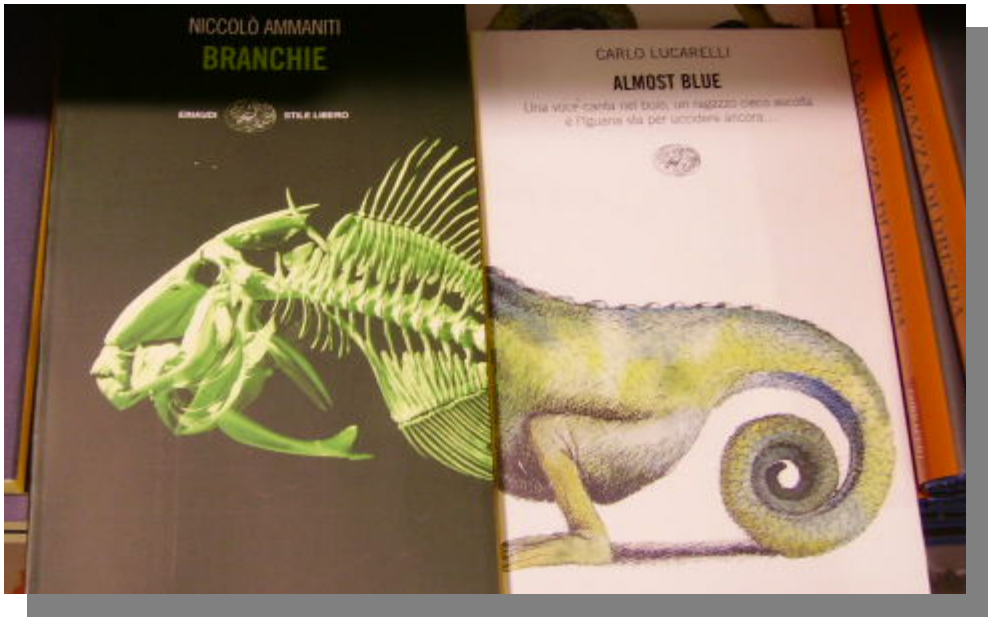
Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması



Çoğunlukla Yaptığım Bazı Temel Hatalar

ForTran yazıp, C kastedip, adına C++ demek

Aşağıdakiler, bir C programını C++'a çevirmek için yeterli değildir; aralarındaki fark "kurgu"dadır, komutta değil:

gcc a.c -o a → g++ a.cxx -o a

#include<bisey.h> → #include<bisey>

printf → cout

scanf → cin

int a=0; → int a(0);

v.b.

Örnek: SIYIRDUiNO

1) ForTran'a daha yakın ama teknik olarak C++
2) "class" kelimesi yok (sadece struct var)
3) OO ihtiyaca iyi ornek (ör: gemi, rakı, bomba)
4) Mimari != Uygulama

```
  }
  memcpy(new_buf, sound->samples, sound->length);
  cvt.buf = new_buf;
  if (SDL_ConvertAudio(&cvt)<0){
    printf("Audio conversion error : %s\n", SDL_GetError());
    free(new_buf);
    SDL_FreeWAV(sound->samples);
    return 1;
  }
  SDL_FreeWAV(sound->samples);
  sound->samples = new_buf;
  sound->length = sound->length * cvt.len_mult;
  return 0;
}

//_____________________________________________
void ClearPlayingSounds()
{
  for (int i=0 ; i<MAX_PLAYING_SOUND ; i++){
    playing[i].active = 0;
  }
}

//_____________________________________________
int PlaySound(sound_p sound)
{
    int i;
  for (i=0 ; i<MAX_PLAYING_SOUND ; i++ ){ // find free slot for sound
    if (playing[i].active == 0) break;
  }
  if (i == MAX_PLAYING_SOUND) return 1; // no free slot for this spound
  SDL_LockAudio();
  playing[i].active = 1;
  playing[i].sound = sound;
  playing[i].position = 0;
  SDL_UnlockAudio();
  return 0;
}

//_____________________________________________
typedef struct player_s {
    int x, y;              // coordinates
    double velocity_x;     // velocity in pixels per frame
    double velocity_y;
    double accel_x;        // acceleration in pixels/frame^2
    double accel_y;
    bool movable;
    bool dead;
    int angle;             // Current direction
} player_t, *player_p;

//_____________________________________________
typedef struct nme_s {
    int x, y;              // coordinates
    double maxVel;         // maximun velocity
    double velocity_x;     // velocity in pixels per frame
    double velocity_y;
    double accel_x;          // acceleration in pixels/frame^2
    double accel_y;
    bool movable;
    bool dead;
    int angle;
} nme_t, *nme_p;

//_____________________________________________
typedef struct bmb_s {
    int x, y;              // coordinates
    double velocity_x;     // velocity in pixels per frame
    double velocity_y;
```

Çoğunlukla Yaptığım Bazı Temel Hatalar

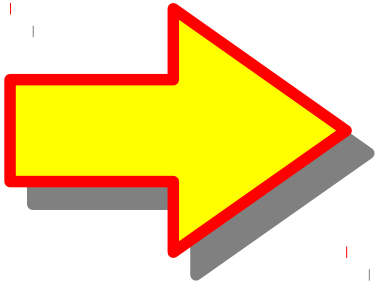
Nesne Yönelimli (OO) Kütüphane Kullanmak != C++ Dilinde Programlamak

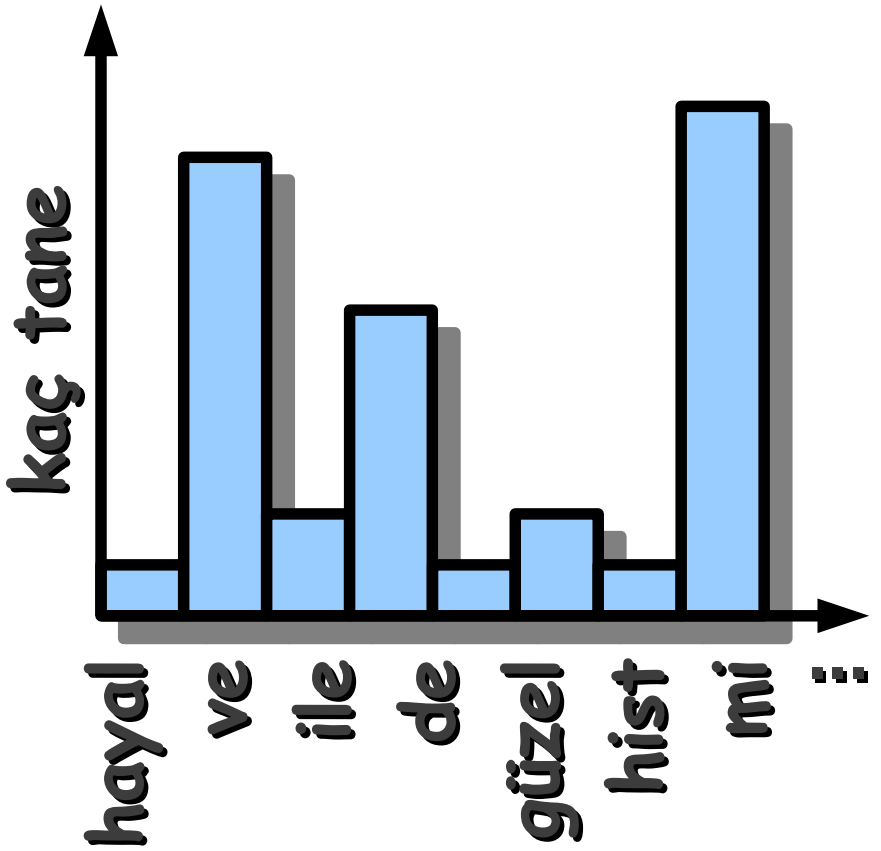
Nesne kütüphanesi kullanmak, class kütüphanesi kullanmak, ROOT kullanmak, v.b. etkinlikler, C++ programlamak anlamına gelmek zorunda değildir; teknik olarak öyle olsa bile (yani g++ gibi bir C++ derleyicisi ile derliyor olsanız bile).

