrar kullanılabilir.

##### yürek = kalp, gönül = kalp gibi... bu nedenle mesela dijital bir sözlük oluşturduğunuz da ArrayList makul olur HashSet makul olmaz yürek için kalp yazdığınızda gönül için kalp yazamazsınız yani unique'dır.

## HashSet'i ekrana yazdırdığımızda key ve value'yu birlikte görürüz.

## [Dil=kalp, Yürek=Kalp, Gönül=Kalp, Yüz=Sayı]

##### mesela öğrenci numarasını girdiğinizde bu keye uygun sadece ilgili kişiyi bulur.

##### Bizim her yazdığımız karaktere Java bir reference değeri atar, buna Hashcode diyoruz.

##### uniqe numara: herkesin essiz bir numarasi var. mesela universite numarasi 9017004 90(yil)17(bolum)004(kazanma sirasi)console de cikan referanslar javanin hashing yontemi buna hashcode denir.

##### Java bizim verdiğimiz her value için hashing tekniğini kullanarak bir unique oluşturur.key her zaman unique dir.

## Normalde List'lerde add() methodu hep yeni eklenen elemanı sona ekler ama HashSet'lerde öyle bir zorunluluk yok.HashSet'ler elemanları ekrana rastgele yazdırır, kural yoktur.

##### Java arka planda her vaule'ya verdiği key unique değerine göre bir reference atar ve ona göre bir sıraya dizer. buda random rastgele olur.

##### HashSet'te bir Value'ya aynı değer eklenmek istendiğinde Compile Time veya Run Time Error alınmaz. son yüklediğimizi eskisinin üzerine yazar, tekrara izin verilmez.

**package** day36collections;

**import** java.util.HashSet;

**public** **class** HashSet01 {

**public** **static** **void** main(String[] args) {

##### Class Data type constructor ismi

##### HashSet<String> hSet = new HashSet<>();

##### hSet.add("Apple");

hSet.add("Mango");

hSet.add("Grape");

hSet.add("Orange");

hSet.add("Fig");

// aynı eleman HashSet'e eklenmek istendiğinde Compile Time veya Run Time Error alınmaz.

// son eklenen eleman ilk eklenen elemanın üstüne yazılır.

hSet.add("Apple");

System.***out***.println(hSet);

## .hashCode() methodu bir valuye Java'nın atamış olduğu hashCode değerini çalıştırır.

##### System.out.println(hSet.hashCode());

}

}

# LinkedHashSet(Class)

##### 1) duplication'a izin vermez, key-value yapısı var.

##### 2) elemanları programcının girdiği sıraya göre dizer.

##### HashSet hepsinden hızlıdır. sıralama ile hiç uğraşmaz.

##### LinkedHashSet'te sizin yazdığınız sıraya göre dizmeye çalışır.

**package** day36collections;

**import** java.util.LinkedHashSet;

**import** java.util.TreeSet;

**public** **class** LinkedHashSet01 {

**public** **static** **void** main(String[] args) {

##### LinkedHashSet<String> lhSet = new LinkedHashSet<>();

##### lhSet.add("Z");

##### lhSet.add("A");

lhSet.add("D");

lhSet.add("B");

##### System.out.println(lhSet); // sıralama olmadan bizim girdiğimiz sıraya göre yazar.

LinkedHashSet<Integer> lhSet1 = **new** LinkedHashSet<>();

lhSet1.add(3);

lhSet1.add(1);

lhSet1.add(5);

lhSet1.add(0);

System.***out***.println(lhSet1);

##### LinkedHashSet'i TreeSet'e çevirme:

##### oluşturduğumuz HashSet'i TreeSet' constructor'ına parametre olarak koyup TreeSet'e çeviririz. böylelikle TreeSet'in elemanları natural order(sıralama) yapma özelliğinden istifade etmiş oluruz.

##### TreeSet<Integer> tSet = new TreeSet<>(lhSet1);

##### System.out.println(tSet);

}

}

# TreeSet(Class)

##### 1) HashSet ile hemen hemen aynıdır. Duplication'a müsaade etmez

##### 2) TreeSet elemanlarını natural order( sayı ise küçükten büyüğe, String ise a dan z ye) göre dizer.

##### 3) HashSet ile farkı: TreeSet elemanlarını natural ordera göre dizer. HashSet ise rastgele elemanları dizer.

##### 4) HashSet ile farkı: HashSet bu sıralama meselesinden dolayı daha hızlıdır.

##### 5) Natural Order'a sahip bir Set lazım olduğunda normalde TreeSet kullanılması lazımken yavaşlığından dolayı HashSet oluşturulur sonra bu HashSet TreeSet'e dönüştürülür böylelikle TreeSet'in hız problemi çözülmüş olur.Direk TreeSet oluşturma genellikle kullanılmaz.

## Sıralama benim için önemli olduğunda TreeSet kullanırım.

##### HashSet oluşturup Java'nın elemanların hızlı eklemesini sağlarız.

**package** day36collections;

**import** java.util.HashSet;

**import** java.util.TreeSet;

**public** **class** TreeSet01 {

**public** **static** **void** main(String[] args) {

##### TreeSet<String> tSet = new TreeSet<>();

##### tSet.add("ABC");

tSet.add("String");

tSet.add("Test");

tSet.add("Pen");

tSet.add("Ink");

tSet.add("Jack");

##### System.out.println(tSet); // alfabetik sırada yazdıracaktır.

## HashSet oluşturup Java'nın elemanların hızlı eklemesini sağlarız.

##### HashSet<String> hSet1 =new HashSet<String>();

##### hSet1.add("ABC");

hSet1.add("String");

hSet1.add("Test");

hSet1.add("Pen");

hSet1.add("Ink");

hSet1.add("Jack");

##### System.out.println(hSet1);// sıralama olmadan rastgele yazar.

## HashSet'i TreeSet'e çevirme:

##### oluşturduğumuz HashSet'i TreeSet' constructor'ına parametre olarak koyup TreeSet'e çeviririz. böylelikle TreeSet'in elemanları natural order(sıralama) yapma özelliğinden istifade etmiş oluruz.

##### TreeSet<String> tSet1 = new TreeSet<>(hSet1);

##### 

##### System.out.println(tSet1);

}

}

# Queue(Interface)

##### Queu'ya eklenen elemanlar listlerde olduğu gibi en sona eklenir. Queue'da silinen elemanlar en baştaki elemandan başlayarak silinir.

##### [1,2,3] ==> bu Queue'dan eleman silmeye başladığınızda önce 1 silinir. FIFO ==> meşhur tabirdir. first in first out. yemekhanelerde ilk gelen ekmeği önce bitirirler.

##### Queue bir interface'dir onun altında iki Class vardır. Birisi LinkedList, diğeri PriorityQueue dir.

##### Elemanlar natural order dizilsin istersek PriorityQueue Constructor'ını kullanırız yok eğer elemanlar girdiğimiz sıraya göre dizilsin istersek LinkedList Constructor'ını kullanırız.

##### .element() methodu ilk elemanı döndürür. Queue'da ilk eleman çok önemlidir.

##### remove() methodunda index yoktur ilk elemanı siler veya remove(object) kullanılırsa parametre içine yazılan elemanı siler.

##### .poll() methodu ilk elemanı siler.

##### remove() methodu boş Queue'lar için kulllanılırsa Exception verir. poll() methodu boş Queue'ler için kullanılırsa null verir. Dolu Queue'lar için ikiside aynı işi yapar ilk elemanı silerler farkları yoktur.

## PriorityQueue Constructor'ı elemanları natural order'a göre sıralar.

**package** day36collections;

**import** java.util.LinkedList;

**import** java.util.PriorityQueue;

**import** java.util.Queue;

**public** **class** Queu01 {

**public** **static** **void** main(String[] args) {

##### Queue<String> q = new PriorityQueue<>();

##### q.add("B");

##### q.add("A");

##### q.add("C");

##### System.out.println(q);

## Constructor olarak LinkedList kullanırsak girdiğimiz sıraya göre dizer.

##### Queue<String> qll = new LinkedList<>();

##### qll.add("B");

##### qll.add("A");

##### qll.add("C");

##### System.out.println(qll);

## remove methodu kullanımı

##### System.out.println("Silinen eleman: "+ q.remove());

##### // bu yazımda remove methodu sildiği elemanı return eder ve her zaman ilk elemanı siler.

##### System.out.println("Kalan q: "+ q); // geriye kalan elemanları görürürüz.

##### System.out.println(q.remove("C"));

##### // true verir, çünkü biz Java'ya hangi eleman silinmesi gerektiğini söyledik, Java true return ederek işlem tamam mesajı verir.

##### System.out.println(q); // [B]

##### System.out.println("Silinen eleman: "+ q.remove()); // B'yi silecektir.

##### System.out.println(q); // []

##### // System.out.println("Silinen eleman: "+ q.remove());

##### // remove methodu() boş Queue'lar için Exception verdiğinden bu kodu comment yaptık.

## .element() methodu ilk elemanı döndürür. Queue'da ilk eleman çok önemlidir.

##### System.out.println(qll.element());

System.***out***.println(qll);

## .poll methodu kullanımı

##### System.out.println(qll.poll()); // B ==> ilk elemanı sildi ve return etti.

##### System.out.println(qll);// [A,C]

##### System.out.println(qll.poll());// A

##### System.out.println(qll);// [C]

##### System.out.println(qll.poll());// C

##### System.out.println(qll);// []

##### System.out.println("Boş Queue için poll(): "+ qll.poll()); // null

}

}

# 37.DERS ( COLLECTIONS )

# Deque(Interface)

##### Açılımı Double ended Queue (iki uçlu Queue)

##### Queue'larda ekleme sona yapılır, silme bastan yapılır. Queue kullanırsanız eklemeyi hem başa hem de sona yapabilirsiniz.

##### Aynı şekilde silmeyi hem baştan, hem de sondan yapabilirsiniz. Dequeue'da FİFO(First In First Out) var, bunun yanında LIFO'da var.(Last In First Out)

## Queue'lara eleman olarak null eklenebilir ama Deque'lara null eleman olarak eklenemez.

##### Deque resizable'dır. Yani kapasitesi değiştirilebilir.

##### ﻿Deque LinkedList’den hizlidir.

## .pop() methodu Deque'larda baştaki elemanı siler ve baştaki elemanı return eder.

.offer() methodu en sona elemena eklemek için kullanılır.

##### sona ekleyince true döner. addLast() ekleyemeyince exception verir..offerFirst() methodu başa ekleyince true döner, addFirst() ekleyemeyince exception verir.