Kullandığınız kütüphaneyi yazan kişinin yaptığı ancak C++ programlamak olabilir (ve bu çok iyi bir şeydir).

Yazar kelime zenginliği belirleyici:
Hangi dildir bu ?

```
#içer<hayalKütüphanesi>
tamsayı ana() {
    m = yükle("makale.txt");
    f = kelimeSay(m);
    çizdir(f);
    çık(0);
}
```





Çoğunlukla Yapıtığım Bazı Temel Hatalar

Nesne Yönelimli (OO) Kütüphane Kullanmak != C++ Dilinde Programlamak

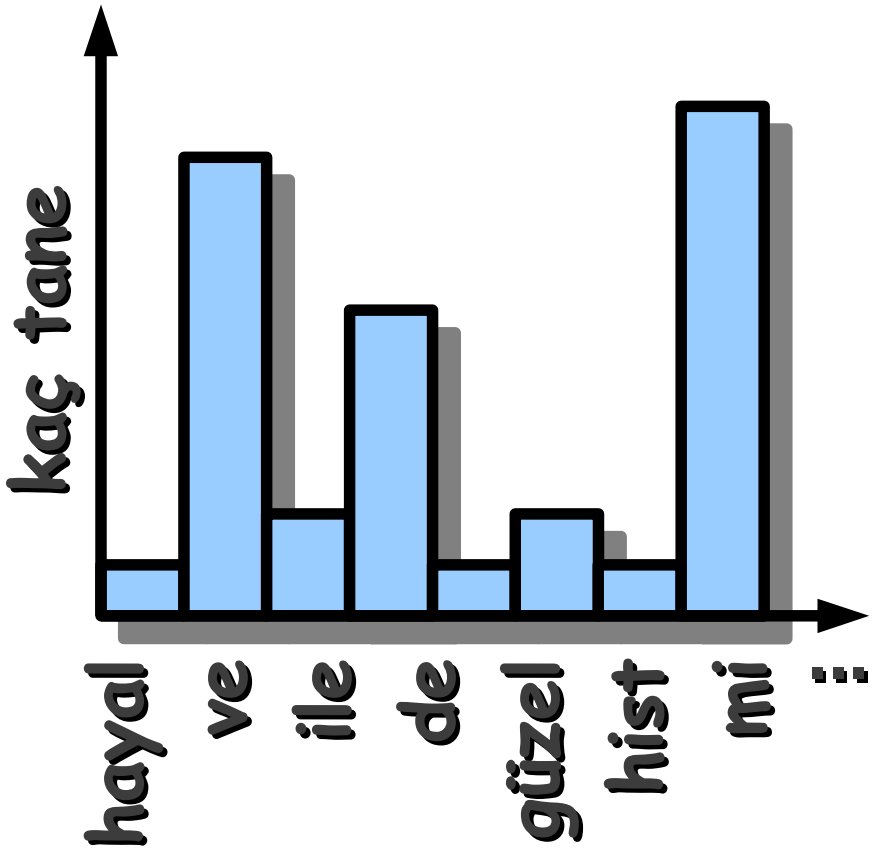
Yazarların yazma yöntemlerini analiz etmek için programlama dili, algoritma, mimari, platform ve yöntem kısıtlaması olmaksızın geliştirilecek bir programa ihtiyaç vardır.

Her bir kelimenin, metin içindeki kullanım sayılarını aşağıda görüldüğü gibi bir histograma dolduran programı yazınız.

Bunu siz nasıl yapardınız (ÖDEV) ?

```
struct { char kelime[];
         int sayi;
};   → ?
```

C değil ama gerçekten C++ kullanarak nasıl yapılabilir ?



Olası Çözüm #1

C'nin struct'ını kullanmak (?)

Başka bir şey ?

< Ödev sonucunuz burada görüntülenecek >

Olası Çözüm #2

C++'ın STL kütüphanesindeki map ya da python'daki dictionary taklit edilir):

- Üyeleri isim ve değer olan, pair adlı bir struct cinsinden üyeleri olan bir dizi tanımla (Satır 25, vec)
- Bu dizinin "indis" operatörüne nasıl cevap vereceğini belirle (Satır 32, operator[])
- Metin kütüğünü, boşluklarla ayrılmış kelimeler olarak oku (Satır 60-62)
- Her kelimeye karşılık o kelimenin değerini bir artır (Satır 63)

```
16
17 //_____________________________________________
18 class structVector {
19     private:
20         struct pair {
21             string name;
22             int val;
23             pair(string n="", int v=0) : name(n), val(v) {}
24         };
25         vector<pair> vec;
26     public:
27         int& operator[](string&);
28         void print_all();
29 };
30
31 //_____________________________________________
32 int& structVector::operator[](string& str) {
33     for (vector<pair>::iterator p = vec.begin() ; p != vec.end() ; p++)
34         if (str == p->name)
35             return p->val;
36     vec.push_back(pair(str, 0));
37     return vec.back().val;
38 }
39
40 //_____________________________________________
41 void structVector::print_all() {
42     for (vector<pair>::iterator p = vec.begin() ; p != vec.end() ; p++)
43         cout << p->name << "\t\t\t" << p->val << endl;
44 }
45
46 string buf;
47 structVector object;
48
49 //_____________________________________________
50 void readWord(const string& s) {
51     buf = s;
52     object[buf]++;
53 }
54
55 //_____________________________________________
56 int main() {
57    string file;
58    cout<<"Enter filename:";
59    cin>>file;
60    ifstream is(file.c_str());       // Read from file
61    istream_iterator<string> ii(is); // Line itself
62    istream_iterator<string> (eos);  // End of line
63    for_each(ii, eos, readWord);
64    object.print_all();
65 }
```

Örnek C++'ın yaratıcısının kitabından alınmıştır: The C++ Programming Language, Bjarne Stroustrup, Special Edition

Olası Çözüm #3

C++'in STL kütüphanesindeki map'i kullan (taklit etmeden)

- Üyeleri isim ve değer tutacak olan, histo adlı bir map yarat (Satır 15)
- Metin kütüğünü, boşluklarla ayrılmış kelimeler olarak oku (Satır 38-40)
- Her kelimeye karşılık o kelimenin değerini bir artır (Satır 43)
- Her map üyesini (kelimeyi ve metinde kaç kez geçtiğini) bastır (Satır 44)

Kendi çözümünüz ile aradaki farkı düşünün: bu, paradigmal bir değişiklik mi yoksa sadece yiğit/yoğurt mu ?

```
 8 #include<iterator>
 9 #include<iostream>
10 #include<algorithm>
11 #include<fstream>
12 #include<map>
13 using namespace std;
14
15 map<string, int> histo; // which holds ordered st
16
17 //____________________________________
18 void collect(const string& s) {
19     // Increments each 'word' bin
20     histo[s]++;
21 }
22
23 //____________________________________
24 void print(const pair<const string, int>& p) {
25     // Outputs each pair
26     cout<<p.first<<"\t\t\t"<<p.second<<endl;
27 }
28
29 //____________________________________
30 int main() {
31
32    // Get filename to read from
33    string file;
34    cout<<"Enter filename:";
35    cin>>file;
36
37    // Set the stream
38    ifstream is(file.c_str());       // Read from
39    istream_iterator<string> ii(is); // Line itsel
40    istream_iterator<string> (eos);  // End of li
41
42    // Calculate the frequency
43    for_each(ii, eos, collect);
44    for_each(histo.begin(), histo.end(), print);
45 }
```

Örnek C++'ın yaratıcısının kitabından alınmıştır: The C++ Programming Language, Bjarne Stroustrup, Special Edition

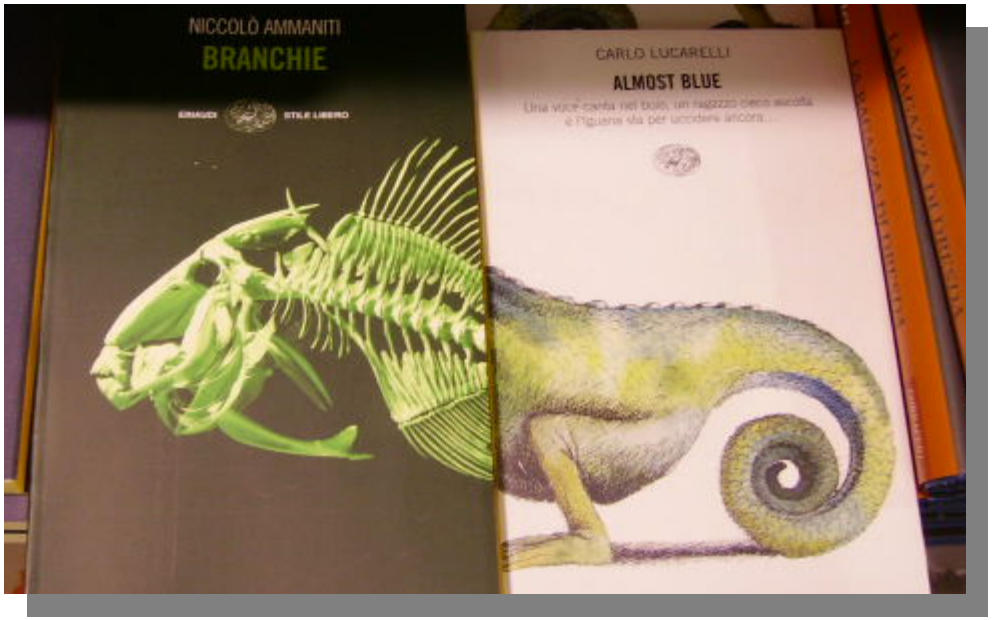
Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması

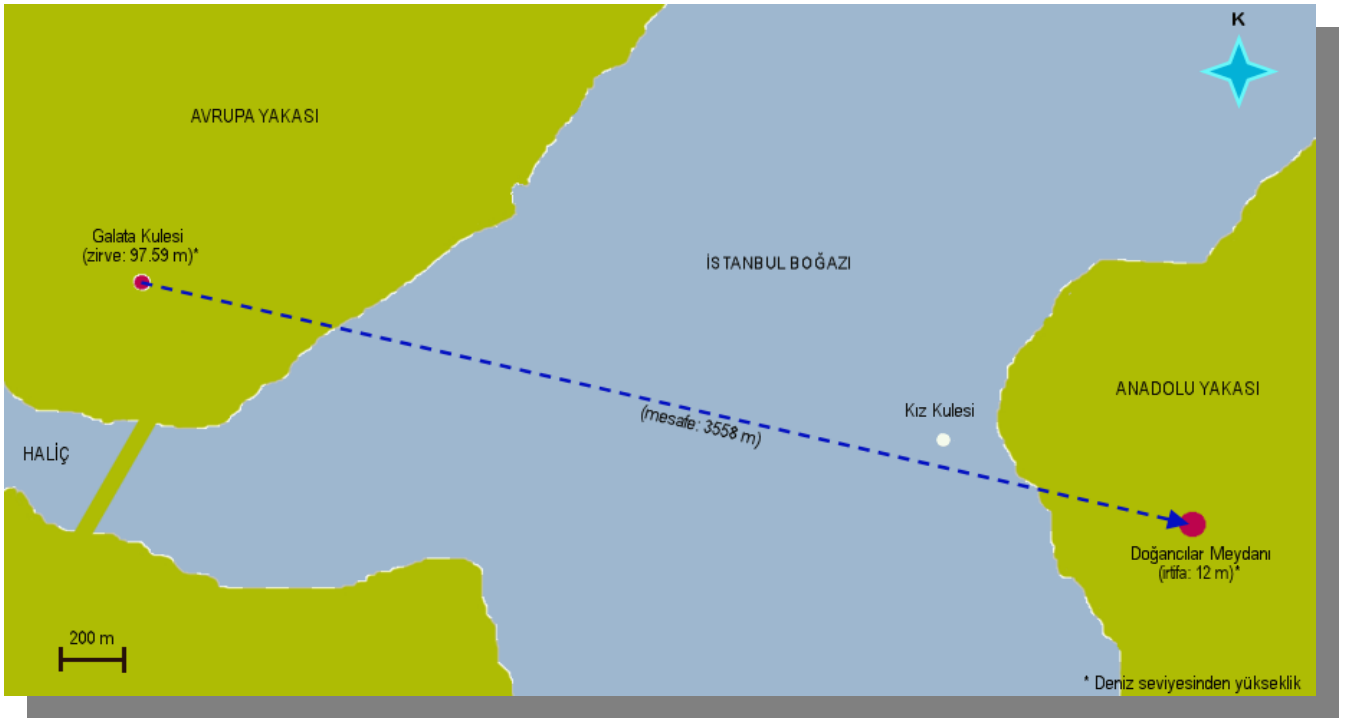


Amaç

Geliştirici Ne İster ?

Bir yazılımın verimli çalışmasından daha önemli olan şeyler var mıdır ?***

parçalanabilirlik *(modularity),* ***kullanıcı için kolaylık*** *(user-friendliness),*
doğruluk *(correctness),* ***geliştirici mesaii*** *(programmer time),*
bakım kolaylığı *(maintainability),* ***sadelik*** *(simplicity),*
işlevsellik *(functionality),* ***genişletilebilirlik*** *(extensibility),*
sağlamlık *(robustness),* ***güvenilirlik*** *(reliability),*
...? (..?)



Temelde

Geliştirici ne yapar ?

Ne ?

Sorunu...

anlamak için **BÖL**,
bölünmüş ve küçük olanı **ANLA**,
anladığına HAKiM OL,
hakimiyetin altındakileri BİRLEŞTİR = **Çözüm**

Temelde nasıl ?

Paradigmanı seç...

Sıralı programlama (procedural)
Nesne yönelimli programlama (object-oriented)
Bakış açısı yönelimli programlama (aspect-oriented)
...

Temelde

Geliştirici ne yapar ?

(sanırım)

Yazmaya başlamadan önce:

Düşünür ve algoritmasını kalemle kağıda çizer (mantık akışı)

Düşünür ve kağıt üstünde çalıştığından emin olur

Düşünür ve sonra yazmaya başlar

(bence)

Programlama = Düşünmek = Tasarım (**Seçkin**)

Kod Yazmak = Ameliye (**Sıradan**)

Dilden bağımsız (Türkçe/Fransızca/C/C++/Perl v.b.)

Hataya en az açık yöntem

Yazılım büyüdükçe **üstel artan bakım güçlüğü**

(Mutlak Doğru !!)

Hiçbir programlama dili, kütüphane ya da araç her zaman en iyi çözüm değildir

Doğru Kararı Başlangıçta Verin

Büyüdükçe üstel artan bakım güçlüğü derken...