## .push() methodu en başa ekleme yapar. push() başa ekleyemeyince false döner.

**package** day37collections;

**import** java.util.Deque;

**import** java.util.LinkedList;

**public** **class** Deque01 {

**public** **static** **void** main(String[] args) {

Deque<String> dq = **new** LinkedList<String>();

dq.add("Son1");

dq.addFirst("Bas1");

dq.addLast("Son2");

dq.push("Bas2");// push() en başa ekleme yapar.

dq.offer("Son3"); // en sona elemena eklemek için kullanılır

dq.offerFirst("Bas3");

dq.offerLast("Son4");

System.***out***.println(dq); // [Bas3, Bas2, Bas1, Son1, Son2, Son3, Son4]

System.***out***.println(dq.pop());// baştaki elemanı siler ve return eder. Bas3

System.***out***.println(dq); // [Bas2, Bas1, Son1, Son2, Son3, Son4]

System.***out***.println(dq.removeLast()); // Son4

System.***out***.println(dq); // [Bas2, Bas1, Son1, Son2, Son3]

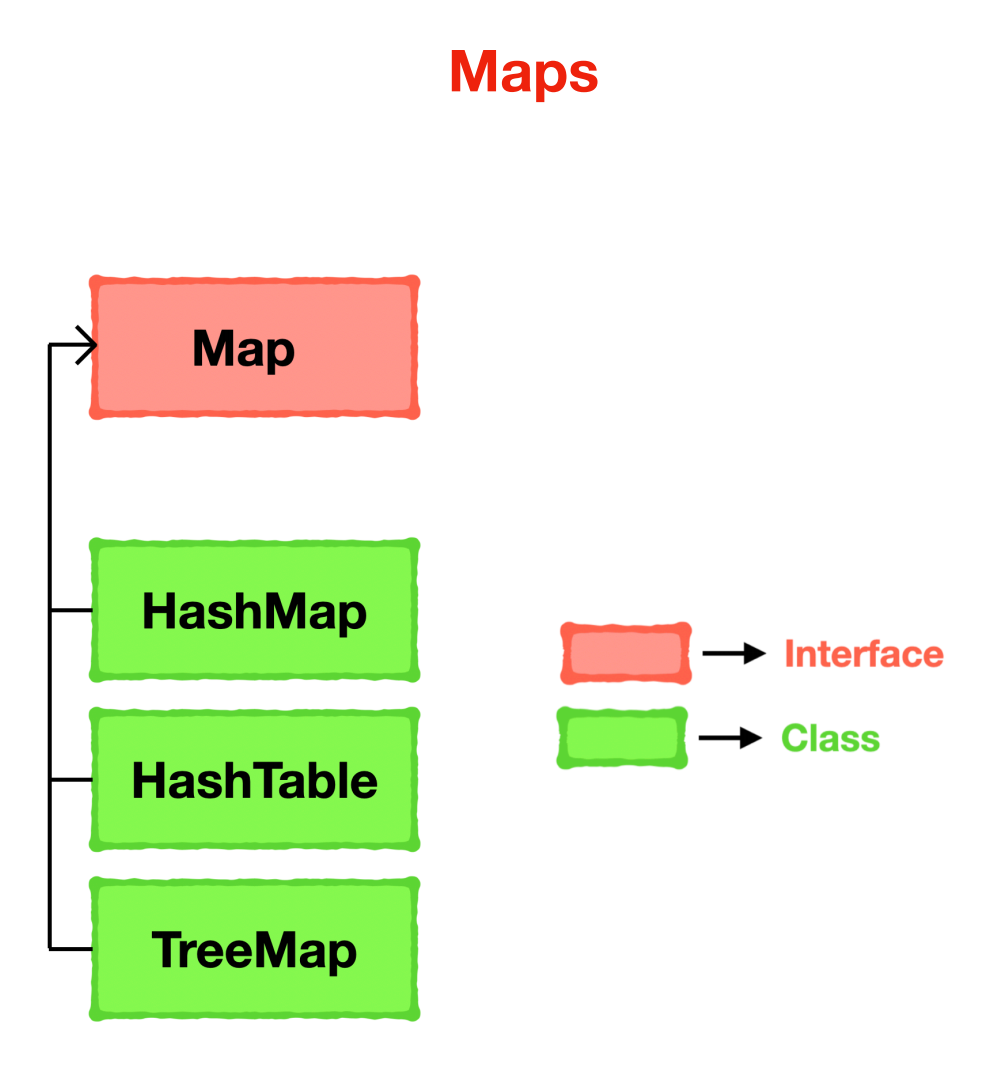
dq.removeFirst();

System.***out***.println(dq); // [Bas1, Son1, Son2, Son3]

}

}

# MAPS



##### Map bir interface'dir. Map interface'inin üç tane child Class'ı vardır. 1) HashMap 2) HashTable 3) TreeMap

##### 

##### HashMap

##### HashMap key-value yapısını kullanır.Key ve Value programcı tarafından yazılır.Key ve Value'da null değeri kullanılabilir. Key'de 1'den fazla null değeri kullanılırsa Java son kullanılan null'u kabul eder, value'da birden fazla null değeri kullanılabilir.HashMap console'a yazdırıldığında hem key değerleri hem de value değerleri aralarına = sembolü konularak yazdırılır. HashMap'ler ekrana yazdırırken rastgele yazdırır. HashMap, Map'ler arasında en hızlısıdır.

##### Normalde Java kendisi Hashing tekniği kullanarak kendisi key atar. Maps'lerde bu keyleri biz kendimiz atıyoruz.

##### HashMap'ler "therad safe" değildir. Bazen bir application'da aynı anda iki app birlikte çalışır. bir application buna müsaitse buna "thread safe" deniyor, yani aynı anda başka şeyler yapabiliyor.

##### 

##### 

##### List<String> isimler ahmet, mehmet nazmi

##### List<Integer>boy 180,200,190 yazacağımıza

##### hashMap'le ahmet 180, mehmet 200, nazmi 190 olarak çalışabiliyoruz.

**package** day37maps;

**import** java.util.HashMap;

**public** **class** Maps01 {

**public** **static** **void** main(String[] args) {

## .put() Methodu

##### Bir HashMap'e eleman eklemek istediğimizde put() methodu kullanırız, burada iki parametre kullanırız key ve Value. key'de sayı yerine null koyabiliriz ama sadece bir kere çünkü keylerde bunu dublication sayar, Exception vermez, eskisinin üzerine yazar.

##### HashMap<Integer,String> hashMap = new HashMap<Integer, String>();

##### hashMap.put(3333, "Ali");

##### hashMap.put(1, "Veli");

##### hashMap.put(55, "Mine");

##### hashMap.put(4, "Kemal");

##### System.out.println(hashMap); // {1=Veli, 4=Kemal, 3333=Ali, 55=Mine}

## .remove() Methodu

##### System.out.println(hashMap.remove(4)); // Kemal

System.***out***.println(hashMap); // {1=Veli, 3333=Ali, 55=Mine}

System.***out***.println(hashMap.remove(1, "Veli")); // true verir.

System.***out***.println(hashMap.remove(55, "Ayse")); // false verir.

System.***out***.println(hashMap); // {3333=Ali, 55=Mine}

## .get() Methodu

##### { } ve key numarası olmadan value okumak için get kullanıyoruz. bu key numarasında value yoksa null yazar.

##### System.out.println(hashMap.get(55)); // Mine

System.***out***.println(hashMap.get(5555)); // null yazar

## .keySet() Methodu

##### Map'teki key numalarını Set halinde görmek için .keySet() methodunu kullanıyoruz.

##### System.out.println(hashMap.keySet()); // [3333, 55]

## .values() Methodu

##### Map'teki vauleleri Set halinde yazdırmak için .values() methodunu kullanıyoruz.

##### System.out.println(hashMap.values()); // [Ali, Mine]

## .size() Methodu

##### Map'te kaç eleman var bunu yazdırmak için .size() methodu kullanılır.

##### System.out.println(hashMap.size()); // 2

## containsKey() Methodu

##### Map içerisinde aradığımız key numarasında value var mı diye bakar, true veya false verir.

##### System.out.println(hashMap.containsKey(3333)); // true

## .containsValue() Methodu

##### Map içerisinde aradığımız value varmı diye arattırmak için .containsValue() methodu kullanılır, True veya false döner.

##### System.out.println(hashMap.containsValue("Mine")); // true

## .replace() Methodu

##### key numarasındaki value yerine yeni value yazar.

##### hashMap.replace(55, "Emin");

System.***out***.println(hashMap);

## .clear() Methodu

##### Tüm elemanları siler. void bir method olduğu için console'a yazar.

##### hashMap.clear(); // {3333=Ali, 55=Emin}

}

}

# HashTable

##### HashMap ile hemen hemen aynıdır.Farkları:

##### 1) Hashtable console'a yazdırıldığında HashMap gibi rastgele sıralamada yazdırılır.

##### 2) Hashtable Key ve Value'larda null'a müsade etmez.

##### 3) Hashtable thread safe'dir.Bazen bir application'da aynı anda iki app birlikte çalışır.bir application buna müsaitse buna "thread safe" deniyor, yani aynı anda başka şeyler yapabiliyor.

##### 4) HashTable daha yavaştır.

##### HashMap'te kullanılan tüm Methodların tamamı Hashtable'da da kullanılabilir.

**package** day37maps;

**import** java.util.Hashtable;

**public** **class** Maps02 {

**public** **static** **void** main(String[] args) {

##### Hashtable<String, String> hTable = new Hashtable<>();

##### hTable.put("Dil","Kalp");

hTable.put("Gönül", "Kalp");

hTable.put("Dil", "Tad alma organı");

##### // hTable.put(null, "Agızlik"); // Run Time Error verir

##### // hTable.put("Kanaat", null); // Run Time Error verir

##### hTable.put("Ali", ""); // null olmadığından problem olmaz.

##### System.out.println(hTable); // {Ali=, Dil=Tad alma organı, Gönül=Kalp}

}

}

# TreeMap

##### TreeMap key'lerde null kullanmaya müsade etmez ama value'larda istediğimiz kadar null kullanmaya müsade eder.

##### Collection'larda ne zaman Tree duyarsanız natural order aklımıza gelmeli. TreeMap elemanlarını console'a yazdırmak istediğimizde key'ler natural order'a göre sıralanır.TreeMap en yavaşlarıdır.

**package** day37maps;

**import** java.util.TreeMap;

**public** **class** Maps03 {

**public** **static** **void** main(String[] args) {

##### TreeMap<String, Integer> tMap = new TreeMap<>();

##### tMap.put("Kanaat", null);

##### tMap.put("Bezelye", 7);

##### tMap.put("Elma", 10);

##### tMap.put("Armut", 8);

##### System.out.println(tMap); // alfabetik sırada yazar

##### // {Armut=8, Bezelye=7, Elma=10, Kanaat=null}

}

}

# Enumerator

**package** day37maps;

**public** **class** Enumerator {

##### iterator Collection'da elemanlar arasında gezer ve Collection'ı update ederdi.

##### Enumeration java.util package içinde bir interface'dir.

##### Enumaration, iterator Farkları:

##### 1) iterator ile hemen hemen aynı işi yapar ama Enumeration'da remove() methodu yoktur. programdan verileri silmek bazen tehlikeli olur, bu tür bir önlem almak istediğimizde Enumeration kullanılması işe yarayabilir. silme yetkisi vermemiş oluyoruz.

##### 2) Iterator'da hasNext(), next(), remove(), hasPrevious(), previous() methodları vardır. Enumaration'da aynı işi yapan farklı isimli methodlar vardır. Mesela; hasMoreElement(), nextElement() gibi

Hashtable<String, String> hTable = **new** Hashtable<>();

hTable.put("Dil","Kalp");

hTable.put("Gönül", "Kalp");

hTable.put("Dil", "Tad alma organı");

// hTable.put(null, "Agızlik"); // Run Time Error verir

// hTable.put("Kanaat", null); // Run Time Error verir

hTable.put("Ali", "");// null olmadığından problem olmaz.

System.***out***.println(hTable); // {Ali=, Dil=Tad alma organı, Gönül=Kalp}

##### Enumeration names = hTable.keys();

##### while (names.hasMoreElements()) {

##### String key = (String) names.nextElement();

##### System.out.println("Key: " + key + " & Value: " + hTable.get(key));

##### // Key: Ali & Value:

##### // Key: Dil & Value: Tad alma organı

##### // Key: Gönül & Value: Kalp

##### }

}

# 38.DERS ( INTERVIEW HAZIRLIK )

# OCA SINAVI ÇALIŞMA NOTLARI

// STRING CLASS

String txt = "We are the so-called \"Vikings\" from the north.";

// \" " işaretini yazdır demektir. \\ \ işaretini yazdır gibi.

System.***out***.println(txt); // We are the so-called "Vikings" from the north.

String n = "Hello\nWorld!";

System.***out***.println(n); // \n "new line" bir sonraki yazılanı alta satıra atar. r de aynı işlemi yapar.

// Hello

// World!

String t = "Hello\tWorld!";

System.***out***.println(t); // \t bir tablık arada boşluk bırakır.

// Hello World!

String b1 = "Hel\blo World!";

System.***out***.println(b1); // \b bakcspace

// Hel\_lo World!

// MATH CLASS

System.***out***.println(Math.*random*()); // 0 ile 1 arası random double sayılar döndürür

// 0.273070524121223

**int** randomNum = (**int**)(Math.*random*() \* 101); // 0 ile 100 arası tam sayılar döndürür.

System.***out***.println(randomNum); // 74

**int** randomNum1 = (**int**)(Math.*random*() \* 100 + 1) ; // 1 ile 100 arası tam sayılar döndürür.

System.***out***.println(randomNum1);

System.***out***.println(Math.*sqrt*(64)); // .sqrt() kök değerini alıyor

// 8.0

System.***out***.println(Math.*abs*(-4.7)); // .abs() absolut mutlak değerini alır.

// 4.7

// parseBooelan() methodu ve new Boolean(String) parametre olarak string kullanır. string true olursa true compile eder, başka hangi string

// değer girilirse hata vermeden false değerini compile eder.

// boolean sadece true ve false değeri alabiliyor.

// Wrapper class olarak Boolean true, false ve null değeri alabiliyor.

// default değerler;

// char c; hiç bir şey

// Character c; null

// boolean a; false

// Boolean a; null Wrapper classların defaultları hep null

// float f; 0.0

// byte e; 0

// int i; 0

// long l; 0

// double d; 0.0

// String s; null

**static** String *b* = "True";

**static** String *c* = "1";

**static** **boolean** *a* = Boolean.*parseBoolean*(*b*);

**static** **boolean** *d* = Boolean.*parseBoolean*(*c*);

**int**[] array = **new** **int**[3];

**public** **static** **void** main(String[] args) {

System.***out***.println(*a*);

System.***out***.println(*d*);

Boolean[] bool = **new** Boolean[3];

bool[0] = **new** Boolean("1");

System.***out***.println(bool[0]);

}

// bu şekilde static ve instance variable yazılımındaki fark; iki farklı obje oluşturulduğunda ve method

// çalıştırıldığında ilkinde count ve i=5 olur. ikinci obje çalıştırıldığında count 5 den devam eder i ise 0'dan.

**static** **int** *count* = 0;

**int** i = 0;

**public** **void** changeCount() {

**while** (i<5) {

i++;

*count*++;

}

}

// indexOf() methodu bulamadığı değerlerde -1 değerini döndürüyor.

// Switch statement ==> boolean, long, double,float ile çalışmıyor

// Switch statement ==> String ve int ile çalışıyor short, byte, char int'ye dönüşebildiği için bu data typleride oluyor.

// javac Test.java --> Test classını çalıştırıyor

// java Test ahmet --> main methoda arguman olarak String ahmet girişi yapıyor.

// b.instanceOf(I) --> b objesi I classının bir objesimidir demek.

// main methodda variable'a değer atamadan tanımlayabiliyoruz ama işlem yapacağımız zaman değer atanmamışsa

// hata veriyor.

// StringBuilder içinde toString methodu var bu nedenle direk print edilebiliyor.

// main methodda sadece args ismi değiştirilebiliyor, farklı yazılımlar main method olmuyor.

// arrayin null değerine bir method (concat gibi) uygulamaya kalkınca NullPointerExeption veriyor.

// Boolean.valueOf("TRUE") true haricinde stringe ne yazılırsa false döndürür.

// static methodlar static import kullanılarak import edilebiliyor.

// Java default olarak boş constructor üretiyor, superclass ta boş constructor olmayınca hata verir.

// obje oluşturulduğunda default olarak parametresiz bir super() mevcuttur. ilgili classta parametresiz constructor

// varsa çalıştırır, yoksa hata vermez.

// arrayler toString methodu kullanılmadan direk array ismi ile yazdırılırsa referans adresini yazdırır.

// static final variable varsa direk veya static initiliasor ile değer atanmamaşsa hata verir.

// objeleri eşitleyince bir atama olunca iki obje gibi gözükse de tek obje gibi düşüneceğiz.

// Objeye null atanınca obje oluşturulmuş sayılmıyor.

// exeption öngürülebilir(checked) olduğu için declare edilmesi şart.

// bir method Exeption atıyorsa ;

// 1. hem methodda hem de main methodda bu method bir Exeption atabilir manasında throws exeption ifadesi

// method parametresinden sonra belirtilmek zorunda.

// public void good() throws Exception { } public static void main(String[] args) Exception { }

// 2. yada main methodda method try catch block içinde çağrılıp declare edilmek zorunda

// yoksa compile etmez.

// catch sırası özelden genele olmalı, sıra yanlış yazılırsa compile error verir.

// continue nun altina bisey yazamayiz yazarsak compile error verir

// instance variable switch dışında mutlaka tanımlanmalı, tanımlanan variable switch içerisinde return type ile

// birlikte yazılmamalı yoksa hata verir.

// LAMBDA DOĞRU YAZIM

// if(arrList.removeIf( s-> { //s degeri

// System.out.print(s); // s array listing tamamini yazdirir

// return s.length()<=2; //sonra da lengthi 2 den kucuk olani remove yapar. return yazılsada olur yazılmasada

// }))

// if (arrList.removeIf((String s) ->{ return s.length()<=2;})) // return yazılmazsa hata verir.

// {

// System.out.println("removed");

// }else{

// System.out.println("not removed");

// }

# OCA JAVA 8 SE KİTABI ÇALIŞMA NOTLARI

## CHAPTER 01

**public** **class** OcaChapter01Notlari {

// javac isim.java Komutunu bilgisayara gönderdiğinizde programınızda hata yoksa ve bilgisayarınız

// java komutunun yerini biliyorsa (path deyimi ) java derleyicisi isim.class dosyasını oluşturur.

// java isim komutu kullanılarak sonuçlar ekranda veya açılan bir grafik penceresinde görülebilir.

// Program 1.1 Hoşgeldiniz.java konsol programı

// class Hosgeldiniz

// {

// public static void main(String args[])

// {

// System.out.println("Java Sinifina Hos Geldiniz!");

// }

// }

// bu programı çalıştırmak için;

// javac Hosgeldiniz.java

// java Hosgeldiniz

// komutlarını konsolda yazmamız yeterlidir. konsol çıktısı olarak: Java Sinifina Hos Geldiniz!

// bir dosyada iki sınıf olabir ama bunlardan sadece bir tanesi public olabilir.

// java.lang.\* paketini import etmeye gerek yok default olarak zaten bu işi yapıyor. bunu yazmak gereksiz olur.

// import java.nio.file.\*; --> Doğru

// import java.nio.file.Files; --> Doğru

// import java.nio.file.Paths; --> Doğru

// import java.nio.\*; --> yanlış

// import java.nio.\*.\*; --> yanlış

// import java.nio.files.Paths.\*; --> yanlış

// import java.util.Date; import java.sql.Date; veya import java.util.\*; import java.sql.\*;

// ikisi aynı anda aynı sınıf isimli paketler yazılırsa conflict olur çalışmaz.

// eğer ikisinide kullanmak istersek birini import yapıyoruz, diğerini aşağıdaki gibi yazıyoruz

// java.util.Date date;

// java.sql.Date sqlDate; veya import java.util.Date; import java.sql.\*; şeklinde de yazılabilir.

// ArrayList java.util package içerisindedir bunu bilmemiz gerekiyor.

// Constructor ismi class ismi ile aynı olacak, return tipi olmayacak

// { } kaç tane bu şekil parantez varsa o kadar kod bloğu var demektir.

// sabit değerler büyük harfle yazılıyor. int PI = 3.14;

// bir variable $ ve \_ işaretleri ile başlıyabilir.

// bir variable harf, sayı, $ ve \_ işareti içerebilir.

// local variable'ların default değeri yoktur, tanımlandığında değerde atanması gerekir.

// finalize() (method) ==> finalize() methodu, Garbage Collector'ın imha edeceği dataları imha edilecek

// hale getirir. çöpleri topla sonra atarım. Garbage collectore'a bak!!

// finalize() methodu ihtiyaç yoksa çalışmaz veya en fazla bir kez çalışır.

// A. int i1 = 1\_234; DOĞRU

// B. double d1 = 1\_234\_.0; YANLIŞ

// C. double d2 = 1\_234.\_0; YANLIŞ

// D. double d3 = 1\_234.0\_; YANLIŞ

// E. double d4 = 1\_234.0; DOĞRU

**public** **static** **void** main(String[] args) {

**int** i2 = 1\_234;

System.***out***.println(i2);

}

// Bir değişkenin tipi belli değilse o değişken void olur.