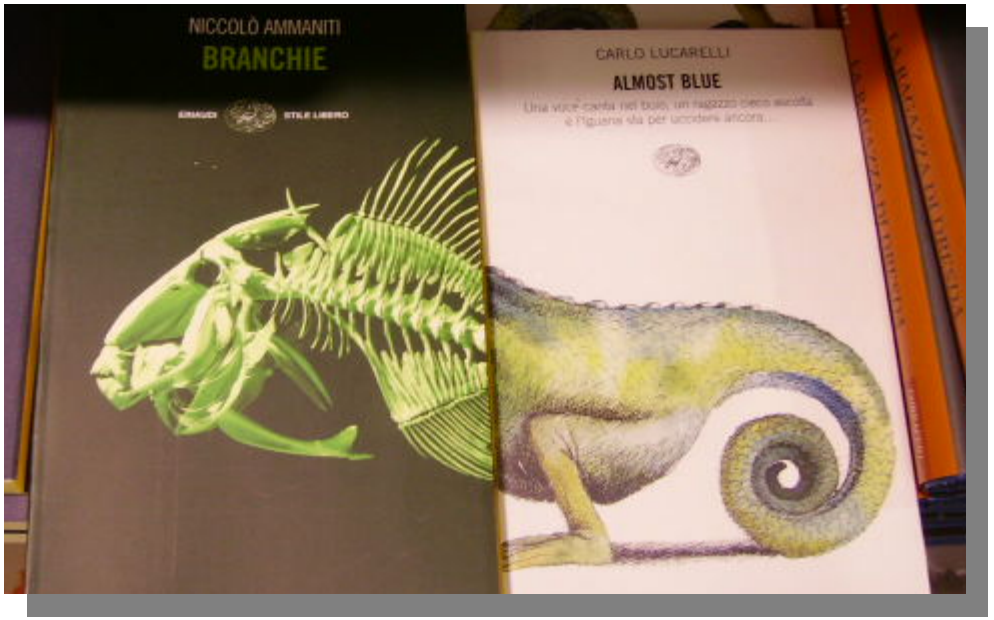
Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması



Sıralı Programlama ***

Örnek: Telefon Defteri Uygulaması

- Yükle işlevinin uygulaması:
  - ***İşaretçi*** yarat (*di* ve *al*)
  - ***Değişken*** yarat (*tane*, *k* ve *I*)
  - Kütük ***okunabiliyor mu*** (*telDef.dat*)
    - Okunabiliyor ise kütüğün sonuna git ve ***boyunu ölç***
  - Ölçülmüş bu boyut, KAYIT isimli struct' ın ***kaç katı***?
  - Kayıt sayısı kadar dönecek bir döngü ile ***tüm kayıtları oku***
  - Okunan ***tüm kayıtları*** dinamik bağlı liste oluşturacak şekilde ***birbirine bağla*** (*dugumEkle* işlevi çağırılıyor)
- Ana program:
  - Menüden yapılan seçime göre *kayıt, arama, sıralama, silme, saklama ve yükleme* işlevlerinden birini ***çağır***.
  - Çıkış (case 7 = seçenek 7) seçilmiş ise uygulamadan ***çık*** (return 0).

File Edit Search Preferences Shell Macro Windows Help

```
207 // yukle()
208 //_____________________________________________________________
209 void yukle() {
210     FILE *di;
211     KAYIT *al;
212     unsigned int tane;
213     int k, i;
214     if ((di=fopen("telDef.dat","r"))==NULL) {
215         printf("\nHATA : Kütük açilamadi !...\n");
216         return;
217     }
218     fseek(di, 0, 2);
219     tane=ftell(di)/(sizeof(KAYIT)-2*sizeof(al));
220     fseek(di, 0, 0);
221     if (tane<1) {
222         printf("\nKütük bos !..\n");
223         return;
224     }
225     printf("\nKütükten yükleniyor..\n");
226     for (int k=0 ; k<tane ; k++) {
227         al=(KAYIT*)malloc(sizeof(KAYIT));
228         if (!al) {
229             printf("\nHATA : Fiziksel hafiza dolu !..\n");
230             break;
231         }
232         fread(al, sizeof(KAYIT)-2*sizeof(al), 1, di);
233         al->sol=NULL;
234         al->sag=NULL;
235         dugumEkle(kok, al);
236     }
237     fclose(di);
238 }
239
240 // main()
241 //_____________________________________________________________
242 int main() {
243     int secim;
244     while(1) {
245         secim = menudenSecimYap();
246         switch(secim) {
247             case 1 :
248                 kayit();
249                 break;
250             case 2 :
251                 arama();
252                 break;
253             case 3 :
254                 agacListele(kok);
255                 break;
256             case 4 :
257                 silme();
258                 break;
259             case 5 :
260                 sakla();
261                 break;
262             case 6 :
263                 yukle();
264                 break;
265             case 7 :
266                 return 0;
267         }
268     }
269 return 0;
270 }
271
```

*** Benzer bir örnek Rifat Çölkesen' in "işte C programlama" adlı kitabında, dinamik bağlı liste başlığında verilmiştir.

Temel İfadeler

Döngüler ve Şart Koşma

- **'bool trackTable (x_1, y_1, x_2, y_2)'** olarak tanımlanmış 4 boyutlu bir dizi, iki RPC algılayıcısı arasında, **A(x_1, y_1)** ve **B(x_2, y_2)** noktaları arasında parçacık izi (track) olup olmadığını, "**evet**" veya "**hayır**" biçiminde aklında tutan bir değişkendir. Yandaki döngü bu **değişkeni sıfırlamak** için yazılmıştır.

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste

kisalt.c | telefonDefteri.C | TTRPC.cxx

```
613     logThis("Histograms are reset.");
614     for (int i=0 ; i<16 ; i++) {
615         for (int ii=0 ; ii<16 ; ii++) {
616             for (int iii=0 ; iii<32 ; iii++) {
617                 for (int iiii=0 ; iiii<32 ; iiii++) {
618                     trackTable[i][ii][iii][iiii]=kFALSE;
619                 }
620             }
621         }
622     }
623     logThis("3D track table is reset.");
```

Ln 613, Col 1 INS

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste Find Replace

binary_vs_thermometer.C | kisalt.c | telefonDefteri.C | TTRPC.cxx

```
588 //
589 void TTRPC::Initialize()
590 {
591 // Initializes :)
592
593     logThis("Initializing...");
594     logThis("-------------------------");
595     if (0!=check4data()) {
596         logThis("Method : Initialize - Can not *open* data file !...");
597     } else {
598         if (0!=readDataFile()) {
599             logThis("Method : Initialize - Can not *read* data file !...");
600         } else {
601             logThis("Data file is read succesfuly.");
602         }
603     }
```