//

// int carpim(int a , int b)

//

// { return a\*b;}

// Yukarıdaki şekilde kodda bir dönüş değeri vardır. Her hangi bir yerden metod çağırıldığı zaman çarpım

// dönüş değeri ekrana basılır.

//

// void carpim(int a, int b)

//

// {System.out.println(a\*b);}

// GARBAGE COLLECTİON

// Ogrenci ogr1 = new Ogrenci();

// Ogrenci ogr2 = new Ogrenci();

// ogr1 = ogr2;

//

// Bu örneğimizde Ogrenci sınıfından iki adet nesne oluşturduk. 1. ve 2. satırlarda ilk oluşturduğumuz

// nesnenin adresini “ogr1” işaretçisine, ikinci oluşturduğumuz nesnenin adresini “ogr2“ işaretçisine aktardık.

// 3. satırda ise ikinci nesnenin adresini, ilk nesnenin adresini tutan “ogr1” işaretçisine aktardık.

// Yani artık ogr1 işaretçisi, ilk nesnenin adresini değil ikinci nesnenin adresini tutmaktadır.

// İlk oluşturulan nesnenin adresi kaybedildiği için artık o nesneye ulaşmak mümkün olmayacaktır.

// Nesneye erişimin mümkün olmadığı durumlarda nesnenin bellekte kalmasının bir anlamı yoktur.

// Bu yüzden Garbage Collection tarafından silinecektir.

// Initialization = başlatma demek bir method çalıştırılınca öncelik olarak { } işareti içine yazılanları çalıştırır,

// sıralama önce super(), sonra statik blok, sonra { }, en son method . Access modifier private da olsa,

// final da olsa çalıştırır.

// 1: public class InitializationOrderSimple {

// 2: private String name = "Torchie";

// 3: { System.out.println(name); }

// 4: private static int COUNT = 0;

// 5: static { System.out.println(COUNT); }

// 6: static { COUNT += 10; System.out.println(COUNT); }

// 7: public InitializationOrderSimple() {

// 8: System.out.println("constructor");

// 9: } }

//

// 1: public class CallInitializationOrderSimple {

// 2: public static void main(String[] args) {

// 3: InitializationOrderSimple init = new InitializationOrderSimple();

// 4: } }

// The output is:

// 0

// 10

// Torchie

// constructor

## CHAPTER 02

**public** **class** OcaChapter02Notlari {

// iki farklı veri türü arasında bir aritmatik işlem varsa otomatik olarak küçük olan büyük

// olan veri türüne çevirilir.

// int ile float arasında işlem yapılırsa int değeri float değerine yükseltilir.

// byte,short,char veri tiplerinde aritmatik işlem yapılıyorsa sayı otomatik olarak int veri tipine çevirilir.

// sonuçlar yükseltilen veri tipi cinsinden olur.

// double x = 39.1 float y = 2.1 x + y = not compile!

**public** **static** **void** main(String[] args) {

// ÖRNEK 1:

**double** x = 39.1;

**float** y = 2.1f;

System.***out***.println(x + y); // 41.19999990463257

**short** a = 3;

**short** b = 4;

**int** c = a + b;

// short d = a + b; java shortu otomatik olarak aritmatik işlemlerde intye çevirir

// öyle olunca iki int değerinin toplamını short olarak atayamazsın diye hata verir

**short** d = (**short**) (a + b); // şeklinde overflow yaparak işlem short tipinde yaptırılabilir.

System.***out***.println(c);

// java bütün ondalık kesirleri otomatik olarak double kabul eder. rakamın soluna f konulmazsa

// virgüllü sayı olduğu için bunu double yap diye uyarı verir. float değer virgüllü değilse ve

// yanına f konulmamışsa hata vermeden float kabul ederek çalışır.

// { } ve key numarası olmadan value okumak için get kullanıyoruz. bu key numarasında value yoksa null yazar.

// x.getValue() < 5 x'e bir değer atanmamışsa null < 5 olacak oda NullPointerExepception verir.

// & ve && ifadelerinin farkı bir tane yazılırsa kıyaslamanın her iki tarafınında değerlendirir, iki tane

// yazılırsa ilk ifadeye bakar ona göre değerlendirme yapar.

// ÖRNEK 2:

**int** x1 = 6;

**boolean** y1 = (x1 >= 6) || (++x <= 7); //ilk ifadenin true olunca ikinci ifadeyi değerlendirmeden atlar true yapar.

System.***out***.println(x1);

System.***out***.println(y1);

System.***out***.println();

// ÖRNEK 3:

**int** k = 2;

**int** l = 3;

**int** m = 4;

l \*= ++k;

System.***out***.println("l = " + l); // 9

k += ++m;

System.***out***.println("k = " + k); // 8

System.***out***.println();

// ÖRNEK 4:

**int** y11 = 1;

**do** {

// int y11 = 1; variable body içinde oluşturulursa while parantezi içinde bu görülmez ve hata verir.

// her zaman do veya while döngüsünden önce variable'ın tanımlanması gerekiyor.

System.***out***.print(y11++ + " ");

} **while**(y11 <= 10);

**final** **char** a1 = 'A', d1 = 'D';

**char** grade = 'B';

**switch**(grade) {

**case** a1:

**case** 'B': System.***out***.print("great");

**case** 'C': System.***out***.print("good"); **break**;

**case** d1:

**case** 'F': System.***out***.print("not good");

}}} // greatgood

// case e bir değer yazmamız yeterli int, String farketmez switch parametresiyle karşılaştırma yapacak.

// break yazmazsak karşılaştırmadaki değerler uyuşmasa bile o değeri yazdırır.

// a1 de bir değer görev olmadığı için bir şey return etmez. 'C' eşleştiği için return eder ve break yapar.

// FİNAL

//

// 1. bir variable final yapılırsa değeri değiştirilemez. Değer atanmamış bir final variable'a(blank final variable)

// ancak bir Constructor vasıtasıyla değer atayabilirim.

// class Bike10{

// final int speedlimit;//blank final variable

//

// Bike10(){

// speedlimit=70;

// System.out.println(speedlimit);

// }

//

// public static void main(String args[]){

// new Bike10();

// }

// }

// 2. değer atanmamış boş static final variable'a sadece bir static blok ile değer atayabilirim.

// class A{

// static final int data;//static blank final variable

// static{ data=50;}

// public static void main(String args[]){

// System.out.println(A.data);

// }

// }

// 3. bir method final yapılırsa inherit edilebilir ama override edilemez.

// class Bike{

// final void run(){System.out.println("running");}

// }

//

// class Honda extends Bike{

// void run(){System.out.println("running safely with 100kmph");} // compile Error

// }

// 4. bir class final yapılırsa extend yapılamaz.

// final class Bike{}

//

// class Honda1 extends Bike{ // compile Error

// void run(){System.out.println("running safely with 100kmph");}

//}

// 5. Herhangi bir parametre final olarak bildirilirse değeri değiştirilemez hata verir.

// class Bike11{

// int cube(final int n){

// n=n+2;//can't be changed as n is final

// n\*n\*n;

// }

// 6. bir Constructor final olamaz.

## CHAPTER 03

**public** **class** OcaChapter03Notlari {

/\* sıralama yapılmadan arraylarda çıktı alınmak istenirse sonuç tahmin edilemez, yanlış çıkar.

\* istisna olarak verilen array zaten sıralı ise sortieren yapmaya gerek olmadan sonucu veriyor.

\* split bir string methodudur. str.split( ); methodunu seçip parametre içine hangi karakteri yazarsak

boşluklara göre Stringi parçalayıp kaç parça olduğunu bize yazar.

Girilen karakteri çıkarır aradaki kelimeleri sayar. " " koyduğumuzda kelimeler ortaya çıkar.

Split methodu kullandığınızda mutlaka bir Array oluşturmanız gereklidir.

\* binarySearch() methodu aranan eleman var olduğunda o elemanın indexini return eder.

\* Arrays.sort() methoduna parametre olarak array'in ismi yazılırsa o Array'in elemanlarını küçükten

büyüğe sıralanmış olur.

\* Arrays.toString() methoduna parametre olarak array'in ismini yazarsanız o array'in tüm elemanlarını

ekranda görürsünüz.

\* Array'in length'ini bulmak için "ArrayIsmi.length" yazmak yeterlidir.

\* System.out.println(Arrays.deepToString(arr)); // .deepToString sonuna kadar git tüm elemanları yazdır demektir.

\* System.out.println(arr[1].length);// ikinci array grubu içerisindeki eleman sayısını verir.

\*

\* List<String> list = new ArrayList<>(); LISTLERİN GENEL KULLANIM ŞEKLİ

\* ArrayList<String> list7 = new List<>(); // DOES NOT COMPILE

\* Collections.sort(list); LİSTLERİ SIRALAMA

\* Object[] arr1=list.toArray(); LISTI ARRAY'A ÇEVİRME

\* Integer array[] = list.toArray(new Integer[4]); BAŞKA BİR ÖRNEK

\* asList( ) methodu ARRAY'İ LİSTE ÇEVİRME

asList( ) methodu parametre olarak array'in ismini kabul eder.

List<String> list1 = Arrays.asList(arr2); Array'den liste çevirme yaptığınızda, elde ettiğiniz list

uzunluk olarak esnek değildir. Yani array'den oluşturduğunuz liste ekleme ve çıkarma yapamazsınız.

\* Array'den liste çevirme yaptığınızda, elde ettiğiniz list uzunluk olarak esnek değildir oluşturduğunuz

\* liste ekleme ve çıkarma yapamazsınız. yaparsanız aşağıdaki gibi

list.remove(1); // throws UnsupportedOperation Exception alırsınız. ACHTUNG!!

\* listaddAll() methodu İKİ LİSTİ BİRLEŞTİRME

\* list1.addAll(list2);

\* list4.addAll(1, list3);; // [2, 1, 3]

\* 3: List<Integer> heights = new ArrayList<>(); 4. SATIRDA DEĞİL 5. SATIRDA HATA VERİR DİKKAT!!

4: heights.add(null); // referans olarak null atıyor, sorun yok.

5: int h = heights.get(0); // NullPointerException // değer olarak atadığı zaman hata veriyor.

\* VARARGS: VARIABLE ARGUMENTS

\* Bir method oluşturduğumuzda argüment olarak istediğimiz kadar argüment girebilmemizi sağlar.

\* int den sonra ... koyunca java bunun Varargs olduğunu anlar.sonu açık olan bir Array gibi düşünebiliriz.