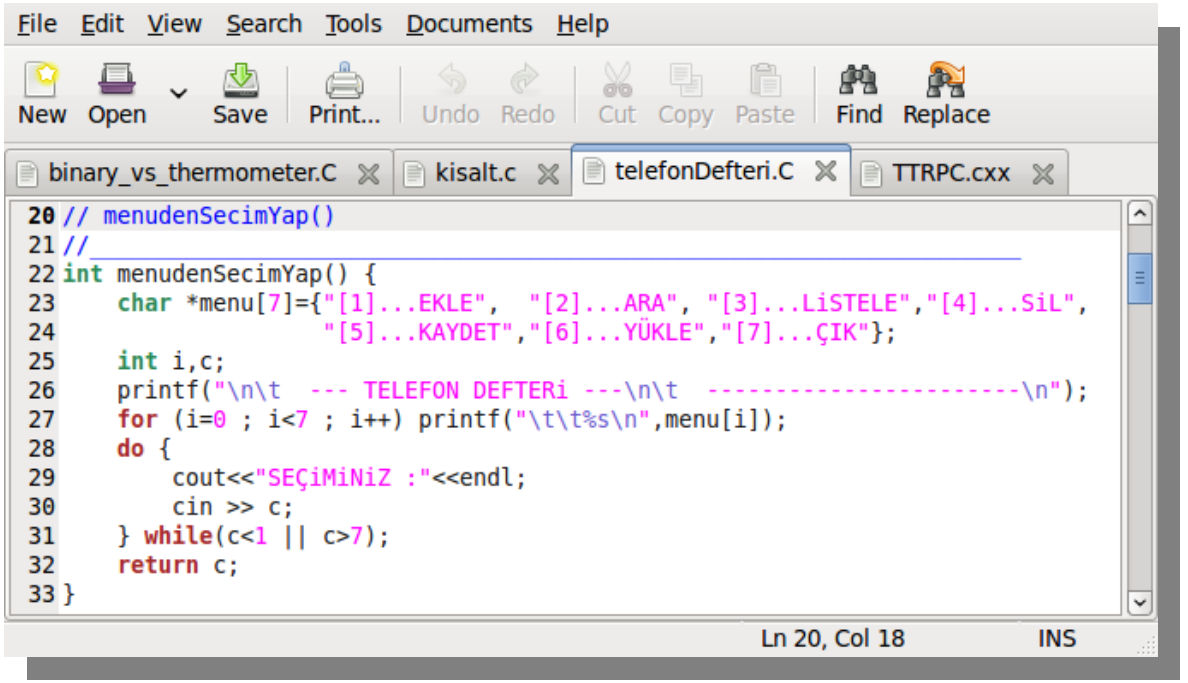
Ln 588, Col 1 INS

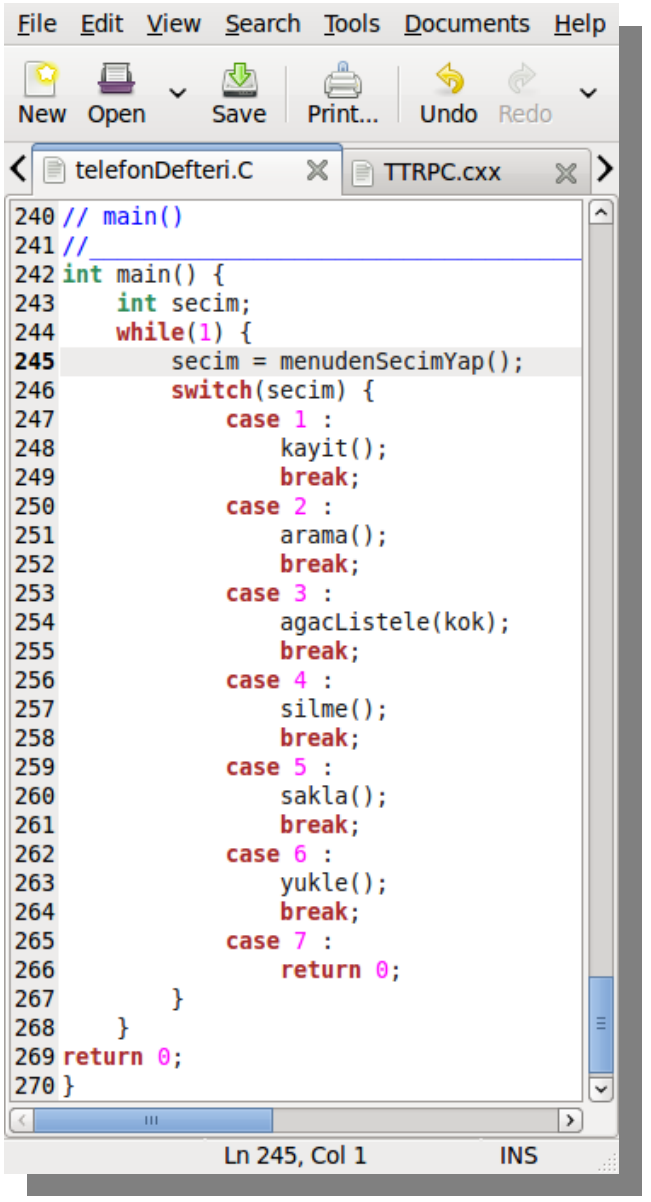
- **check4data()** işlevi başarı ile tamamlan**ma**mışsa (yani sıfır döndür**me**miş ise) log kütüğüne ilgili hatayı yaz; başarılı ise bu sefer veriyi okumayı dene (**readDataFile()** işlevi) ve başarılı olup olmadığına göre log kütüğüne ilgili çıktıları yaz.

Temel İfadeler

Döngüler ve Şart Koşma

- Kullanıcı **devamlı** seçim yapar; while şartının **her zaman** sağlandığına dikkat edin (244. satır)
- Tam sayı olarak tanımlanmış "secim" değişkenine **şart koş**:
  - seçim, 1' e **eşitse** *kayit()* işlevini çağır (248. satır)
  - seçim, 7 **ise** programdan çık (266. satır)