\* public static void countNum(int... a) { // Vararg'ın ismi a

\* VARARGS'IN YANLIŞ KULLANIMLARI

Kural 1: Varargs her zaman son parametre olmalıdır.

Kural 2: parametre olarak 1'den fazla varargs kullanamazsınız çünkü varargs en sonda olmalıdır,

public static void valueChar(String s, char... c) // doğru

public static void product(int... x, int y) { // yanlış

public static void product1(int... x, int... y){// yanlış

\* FOR EACH

\* // {{1, 2}, {5}, {6, 7, 8}} arrayindeki tüm elemanların toplamını bulunuz.

int arr2[ ][ ]={{1, 2}, {5}, {6, 7, 8}};

int sum1=0;

for(int[ ] w:arr2) {

for(int z: w) {

sum1=sum1+z;

}

}

\*/

**public** **static** **void** main(String[] args) {

System.***out***.println("abcabc".replace("a", "A")); // AbcAbc

System.***out***.println("abcabc".replace('a', 'A')); // AbcAbc

String result = "AniMaL ".trim().toLowerCase().replace('a', 'A');

System.***out***.println(result); // AnimAl

// String'le int, char ve boolean'ı + ile concatination yapabiliyorum.

String a1 = "";

a1 += 2;

a1 += 'c';

a1 += **false**; // BU ŞEKİLDE BİRLEŞTİRİYOR SORUN YAPMIYOR

**if** ( a1 == "2cfalse") System.***out***.println("==");

**if** ( a1.equals("2cfalse")) System.***out***.println("equals");

// StringBuilder b = "rumble"; YANLIŞ StringBuilder'a String atayamıyorum. The code does not compile.

StringBuilder b = **new** StringBuilder("rumble"); // DOĞRUSU

// ARRAYLERDE EN AZ BİR TANE ARRAY UZUNLUĞUNUN TANIMLANMASI, ÇOKBOYUTLU ARRAYLERDE İSE İLK BOYUTUN

// TANIMLANMASI GEREKİYOR YOKSA

**int**[][] scores = **new** **int**[5][]; // ÇALIŞIR

// int[][] java = new int[][]; // COMPİLE ERROR

// Two arrays with the same content are equal. // FALSE

// Two ArrayLists with the same content are equal. // TRUE

// ARRAYLERDE equal() methodu yok ArrayListlerde var. ACHTUNG!!!

// eğer aranan sayı Array içinde yoksa, olsaydı hangi indexte olacaksa ona +1 ekleyip sonucu - ile çarpıyoruz.

**int**[] numbers = {2,4,6,8};

System.***out***.println(Arrays.*binarySearch*(numbers, 2)); // 0

System.***out***.println(Arrays.*binarySearch*(numbers, 4)); // 1

System.***out***.println(Arrays.*binarySearch*(numbers, 1)); // -1

System.***out***.println(Arrays.*binarySearch*(numbers, 3)); // -2

System.***out***.println(Arrays.*binarySearch*(numbers, 9)); // -5

// String'i StringBuilder yapma

String a = "abc";

StringBuilder b1 = **new** StringBuilder(a);

System.***out***.println(b1.toString());

String s1 = "java";

StringBuilder s2 = **new** StringBuilder("java");

// if (s1.equals(s2)) // String ile StringBuilder valuelerini karşılaştırırsak does not Compile alırız.

System.***out***.print("2");

// total += letters.substring(6, 5).length(); BU ŞEKİLDE BİR YAZIMDA An exception is thrown VERİR.

// MULTİDİMENSİONAL ARRAY YAZDIRMA

**int** arr[][]=**new** **int**[3][4];

System.***out***.println(Arrays.*toString*(arr)); // [[I@1540e19d, [I@677327b6, [I@14ae5a5]

System.***out***.println(Arrays.*deepToString*(arr)); // .deepToString sonuna kadar git tüm elemanları yazdır demektir. 225

// [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]] yazdırır.

// REMOVE METHODU

List<Integer> numbers1 = **new** ArrayList<>();

numbers1.add(1);

numbers1.add(2);

numbers1.add(3);

numbers1.remove(1); // 1. indexi yani 2 sayısını siler.

numbers1.remove(**new** Integer(3)); // 3 rakamını bulup siler.

System.***out***.println(numbers1); // Listte toString metodu kullanmadan direk listeyi yazıyor. [1]

// LOCALDATETIME

LocalDate date = LocalDate.*now*();

System.***out***.println(date);

LocalTime time = LocalTime.*now*();

System.***out***.println(time);

LocalDate date1 = LocalDate.*of*(2020, Month.***JANUARY***, 20);

LocalTime time1 = LocalTime.*of*(5, 15);

LocalDateTime dateTime = LocalDateTime.*of*(date1, time1)

.minusDays(1).minusHours(10).minusSeconds(30);

System.***out***.println(dateTime); // 2020-01-18T19:14:30

// PERIOD

LocalDate start = LocalDate.*of*(2015, Month.***JANUARY***, 1);

LocalDate end = LocalDate.*of*(2015, Month.***MARCH***, 30);

Period period = Period.*ofMonths*(1); // create a period

*performAnimalEnrichment*(start, end, period);

// give new toy: 2015-01-01

// give new toy: 2015-02-01

// give new toy: 2015-03-01

Period annually = Period.*ofYears*(1); // every 1 year

Period quarterly = Period.*ofMonths*(3); // every 3 monthsWorking with Dates and Times

Period everyThreeWeeks = Period.*ofWeeks*(3); // every 3 weeks

Period everyOtherDay = Period.*ofDays*(2); // every 2 days

Period everyYearAndAWeek = Period.*of*(1, 0, 7); // every year and 7 days

Period p = Period.*ofDays*(1).*ofYears*(2); // ARD ARDA METHOD YAZMASINA İZİN VERİR(COMPİLE), KOD ÇALIŞIR AMA

// SADECE SON METHOD YANİ İKİ YIL ÇALIŞIR GÜNÜ DİKKATE ALMAZ. !!!

System.***out***.println(date.plus(period)); // 2015-02-20

System.***out***.println(dateTime.plus(period)); // 2015-02-20T06:15

// System.out.println(time.plus(period)); // UnsupportedTemporalTypeException TİME'A AY EKLEYEMEYİZ

// HATA VERİR.

System.***out***.println(date.getDayOfWeek()); // MONDAY

System.***out***.println(date.getMonth()); // JANUARY

System.***out***.println(date.getYear()); // 2020

System.***out***.println(date.getDayOfYear()); // 20

// DATE TİME FORMAT

System.***out***.println(time.format(DateTimeFormatter.***ISO\_LOCAL\_TIME***));

System.***out***.println(dateTime.format(DateTimeFormatter.***ISO\_LOCAL\_DATE\_TIME***));

DateTimeFormatter shortDateTime = DateTimeFormatter.*ofLocalizedDate*(FormatStyle.***SHORT***);

System.***out***.println(shortDateTime.format(dateTime)); // 1/20/20

System.***out***.println(shortDateTime.format(date)); // 1/20/20

// System.out.println(shortDateTime.format(time)); // UnsupportedTemporalTypeException

// üstte obje .ofLocalizedDate olarak tanımlanmış altta time istenmiş hata verir, .ofLocalizedTime

// olsayı sorun olmazdı. birde .ofLocalizedDateTime var. 3 ayrı format mevcut.

// LocalDate date11 = LocalDate.of(2018, Month.APRIL, 40); // A runtime exception is thrown.

DateTimeFormatter shortF = DateTimeFormatter.*ofLocalizedDateTime*(FormatStyle.***SHORT***);

DateTimeFormatter mediumF = DateTimeFormatter.*ofLocalizedDateTime*(FormatStyle.***MEDIUM***);

System.***out***.println(shortF.format(dateTime)); // 1/20/20 11:12 AM AY/GÜN/YIL

System.***out***.println(mediumF.format(dateTime)); // Jan 20, 2020 11:12:34 AM AY/GÜN/YIL

DateTimeFormatter f = DateTimeFormatter.*ofPattern*("MMMM dd, yyyy, hh:mm");

System.***out***.println(dateTime.format(f)); // January 20, 2020, 11:12

// What is the output of the following code?

LocalDate date11 = LocalDate.*of*(2018, Month.***APRIL***, 30);

date11.plusDays(2);

date11.plusYears(3);

System.***out***.println(date11.getYear() + " " + date11.getMonth() + " "

+ date11.getDayOfMonth()); // 2018 APRIL 30 dates are immutable WİCHTİG!!

// PARSE() METHODU

// create an LocalDate object

LocalDate lt = LocalDate.*parse*("2018-12-27");

// print result

System.***out***.println("LocalDate : " + lt); // LocalDate : 2018-12-27

// DİĞER BİR KULLANIM ŞEKLİ

// create a formater

DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("dd MMM uuuu");

// create a LocalDate object and

LocalDate lt1 = LocalDate.*parse*("31 Dec 2018", formatter);

// print result

System.***out***.println("LocalDate : " + lt1.toString());

}

**private** **static** **void** performAnimalEnrichment(LocalDate start, LocalDate end,

Period period) { // uses the generic period

LocalDate upTo = start;

**while** (upTo.isBefore(end)) {

System.***out***.println("give new toy: " + upTo);

upTo = upTo.plus(period); // adds the period

}}

}

## CHAPTER 04

**public** **class** OcaChapter04 {

// Kural olarak main methodda aynı sınıfın bir constructoru oluşturulursa hemen çalıştırılır.

// constructor çalıştırılınca ilk olarak constructor dışındaki instance initilazier çağrılır sonra

// constructor çağrılır.

// variable private olursa ya setter yada initilazation { } kullanarak değer atanabilir, diğer türlü

// yapılmış atamalar varsa compile Error verir.

// import static java.util.Collections; // method static olması durumunda bu import yeterlidir.

// public void static method(ArrayList<String> list) {

// sort(list);

// import static java.util.Collections.\*; // method static olmaması durumunda bu import kullanılır.

// public void method(ArrayList<String> list) {

// sort(list);

//aşağıdakilerden hangileri default Constructor kullanır ve compile eder?

//A. public class Bird { } // doğru

//

//B. public class Bird { public bird() {} } // class ismi Bird ile aynı değil Constructor olmadı.

// metod ismi bird olabilirdi ama return type yok metodda olmuyor. does not compile

//

//C. public class Bird { public bird(String name) {} } // class ismi Bird ile aynı değil Constructor olmadı.

// return type yok metodda olmuyor. does not compile

//D. public class Bird { public Bird() {} } // Constructor oluşturdu default iptal

//

//E. public class Bird { Bird(String name) {} } // Constructor oluşturdu default iptal

//

//F. public class Bird { private Bird(int age) {} } // Constructor oluşturdu default iptal

//

//G. public class Bird { void Bird() { } } // return type olunca method oluşturdu. Constructor default kullanır DOĞRU

// Constructorlar new kullanılmadan compile edilemezler. new BirdSeed(); bu isimdeki constructorın

// süslü parantezini çalıştır demektir.

// static methodlar override edilemezler çünkü static method sınıfa aittir, instance method bir objeye aittir.

// bir methodda iki tane return type olmaz. hem int hem void, hem void hem String gibi..

// kural olarak final, static ve access modifierler return type den önce yazılmaları gerekiyor.

// return type herhangi bir değer döndürmezse void ile birlikte yazılabilir aksi halde does not compile

// public void methodA() { return;} --> DOĞRU

// public void methodB() { return null;} --> YANLIŞ

// varargs parametre içinde mutlaka en sonda ve bir tane olması gerekiyor.

// public void moreB(String values, int... nums) {}

// main methodda obje oluşturulunca sırasıyla ilk olarak sadece bir kez olmak üzere static initiliziar çağrılır,

// sonra instance initiliziar, daha sonra da constructor çağrılır. aşağıda iki obje oluşturulmuş, bir kez

// static initiliziar, iki kez instance initiliziar, iki kez de constructor çağrılmış oldu.

**int** speed;

OcaChapter04(){System.***out***.println("constructor çağrılır");}

{System.***out***.println("instance initiliziar bloğu çağırılır");}

**static** {

System.***out***.println("static blok obje tarafından yalnız bir kere çağrılır");}

**public** **static** **void** main(String args[]){

OcaChapter04 b1=**new** OcaChapter04();

OcaChapter04 b2=**new** OcaChapter04();

// ÇIKTI:

// static blok obje tarafından yalnız bir kere çağrılır

// instance initiliziar bloğu çağırılır

// constructor çağrılır

// instance initiliziar bloğu çağırılır

// constructor çağrılır

// STATİK BLOK

// aşağıdaki şekilde statik blok yazıldığında main method çalıştırıldığında ilk olarak bu blok çalıştırılır

// sonra main method içerisindeki kod çalıştırılır.

//

// statik {System.out.println ( "statik blok çağrılır" );}

// public static void main (String args []) {

// System.out.println ( "Merhaba" );

//

// statik blok çağrılır

// Merhaba

// static final variable lara değer ya ilk yazıldığında atanmalı, yada static initilazation içinde atanmalıdır.

// her iki yerde değere ataması olursa veya hiç değer atanmazsa compile error verir.

// instance variable'lara sadece instance initializer'lar ulaşabilir, static initializer ulaşamazlar.

// static variable'lara hem instance hem de static initializer'lar ulaşıp değişiklik yapabilirler.

// final variable'lara hiç biri ulaşamaz

//

// public class Order3 {

// final String value1 = "1";

// static String value2 = "2";

// String value3 = "3";

// {

// // CODE SNIPPET 1

// }

// static {

// // CODE SNIPPET 2

// }

// }

//

// A. value1 = "d"; instead of // kod parçacığı 1 // does not compile çünkü final

//

// B. value2 = "e"; instead of // kod parçacığı 1 // compile

//

// C. value3 = "f"; instead of // kod parçacığı 1 // compile

//

// D. value1 = "g"; instead of // kod parçacığı 2 // does not compile

//

// E. value2 = "h"; instead of // kod parçacığı 2 // compile

//

// F. value3 = "i"; instead of // kod parçacığı 2 // does not compile

// Statik blok aracılığıyla sadece bir kez olmak üzere static final variable'lara değer atanabiliyor.

// 14: private static int one;

// 15: private static final int two;

// 16: private static final int three = 3;

// 17: private static final int four; // DOES NOT COMPILE

// 18: static {

// 19: one = 1;

// 20: two = 2;

// 21: three = 3; // DOES NOT COMPILE

// 22: two = 4; // DOES NOT COMPILE

StringBuilder s2 = **new** StringBuilder("s2");

s2.append("b"); // atama yapmaya ihtiyaç duymadan variable s2 = s2b olur direk.

}

// static variable ile instance variable arasında işlem yaparsak does not compile verir

**public** **int** total;

**public** **static** **int** *count*;

// public static average = total / count; // DOES NOT COMPILE

// letters = letters(letters); // String data typelerinde bu şekilde atama yapmadıkça variable değeri değişmiyor.

// 1: public class ReturningValues {

// 2: public static void main(String[] args) {

// 3: int number = 1; // 1

// 4: String letters = "abc"; // abc

// 5: number(number); // 1

// 6: letters = letters(letters); // abcd

// 7: System.out.println(number + letters); // 1abcd

// 8: }

// 9: public static int number(int number) {

// 10: number++;

// 11: return number;

// 12: }

// 13: public static String letters(String letters) {

// 14: letters += "d";

// 15: return letters;

// 16: }

// 17: }

**public** **static** **int** *length* = 0; // bu şekilde static olarak tanımlanmış bir değişkene, oluşturulan bir obje

// tarafından değer ataması yapılırsa variable değeri kalıcı olarak değişir.

// LAMBDA doğru syntax yazılımı: return type parantez içinde olmalı (String s)

// süslü parantez yazılırsa parentez içinde return yazılmalı ve sonunda ; olmalı.

// removeIf methodu şartı sağlıyorsa bunları kaldır

// List<String> list = new ArrayList<>();

// list.removeIf(s -> s.isEmpty());

// list.removeIf(s -> {return s.isEmpty();});

// list.removeIf((String s) -> s.isEmpty());

// print(() -> true); // 0 parameters

// print(a -> a.startsWith("test")); // 1 parameter

// print((String a) -> a.startsWith("test")); // 1 parameter

// print((a, b) -> a.startsWith("test")); // 2 parameters

// print((String a, String b) -> a.startsWith("test")); // 2 parameters

// LAMBDA aynı değişkenin tekrarlı tanımlanması halinde hata verir.

// (a, b) -> { int a = 0; return 5;} // DOES NOT COMPILE a tekrar tanımlandığı için hata verdi

// (a, b) -> { int c = 0; return 5;} // COMPILE

// Predicate sınavda lambda için bilmemiz gerekiyor. boolean döndüren bir interface

// BU KONU İÇİN ALTTAKİ İKİ ÖRNEĞİ BİLMEMİZ YETERLİ

// 1: import java.util.\*;

// 2: import java.util.function.\*;

// 3: public class PredicateSearch {

// 4: public static void main(String[] args) {

// 5: List<Animal> animals = new ArrayList<Animal>();

// 6: animals.add(new Animal("fish", false, true));

// 7:

// 8: print(animals, a -> a.canHop());

// 9: }

// 10: private static void print(List<Animal> animals, Predicate<Animal>↵checker) {

// 11: for (Animal animal : animals) {

// 12: if (checker.test(animal))

// 13: System.out.print(animal + " ");

// 14: }

// 15: System.out.println();

// 16: }

// 17: }

// removeIf methodu şartı sağlıyorsa bunları kaldır

// 3: List<String> bunnies = new ArrayList<>();

// 4: bunnies.add("long ear");

// 5: bunnies.add("floppy");

// 6: bunnies.add("hoppy");

// 7: System.out.println(bunnies); // [long ear, floppy, hoppy]

// 8: bunnies.removeIf(s -> s.charAt(0) != 'h');

// 9: System.out.println(bunnies); // [hoppy]

}

## CHAPTER 05

**class** OcaChapter05Notlari {

String msg;

OcaChapter05Notlari(String msg){

**this**.msg = msg;

}

// subclass(child), parent class ta bir Constructor oluşturulmuşsa bunu mutlaka inheritance yapmak zorunda.

// Constructor oluşturulmamış veya sadece method oluşturulmuşsa böyle bir zorunluluk yok.

// oluşturulmuş bir constructor varsa subclass ta bir constructor oluşturup aynı parametrede bir super()

// ile bunu çağırmak zorundayız yoksa compile olmaz.

// Her Classta default olarak parametresiz bir constructor vardır ve bu Constructor'ın ilk ifadesi default

// olarak super() dir.

// public final class Mammal extends Animal final yazarak kalıtımın genişletilmesini durdurabiliriz.

// final ifadesinden sonra public class Cat extends Mammal yazarsak does not compile olur.

// parent child ilişkisi içerisindeki classların access modifieri sadece bir tanesinin ki public olabilir.

// bir java dosyasında en fazla bir public class veya interface olabilir.

// public class Parent {

// private String age;

// public Parent(String age) {

// }}

//

// public class Child extends Parent{

// Child(){

// super("Ali");

// }}

// bir method kendi kendini çağırırsa sonsuz döngü oluşur buna "Recursion in Java" deniyor. Özyineleme manasına

// static void p(){

// System.out.println("hello");

// p();

// }

//

// public static void main(String[] args) {

// p();

// }

// methodun kendi kendisini çağırarak döngü oluşturmasını sağlayıp faktoriyel hesaplama

// static int factorial(int n){

// if (n == 1)

// return 1;

// else

// return(n \* factorial(n-1));

// }

//

// public static void main(String[] args) {

// System.out.println("Factorial of 5 is: "+factorial(5));

// }

// parent child ilişkisinde return type'ler covariant olmalı. Yani parent child arasında return typeler yönüyle

// parentın return type'i child'in return type'ini kapsamalı.

// public class Rodent {

// protected static Integer chew() throws Exception {

// System.out.println("Rodent is chewing");

// return 1;

// }

// }

// public class Beaver extends Rodent {

// public Number chew() throws RuntimeException { // NOT COMPİLE üst method static alt method instance

// // NOT COMPİLE return typler covariant değil Number Integer'ın bir subclass'ı değil.

//

// System.out.println("Beaver is chewing on wood");

// return 2;

// }

// }

// method overriding'de kural olarak exeption yazıldığında childdaki exeption daha yeni veya daha kapsamlı olamaz

// exeptionlarda da aynı şekilde parent child ilişkisi olmalı.

// I. Örnek

// public class InsufficientDataException extends Exception {}

//

// public class Reptile {

// protected boolean hasLegs() throws InsufficientDataException {

// throw new InsufficientDataException();

// }

//

// protected double getWeight() throws Exception {

// return 2;

// }

// }

//

//

// public class Snake extends Reptile {

// protected boolean hasLegs() {

// return false;

// }

//

// protected double getWeight() throws InsufficientDataException{

// return 2;

// }

// }

// II.Örnek

// public class InsufficientDataException extends Exception {}

//

// public class Reptile {

// protected double getHeight() throws InsufficientDataException {

// return 2;

// }

//

// protected int getLength() {

// return 10;

// }

// }

// public class Snake extends Reptile {

// protected double getHeight() throws Exception { // DOES NOT COMPILE

// return 2; }

//

// protected int getLength() throws InsufficientDataException { // DOES NOT COMPILE

// return 10; }

// }

// }

// KURAL : Parent ve Child deki methodların ikisininde static olması veya olmaması durumunda düzgün çalışır.