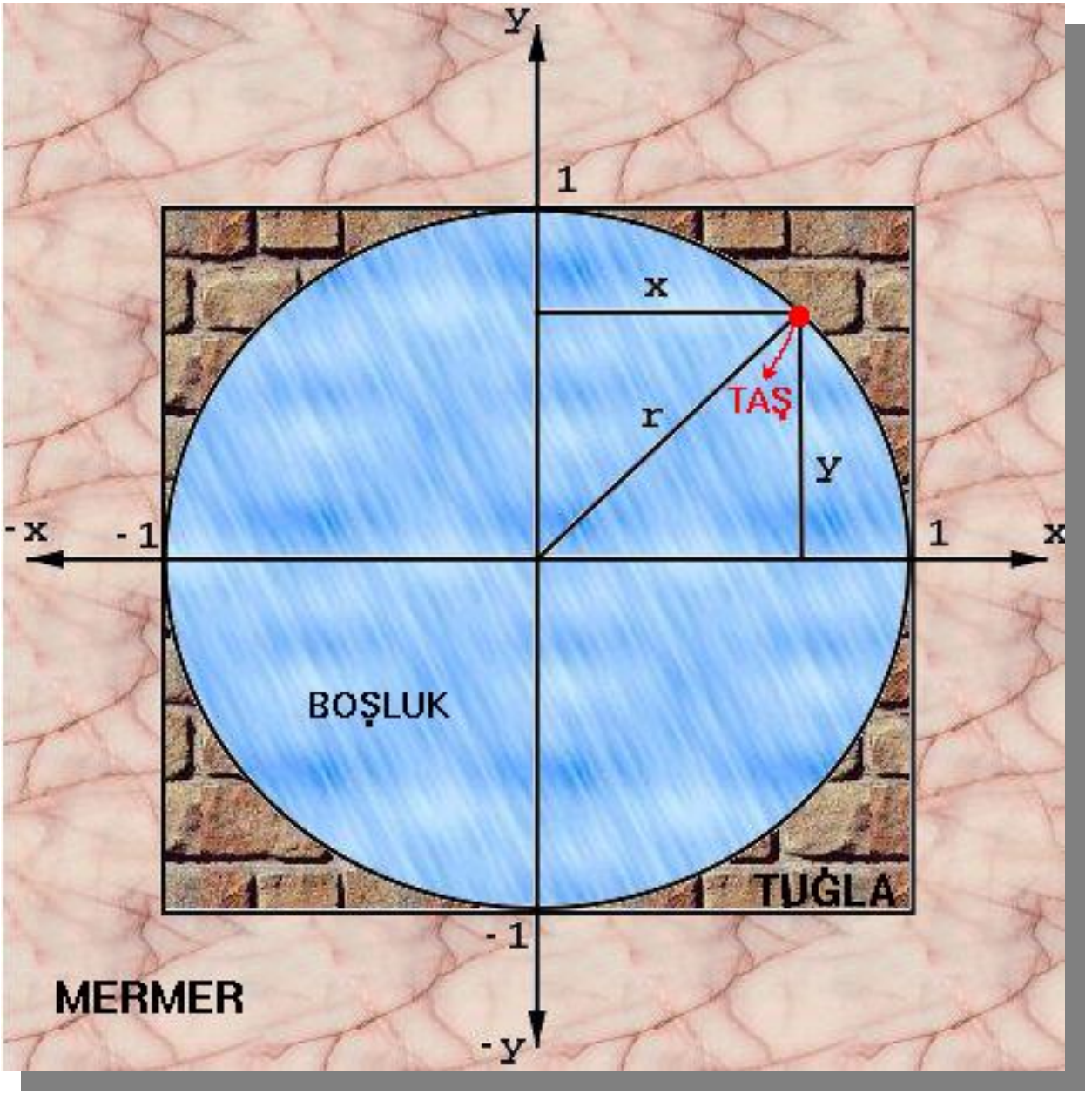


- Telefon defteri uygulamasında kullanıcıya bir komut dizgesi gösteren ve kullanıcının **seçimini kendisini çağırana göndüren** (return c) işlev.
  - **Döngü** ile dizge elemanları basılıyor (27. satır)
  - Kullanıcı 1 ile 7 arasında bir **seçim** girene kadar soru **tekrarlanıyor** (31. satır)
  - Girdi alındığında seçim **çağırana döndürülüyor** (32. satır)

Temel İfadeler

Rastlantısal sayılar ile yaklaşık π sayısını bulmak

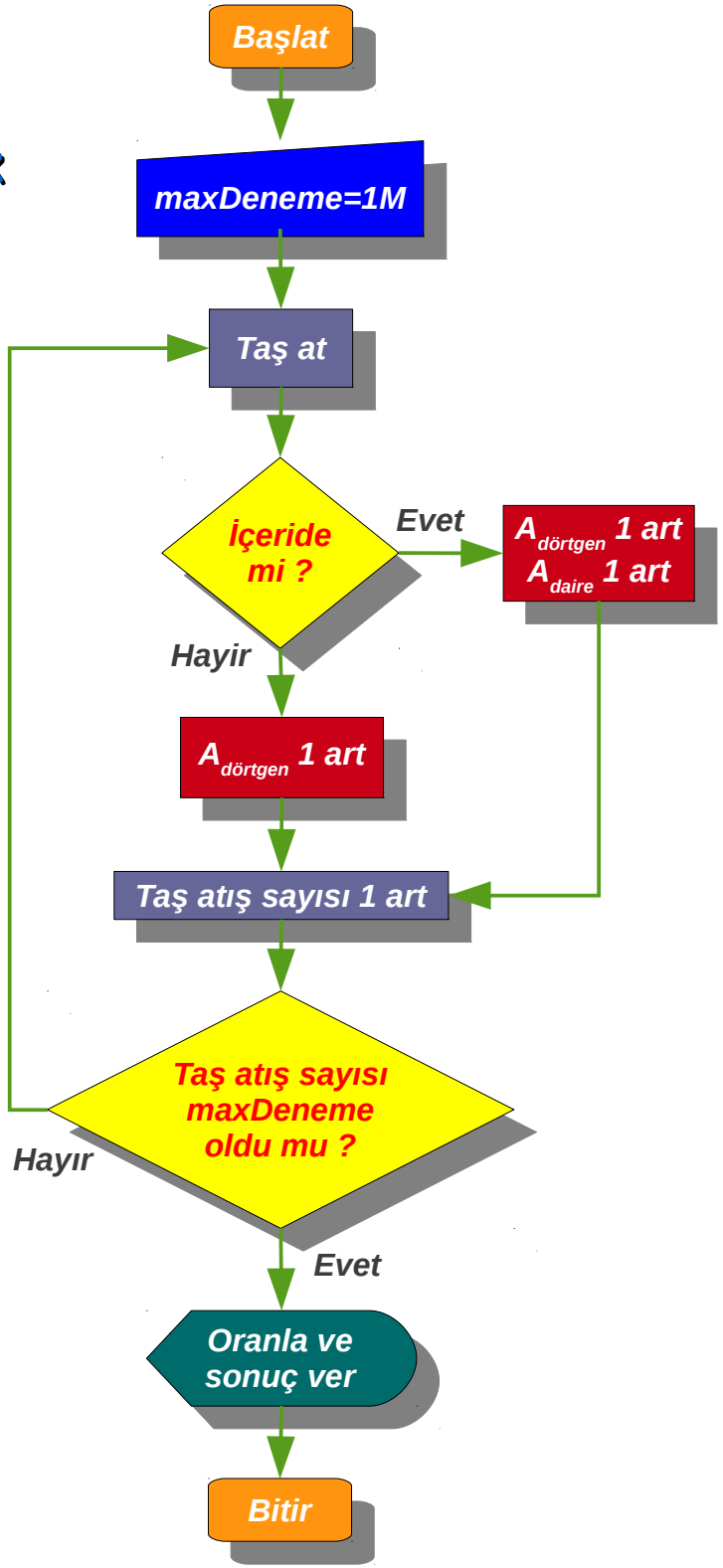
- Üzerinde r yarıçaplı bir **delik** bulunan bir **duvara**, kenarları bu deliğe değecek şekilde kenar uzunluğu 2r olan bir eşkenar **dörtgen** çizilir:
  - $A_{daire} = \pi r^2$
  - $A_{dörtgen} = 4r^2$
  - $A_{daire}/A_{dörtgen} = \pi/4$
- Bu **oranı** hesaplamak için birim uzunluktaki dörtgen içine **rastgele** taş atılıp, taşın dairenin içine denk gelip gelmediğine şart koşulur:
  - $r^2 = 1 < x^2+y^2$
- Şartın sağlandığı atış sayısının, toplam atılan taş sayısına **oranı**, aranılan **sonucun dörtte birine** eşit olacaktır.



Temel İfadeler

Rastlantısal sayılar ile yaklaşık π sayısını bulmak

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    float x, y, r, pi;
    int i, maxDeneme=1000000;
    int daireAlan, kareAlan;

    // RND sayi uretecini baslatiyoruz

    srand(time(NULL));
    kareAlan  = 0;
    daireAlan = 0;
    pi        = 0;

    // Dortgen icine tas atmaya basliyoruz

    for (i=0 ; i<maxDeneme ; i++) {
        x=rand()/(1.0*RAND_MAX);
        y=rand()/(1.0*RAND_MAX);
        r=sqrt(x*x+y*y);
        if (r<=1.0) {
            daireAlan += 1;
            kareAlan  += 1;
        } else {
            kareAlan += 1;
        }
    }

    // Programi bitiriyoruz

    printf("Dairenin icine girenler = %d \n", daireAlan);
    printf("Karenin  icine girenler = %d \n", kareAlan );
    printf("Hesaplanan pi  = %f \n", 4.0*daireAlan/kareAlan);

    return 0;
}
```



Temel İfadeler

Rastlantısal sayılar ile yaklaşık π sayısını bulmak

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste

piBul.cxx | piBul.f

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    float x, y, r, pi;
    int i, maxDeneme=1000000;
    int daireAlan, kareAlan;

    // RND sayi uretecini baslatiyoruz

    srand(time(NULL));
    kareAlan  = 0;
    daireAlan = 0;
    pi        = 0;

    // Dortgen icine tas atmaya basliyoruz

    for (i=0 ; i<maxDeneme ; i++) {
        x=rand()/(1.0*RAND_MAX);
        y=rand()/(1.0*RAND_MAX);
        r=sqrt(x*x+y*y);
        if (r<=1.0) {
            daireAlan += 1;
            kareAlan  += 1;
        } else {
            kareAlan += 1;
        }
    }

    // Programi bitiriyoruz

    printf("Dairenin icine girenler = %d \n", daireAlan);
    printf("Karenin  icine girenler = %d \n", kareAlan );
    printf("Hesaplanan pi  = %f \n", 4.0*daireAlan/kareAlan);

    return 0;
}
```