// Aksi halde compile etmez. Static olma durumunda buna method hiding deniyor. iki ayrı sınıfta iki ayrı method

// gibi düşünülebilir. Aşağıdaki iki örnekten anlaşılabilir. methodun static olması demek methodun o classa

// özel has olması demek.

// 1.Örnek: Methodların Static olması durumunda hiding method oluyor.

// public class Marsupial {

// public static boolean isBiped() {

// return false;

// }

//

// public void getMarsupialDescription() {

// System.out.println("Marsupial walks on two legs: "+isBiped());

// }

// }

//

//

// public class Kangaroo extends Marsupial {

// public static boolean isBiped() {

// return true;

// }

//

// public void getKangarooDescription() {

// System.out.println("Kangaroo hops on two legs: "+isBiped());

// }

//

// public static void main(String[] args) {

// Kangaroo joey = new Kangaroo();

// joey.getMarsupialDescription(); // Marsupial walks on two legs: false

// joey.getKangarooDescription(); } // Kangaroo hops on two legs: true

// }

//

// 2.Örnek Methodların Static olmaması durumunda;

//

// public class Marsupial {

// public boolean isBiped() {

// return false;

// }

//

// public void getMarsupialDescription() {

// System.out.println("Marsupial walks on two legs: "+isBiped());

// }

// }

//

//

// public class Kangaroo extends Marsupial {

// public boolean isBiped() {

// return true;

// }

//

// public void getKangarooDescription() {

// System.out.println("Kangaroo hops on two legs: "+isBiped());

// }

//

// public static void main(String[] args) {

// Kangaroo joey = new Kangaroo();

// joey.getMarsupialDescription(); // Marsupial walks on two legs: true

// joey.getKangarooDescription(); } // Kangaroo hops on two legs: true

// }

// ABSTRACT CLASS, METHOD

// abstract calsslarda body li ve bodysiz methodlar olabilir. bodysiz methodlar abstract class'ın concrete

// subclass'ında bodyli olarak mutlaka overriding yapılması lazım.

// bodyli abstract methodun extends yapılan abstract class tarafından override edilmesi gerekmiyor.

//

// public abstract class Animal {

// protected int age;

// public void eat() {

// System.out.println("Animal is eating");

// }

//

// public abstract String getName();

// }

//

// public class Swan extends Animal {

// public String getName() {

// return "Swan";

// }

// }

// abstract methodlar sadece abstract classlarda oluşturulabilir ve kesinlikle bodyleri olamaz.

// public abstract class Turtle {

// public abstract void swim() {} // DOES NOT COMPILE {} body işareti olduğu için

// public abstract void run(); // COMPİLE

// public abstract int getAge() { // DOES NOT COMPILE

// return 10; }

// }

// }

// non-static, non-final, non-private olan methodlara virtual(gerçek) method denilir.

// abstract classtaki methodların derlenebilmesi için bir subclass'ının olması lazım. Bu nedenle

// abstract class lar final olamazlar,

// public final abstract class Tortoise { // DOES NOT COMPILE}

// Aynı şekilde abstract methodlar bir concrete class tarafından overriden yapılması lazım.

// method final olursa bu olamayacağından abstract methodlarda final olamazlar.

// public abstract class Goat {

// public abstract final void chew(); // DOES NOT COMPILE

// }

// abstract methodlar, subclass tarafından erişilemeyeceğinden private'ta olamazlar.

// public abstract class Whale {

// private abstract void sing(); // DOES NOT COMPILE}

// }

// subclasstaki method accesmodifier parent class'ınkini daraltırsa hata verir.

// TÜM OVERRİDE İŞLEMLERİNDE BUNU KONTROL ETMELİYİZ. ÇOK ÖNEMLİ BİR KURAL

// public abstract class Whale {

// protected abstract void sing();

// }

//

// public class HumpbackWhale extends Whale {

// private void sing() { // DOES NOT COMPILE private -->protected'i daraltmış olur.

// System.out.println("Humpback whale is singing");

// }

// }

// }

// }

// abstract classta main methodda bir şeyler run etmeye çalışmak hata verir. soyut classta somut işler yapılamıyor.

// public abstract class Eel {

// public static void main(String[] args) {

// final Eel eel = new Eel(); // DOES NOT COMPILE

// }

// }

// bir abstract class diğer bir abstract classın subclass'ı olabilir.

// bir concrat class multilevel olarak abstract classların subclassı ise tüm bodysiz methodları overriding

// etmelidir. bir abstract subclass ta body li olarak method tanımlanmışsa bu methodun concret class tarafından

// override edilmesi gerekmez.

// public abstract class Animal {

// public abstract String getName();

// }

//

// public abstract class BigCat extends Animal {

// public String getName() {

// return "BigCat";

// }

// public abstract void roar();

// }

//

// public class Lion extends BigCat {

// public void roar() {

// System.out.println("The Lion lets out a loud ROAR!");

// }

// }

// DEFAULT İNTERFACE

// interface syntax'ı aşağıdaki gibidir. access modifier public veya default olmalıdır.

// public abstract interface CanFly {

// }

// public interface IsWarmBlooded {

// }

// DEFAULT İNTERFACE VARİABLE

// interface variable syntax aşağıdaki gibidir. public static final

// public interface CanSwim {

// public static final int MAXIMUM\_DEPTH = 100; // int MAXIMUM\_DEPTH = 100; şeklinde de yazılabilir.

// public static String TYPE; // DOES NOT COMPILE static final valuelere değer atanması gerekiyor

// protected abstract boolean UNDERWATER = false; // DOES NOT COMPILE hem protected, hem abstract ikiside olmaz.

// }

// DEFAULT İNTERFACE METHOD

// normalde interface methodlar abstract yani body'siz olur. sonradan eklenen bir özellikle default eklenerek

// kullanmak isteyenler için hazır bodyli method yazma imkanı tanınmış, istemeyenler için override edebilme

// imkanı da sunulmuş.

// public interface IsWarmBlooded {

// public abstract void fly(int speed); // void fly(int speed) şekilnde de yazılabilir. default olarak

// // public ve abstract keywordlarını kendisi ekler.

// boolean hasScales();

// public default double getTemperature() { // default keyword eklenerek body'li method oluşturabiliyoruz.

// return 10.0; // Bu sadece interface'a has bir method yazımı.

// method public olduğundan başka packagelardan ulaşılabilir

// default yazıyorsa mutlaka body olmalı yoksa COMPİLE ERROR

// public default yazıyorsa mutlaka interface olmalı yoksa COMPİLE ERROR

// default methodlar body li olduğu için abstract olamaz, override yapılacağı içinde static ve final olamazlar.

// interface de hem abstract hem de default methodlar private ve protected olamazlar.

// interface de tüm methodlar publictir yazılmazsa default olarak eklenir.

// İNTERFACE VARİABLE SYNTAX --> public static final BU DARALTILAMAZ coveriant olması lazım

// İNTERFACE METHOD SYNTAX --> public abstract BU DARALTILAMAZ coveriant olması lazım

// STATİC İNTERFACE METHOD

// Interface'lerde aşağıdaki gibi bir yazımla static methodlarda oluşturulabiliyor.

// hem default hem static olamıyor. interface'de body'li method yazılacakcsa ya default yada static keywordları

// kullanılması gerekiyor. method'un override edilmesi düşünülüyorsa default method kullanıyoruz.

// public interface Hop {

// static int getJumpHeight() {

// return 8;

// }

// }

//

// public class Bunny implements Hop {

// public void printDetails() {

// System.out.println(Hop.getJumpHeight());

// }

// }

// Sadece static interface method ve private instance methodlarda hiding yapar, overriding yapmaz.

// POLYMORPHİSM

// public class Primate {

// public boolean hasHair() {

// return true;

// }

// }

//

// public interface HasTail {

// public boolean isTailStriped();

// }

//

// public class Lemur extends Primate implements HasTail {

// public boolean isTailStriped() {

// return false;

// }

// public int age = 10;

//

// public static void main(String[] args) {

// Lemur lemur = new Lemur();

// System.out.println(lemur.age); // 10

// HasTail hasTail = lemur; // Lemur bir HasTail'dir şeklinde de okunabilir. Child Parent ilişkisinde is A olmalı.

// System.out.println(hasTail.isTailStriped()); // false

// Primate primate = lemur;

// System.out.println(primate.hasHair()); // true

// }

// }

// HasTail hasTail = lemur;

// System.out.println(hasTail.age); // DOES NOT COMPILE hastail objesi ile age variable'ına ulaşamayız.

// Primate primate = lemur;

// System.out.println(primate.isTailStriped()); // DOES NOT COMPILE Primate class'ında böyle bir method yok ulaşamayız.

//

// Lemur lemur = new Lemur();

// Object lemurAsObject = lemur;

//

// Primate primate = lemur;

// Lemur lemur2 = primate; // DOES NOT COMPILE primate parent olunca casting yapmadan atama olmuyor

// Lemur lemur3 = (Lemur)primate; // Primate bir Lemur olmadığında cast yapmak gerekiyor.

// System.out.println(lemur3.age);

// public class Bird {

//

// }

//

// public class Fish {

// public static void main(String[] args) {

// Fish fish = new Fish();

// Bird bird = (Bird)fish; // DOES NOT COMPILE iki class arasında parent child ilişkisi yok

// }

// }

// }

//

// public class Rodent {

//

// }

// public class Capybara extends Rodent {

// public static void main(String[] args) {

// Rodent rodent = new Rodent();

// Capybara capybara = (Capybara)rodent; // Throws ClassCastException at runtime parent child ilişkisi

// // var ama referansta herhangi bir method ve variable olmayınca bu hatayı veriyor

// }

// }

// }

//}

// public class Bird {

// public String getName() {

// return "Unknown";

// }

// public void displayInformation() {

// System.out.println("The bird name is: "+getName());

// }

// }

//

// public class Peacock extends Bird {

// public String getName() {

// return "Peacock";

// }

//

// public static void main(String[] args) {

// Bird bird = new Peacock();

// bird.displayInformation(); // COMPİLE The bird name is: Peacock

// }

// }

// Polimorfizmde çokça kullanılan bir kullanım şekli

// public class Reptile {

// public String getName() {

// return "Reptile";

// }

// }

//

// public class Alligator extends Reptile {

// public String getName() {

// return "Alligator";

// }

// }

//

// public class Crocodile extends Reptile {

// public String getName() {

// return "Crocodile";

// }

// }

//

// public class ZooWorker {

// public static void feed(Reptile reptile) {

// System.out.println("Feeding reptile "+reptile.getName());

// }

// }

//

// public static void main(String[] args) {

// feed(new Alligator()); // Alligator

// feed(new Crocodile()); // Crocodile

// feed(new Reptile()); // Reptile

// }

// }

// polimorphism ile ilgili önemli kurallar

// Math math = new Geometry(); // geometri classından referansı Math classı olan math objesi oluşturuldu.