Ln 18, Col 43 INS

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste

piBul.cxx | piBul.f

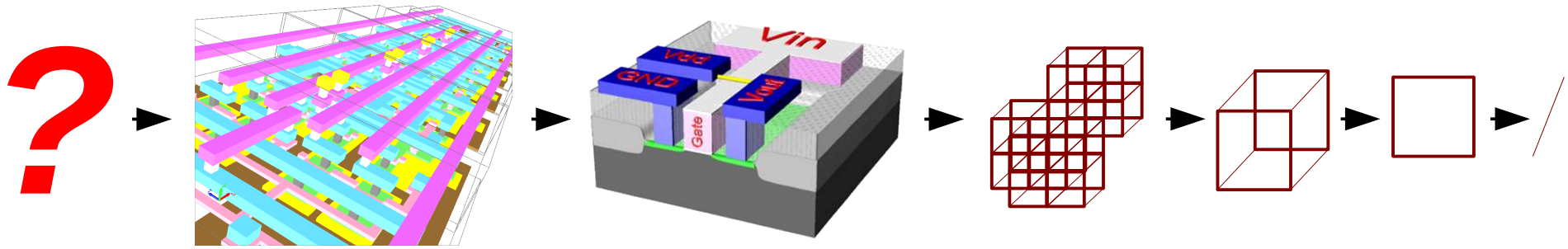
```
      PROGRAM piBul
      IMPLICIT NONE

      REAL x, y, r, pi
      INTEGER i, maxDeneme
      INTEGER daireAlan, kareAlan
      PARAMETER (maxDeneme = 1000000)

c Rnd sayi uretecini baslatiyoruz

      r = RAND(TIME())
      kareAlan  = 0
      daireAlan = 0
      pi = 0

c Dortgen icine tas atmaya basliyoruz

      DO i = 1, maxDeneme

          x = RAND()
          y = RAND()
          r = SQRT(x*x+y*y)

          IF (r.LE.1.0) THEN
              daireAlan = daireAlan + 1
              kareAlan = kareAlan + 1
          ELSE IF (r.GT.1.0) THEN
              kareAlan = kareAlan + 1
          END IF

      END DO

c Programi bitiriyoruz

      PRINT*,'Dairenin icine girenler = ', daireAlan
      PRINT*,'Karenin  icine girenler = ', kareAlan
      pi = 4.0*daireAlan/kareAlan
      PRINT*,'Hesaplanan pi = ', pi
      END
```

Ln 16, Col 38 INS

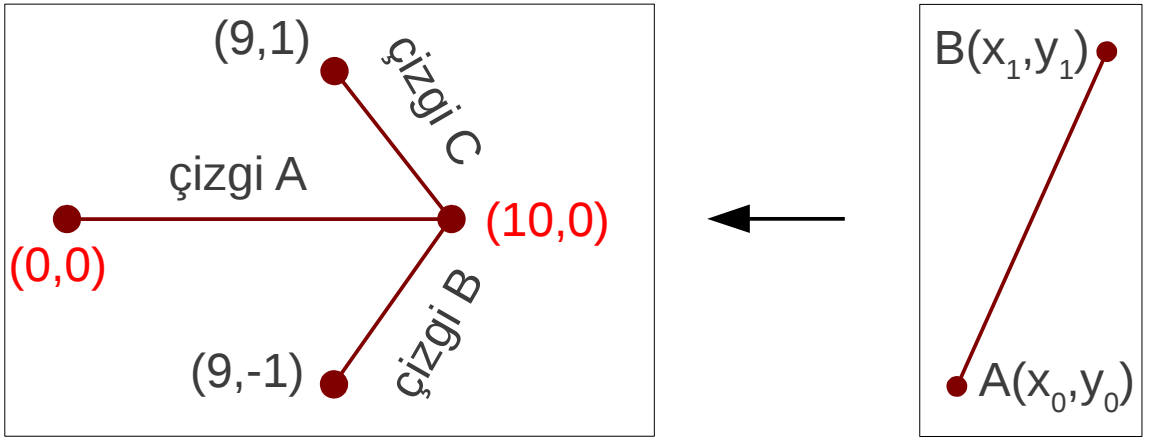
Nesneye Yönelmek

Karmaşadaki sadelik

- ***Karmaşık mimariler, nesne yönelimi*** kullanılarak ***kolayca*** oluşturulabilir
- Bu, ***en verimli*** (*yani başımızı en az ağrıtacak*) yazılım geliştirme şeklidir



- Bir ***ok*** yapmak istediğimizde, aslında programlamamız gereken tek şey bir ***çizgi***dir; *ok*, ***üç tane çizgi*** ile oluşturulabilecek bir *nesne*dir.



Class ***ok*** Class ***cizgi***

Örnek #0

ÇokGen' den türeyen Üçgen ve Dörtgen

- Bir ***çokgen***, ***genişlik*** ve ***yükseklik*** ile tanımlanabilir.
- ***Dörtgen*** ve ***üçgen*** birer ***çokgendir***.
- *Yükseklik* ve *genişlik* bilgisini tutacak olan ***ÇokGen class***' ını yaz ve bu bilgiyi miras alacak (inherit) olan ***üçgen*** ile **dörtgen** class' larını ondan ***türet*** (derive).
- Genişlik ve yükseklikleri sırasıyla 4 ve 5 olan üçgen ile dörtgen ***nesneler*** yarat
- Üçgen ve dörtgen'de, ***DegerAyarla() işlevi*** (fonksiyon) yok ancak kendisinden türedikleri ÇokGen class' ı (base class) sahip
- ÇokGen class'ı ***Alan() işlevine*** (fonksiyon) sahip olmasa da hem Üçgen hem Dörtgen class'ları sahip.

File Edit Search Preferences Shell Macro Windows Help

gen.cxx | cikti

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 class CokGen {
 5   protected:
 6     int genislik, yukseklik;
 7   public:
 8     void DegerAyarla (int a, int b)
 9       { genislik=a; yukseklik=b;}
10   };
11
12 class Dortgen: public CokGen {
13   public:
14     int Alan ()
15       { return (genislik * yukseklik); }
16   };
17
18 class Ucgen: public CokGen {
19   public:
20     int Alan ()
21       { return (genislik * yukseklik / 2); }
22   };
23
24 int main () {
25   Dortgen BenimGuzelDortgenim;
26   Ucgen   BenimGuzelUcgenim;
27
28   BenimGuzelDortgenim.DegerAyarla (4,5);
29   BenimGuzelUcgenim.DegerAyarla (4,5);
30
31   cout << BenimGuzelDortgenim.Alan() << endl;
32   cout << BenimGuzelUcgenim.Alan() << endl;
33
34   return 0;
35 }
36
```

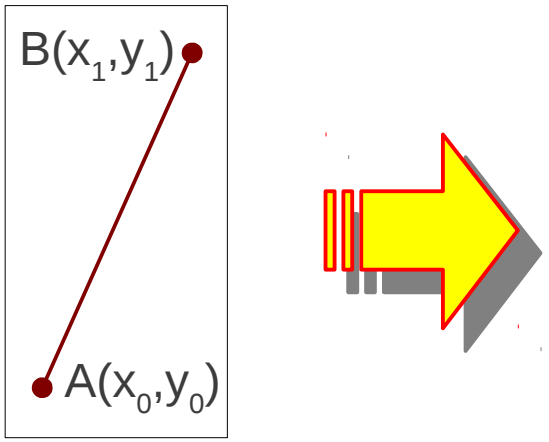
Ana class

Türemiş Class'lar

Örnek #1

Çizgi class' ı (header)

- Bir çizgi farklı şekillerde ***tanımlanabilir:***
  - İki nokta
  - Nokta, yön, uzunluk
  - Vektör
  - v.b.
- Yandaki örnekte ***iki nokta*** kullanılmıştır



Class **cizgi**

File Edit Search Preferences Shell Macro Windows Help

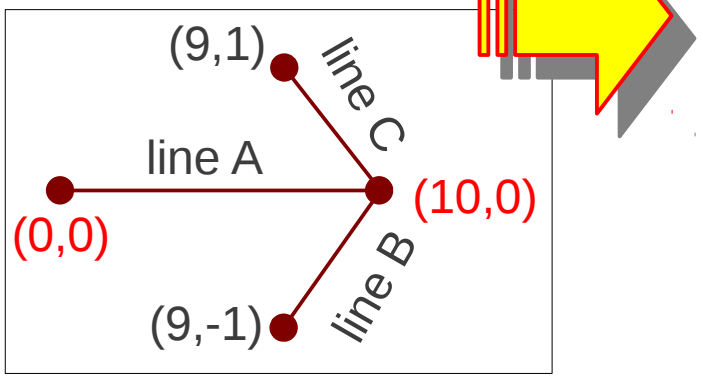
ok.cxx | ok.h | **cizgi.h** | cizgi.cxx | anaProgram.c | cikti