// math ile bir value çağrılsa Math Classındaki ve ne kadar superclassı varsa hepsinin value'si ve methodu

// varsa çağırabilir. methodlardan overriding yapılan varsa yukarıdan aşağıya doğru en son override yapılanı çağırır.

// Math math = new Geometry(); // Geometry Class'ının kendisi ve extends yaptığı tüm Classlar için Math math

// gibi referans yazılabilir.

// Child'den parenta ulaşılabiliyor, çünkü miras alıyor, tam tersi Parenttan childe ulaşmak gerekirse aşağıdaki

// gibi explicit cast yapılması gerek.

// Geometry geometry = (Geometry)new Math(); Math'ı Geometriye dönüştür ==> Geometri bir Geometridir.

}

## CHAPTER 06

**public** **class** OcaChapter06Notlari {

// iki tür exeption var. checked(kontrol edilebilen, öngörülebilen) ve unchecked(kontrol edilemeyen, öngörülemeyen).

// RunTimeExeption gördüğümüzde bunun bir unchecked Exeption olduğunu anlarız.

// void fall() throws Exception { // throws Exeption: bu method bir exeption atabilir demek.

// throw new Exception(); // bir exeption atmak istiyorum demek.

// }

// throw new Exception("Ow! I fell."); // parametresiz yazarsam varsayılanı kullanır, parametre içine buradaki gibi

// kendi mesajımıda yazabilirim.

// TRY - CATCH SYNTAX

// try {

// fall();

// }

// catch (Exception e) { // catch ( exceptiontype isim )parametre içinde hangi tip exeption yakalanacağı belirtiliyor

// System.out.println("get up");

// }

// try bloğundaki kod çalıştırılır, eğer bu kod catch bloğunda listelenen exeptionlardan birini atarsa çalışmayı

// durdurur ve catch ifadesine gider. Kod herhangi bir exeptionla karşılaşmıyorsa catch cümlesi çalışmaz.

// catch block{} ve catch clause(cümle) aynı manaya geliyor, sınavda bu şekilde de ifade edilebiliyor.

// try {// DOES NOT COMPILE fall(); try ifadesi koşullu cümledir, koşuldan sonra yapacağı işlem tarif edilmezse

// hata verir.

// }

// try - catch - finally block 4 KURAL

// try catch finally şeklinde düzgün bir sıralama ile yazıldığında exeption olursa catch bloğunu çalıştırır ve

// finally bloğunu atlar. exeption olmazsa finally bloğunu çalıştırır. Bu kuralın bir istisnası vardır, try veya

// catch blok içerisine System.exit(0) yazılırsa dur programdan çık der ve finally blok çalıştırılmaz.

// void explore() {

// try {

// seeAnimals();

// fall();

// } catch (Exception e) {

// getHugFromDaddy();

// }

// finally {

// seeMoreAnimals();

// }

//

// goHome();

// }

// try finally catch şeklinde sıralama yanlış yazılırsa DOES NOT COMPİLE

// try { // DOES NOT COMPILE:

// fall();

// }

// finally {

// System.out.println("all better");

// }

// catch (Exception e) {

// System.out.println("get up");

// }

// try blok tek başına catch blok veya final blok olmadan yazılırsa DOES NOT COMPİLE

// try { // DOES NOT COMPILE

// fall();

// }

//try finally blok şeklinde yazılırsa bir Exeption atıp atmamasına bakılmaksızın finally blok her zaman çalıştırılır.

// try {

// fall();

// }

// finally {

// System.out.println("all better");

// }

// try catch bloklarında exeptionın checked veya unchecked olması önemlidir. RunTimeExeption bir unchecked

// exeptiondır ve direk veya dolaylı subclasslarda unchecked exeption olur.

// class AnimalsOutForAWalk extends RuntimeException { }

// class ExhibitClosed extends RuntimeException { }

// class ExhibitClosedForLunch extends ExhibitClosed { }

// try catch bloğunda ard arda farklı catch blokları spesifik Classlardan genel Classlara yani subclassdan

// superclasslara doğru yazılmalıdır yoksa sıralaması yanlış olan blokta hata verir. Classlar arasında inheritance

// durumu yoksa sıralama önemli değildir. Sıralama doğru yazıldığında birden fazla exeption olması durumunda ilk

// catch blok çalışır diğerleri çalışmaz.

// public void visitMonkeys() {

// try { seeAnimal();

// }

// catch (ExhibitClosed e) {

// System.out.print("not today");

// }

// catch (ExhibitClosedForLunch e) {// DOES NOT COMPILE

// System.out.print("try back later"); }

// }

// }

// Runtime exeption : RunTimeExeption'ının subclassıdır(child). böyle bir hata vereceği öngürelmediğinden, unchecked

// Exeption olduğundan önceden declare edilemez ama declare edilirse sorun yapmaz. Program (catch) yakalayabilir.

// ArithmeticException --> bir sayının sıfıra bölünmesi

// Exception in thread "main" java.lang.ArithmeticException: / by zero

// ArrayIndexOutOfBoundsException --> geçersiz array dizeyi kullanılması

// Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -1

// ClassCastException --> class cast yaparken stringi booleana çevirmek gibi olmayacak bir şey yazıldığında

// Exception in thread "main" java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Boolean

// IllegalArgumentException --> yumurta sayısını girerken -2 gibi bir değer yazılması. kod çalışır ama programcı

// tarafından bu tür durumlar için try catch yazılması gerekir.

// Exception in thread "main" java.lang.IllegalArgumentException: # eggs must not be negative

// NullPointerException --> instance variable ve methodların referansı null olduğu zaman atılır.

// null değerli bir Stringin length() inin sorulması durumunda bu exeptionı atar.

// Exception in thread "main" java.lang.NullPointerException

// NumberFormatException --> Integer.parseInt("abc"); Bir stringi method yardımıyla integera dönüştürmeye çalışma

// gibi durumlarda atılır.

// Exception in thread "main" java.lang.NumberFormatException: For input string: "abc"

// Checked exeption : Exeption'ının subclassıdır, RunTimeExeption'ının değil. böyle bir hata vereceği

// öngürelebildiğinden, checked Exeption olduğundan önceden declare edilmelidir. Program (catch) yakalayabilir.

// FileNotFoundException --> kod var olmayan dosyaya başvurduğunda atılır.

// IOException --> bir dosyayı okurken veya yazarken sorun olduğunda atılır.

// Error : Error'ın subclassıdır. önceden declare edilemez. Program (catch)

// yakalayabilir(handle edilebilir ama iyi bir fikir değildir)

// ExceptionInInitializerError --> statik initiliziar(blok)ta bir exeption atılırsa java tüm sınıfı yükleyemediği

// için bu hatayı atar.

// Exception in thread "main" java.lang.ExceptionInInitializerError yazar.

// StackOverflowError --> bir method kendini çağırdığında sonsuz yenileme oluştuğunda bu hata atılır. sonsuz

// döngüden daha iyidir, daha hızlı sistem tarafından durdurulur.

// Exception in thread "main" java.lang.StackOverflowError yazar.

// NoClassDefFoundError --> Java çalışma zamanında classı bulamadığında ortaya çıkar. Sınavda sorulmuyor.

// JVM tarafından yakalanamayan programcı tarafından catch yapılması gereken exeptionlar :

// java.io.IOException

// NumberFormatException --> RunTimeExeptiondır.

// Exeption lar checked(öngürülebilir)dir. throws NoMoreCarrotsException yazınca bunun mutlaka handle(düzeltilmesi)

// veya main methodda declare edilmesi gerekir. aşağıda handle edilmediğinden main method içinde hata verdi.

// class NoMoreCarrotsException extends Exception {}

// public class Bunny {

// public static void main(String[] args) {

// eatCarrot();// DOES NOT COMPILE

// }

//

// private static void eatCarrot() throws NoMoreCarrotsException { }

// }

// }

//

// Düzgün çalışması için main methodda exeption olabilir diye throws ile declare ediyoruz veya

// methodu try catch ile yazarak handle ediyoruz. NoMoreCarrotsException ile karşılaşırsan sad rabbit yaz.

// public static void main(String[] args) throws NoMoreCarrotsException {// declare exception

// (ya bu şekilde deklare edeceğiz yada aşağıdaki şekilde try catch yapmak zorundayız)

// eatCarrot();

// }

//

// public static void main(String[] args) {

// try {

// eatCarrot();

// } catch (NoMoreCarrotsException e ) {// handle exception

// System.out.print("sad rabbit");

// }

// }

// SONUÇ OLARAK; bir method Exeption atıyorsa ;

// 1. hem methodda hem de main methodda bu method bir Exeption atabilir manasında throws exeption ifadesi

// method parametresinden sonra belirtilmek zorunda.

// public void good() throws Exception { } public static void main(String[] args) Exception { }

// 2. yada main methodda method try catch block içinde çağrılıp declare edilmek zorunda

// yoksa compile etmez.

// ulaşılamayan boşa düşen bir declare olursa compile etmez.

// public void bad() {

// try {

// eatCarrot();

// }

// catch (NoMoreCarrotsException e ) {// DOES NOT COMPILE // bir exeption declare edilmiş ama bad() method

// ismi ile çağrılmadığından ulaşılamayan bir declare olunca hata verir.

// System.out.print("sad rabbit");

// }

// }

//

// public void good() throws NoMoreCarrotsException {

// eatCarrot();

// }

// private static void eatCarrot() { }

// }

// parent child ilişkisine bakılmadan methodlar herhangi bir RunTimeExeption belirtebilir.

// class Hopper {

// public void hop() { }

// }

//

// class Bunny extends Hopper {

// public void hop() throws IllegalStateException { }

// }

// superclassta herhangi bir exeption declare edilmeden sublclassda overriding yapılmak istenirse compile etmez.

// class CanNotHopException extends Exception { }

// class Hopper {

// public void hop() { }

// }

//

// class Bunny extends Hopper {

// public void hop() throws CanNotHopException { } // DOES NOT COMPILE}

// }

// subclassta veya interfaceoe exeption declare edilir ve sublclassda overriding yapılmak istenirse sorun yaşanmaz.

// class CanNotHopException extends Exception { }

// class Hopper {

// public void hop() throws CanNotHopException { }

// }

//

// class Bunny extends Hopper {

// public void hop() { }

}