```
 1 #ifndef CIZGI_H
 2 #define CIZGI_H
 3
 4 #include <math.h>
 5 #include <iostream>
 6
 7 //_____________________________________________________________
 8 class cizgi
 9 {
10     private:
11         float x0, y0;      // ilk nokta
12         float x1, y1;      // ikinci nokta
13
14     public:
15         void degerAta(float a,
16                       float b,
17                       float c,
18                       float d);   // noktalari gir
19         virtual void bas();       // noktalari goster
20         virtual float uzunluk();  // uzunlugunu dondurur
21 };
22
23 #endif
24
```

Örnek #1

Ok class' ı (header)

- Bir ***ok***, ***üç çizgi*** ile oluşturulabilir
  - Bir ***ana çizgi*** (this)
  - ***Okun ucu***nu oluşturan ***2*** çizgi daha
- Okun ucunu oluşturan iki çizginin, ana çizgiden ne kadar ***uzaklaşacağı*** ve boyunun yine ana çizginin ***kaçta kaçı*** olacağı ***okGenis*** değişkeni (private variable) ile belirlenecek



Class ***ok***

File Edit Search Preferences Shell Macro Windows Help

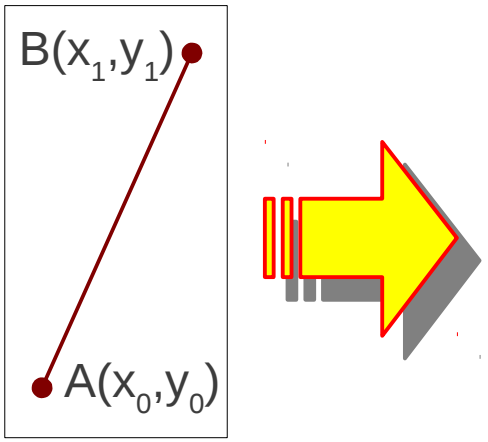
ok.cxx | **ok.h** | cizgi.h | cizgi.cxx | anaProgram.c | cikti

```
 1 #ifndef OK_H
 2 #define OK_H
 3
 4 #include "cizgi.h"
 5
 6 //_____________________________________________________________
 7 class ok : public cizgi
 8 {
 9     private:
10         cizgi A;            // Okun ana cizgisi
11         cizgi B, C;         // Ana cizginin iki yanindakiler
12         float okGenis;      // okun genisligi
13
14     public:
15         void degerAta(float a,
16                       float b,
17                       float c,
18                       float d);   // noktalari gir
19         virtual void  bas();      // noktalari goster
20         virtual float uzunluk();  // toplam uzunluk
21 };
22
23 #endif
24
```

Örnek #1

Cizgi class' ının uygulaması (implementation)

- Tüm **üye işlevler** (*member function* ya da *method*) class' ın **private değişkenlerine** erişim hakkına sahiptir:
  - degerAta()
  - uzunluk()
  - bas()



Class **cizgi**

File Edit Search Preferences Shell Macro Windows Help

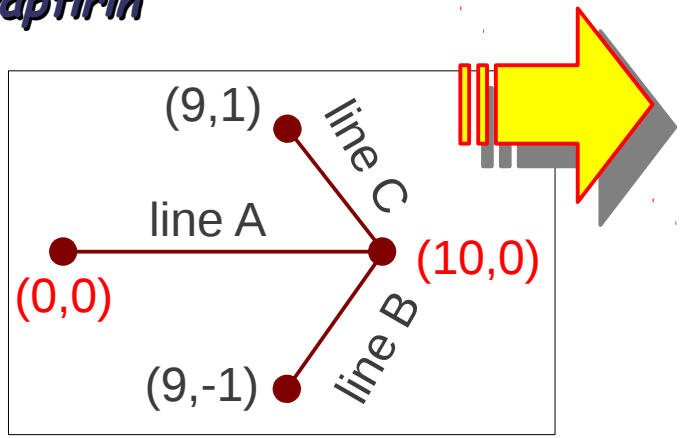
ok.cxx | ok.h | cizgi.h | **cizgi.cxx** | anaProgram.c | cikti

```
 1 #include "cizgi.h"
 2
 3 //_____________________________________________________________
 4 void cizgi::degerAta(float a, float b, float c, float d)
 5 {
 6 // Duzlemde tanimlanan iki noktanin koordinatlari
 7 // sadece class'a ait islevlerin (method ya da
 8 // member function) erisim hakki bulunan
 9 // degiskenlere ataniyor.
10
11     if (a==c && b==d) {
12         printf("HATA: Bu bir nokta ! \n");
13     } else {
14         x0 = a;                    // 1. noktanin x ordinati
15         y0 = b;                    // 1. noktanin y ordinati
16         x1 = c;                    // 2. noktanin x ordinati
17         y1 = d;                    // 2. noktanin y ordinati
18     }
19 }
20
21 //_____________________________________________________________
22 float cizgi::uzunluk()
23 {
24 // Cizginin uzunlugunu dondurur.
25
26     return sqrt( (y1-y0)*(y1-y0) + (x1-x0)*(x1-x0) );
27 }
28
29 //_____________________________________________________________
30 void cizgi::bas()
31 {
32 // Cizgiyi olusturan noktalarin koordinatlarini dondurur
33
34     printf("1. Nokta = (%f, %f) \n", x0, y0);
35     printf("2. Nokta = (%f, %f) \n", x1, y1);
36 }
37
```

Örnek #1

Ok class' ının uygulaması (implementation)

- Tekerleği yeniden icat etmeyin; varolanları kullanın:
  - ***Sıfırdan*** bir *ok* ***yapmak yerine*** varolan ***çizgileri kullanın*** (***degerAta()*** işlevi)
  - Her bir noktayı tek tek ***hesaplamak yerine***, oku oluşturan çizgilerin her birine hangi noktalardan oluştuklarını ***sorun*** (***bas()*** işlevi)
  - Tek tek her bir çizginin uzunluğunu ***hesaplamak yerine*** bu işi, oku oluşturan çizgilere ***yaptırın***



Class ***ok***

File Edit Search Preferences Shell Macro Windows Help

ok.cxx | ok.h | cizgi.h | cizgi.cxx | anaProgram.c | cikti

```
 1 #include "ok.h"
 2
 3 //_____________________________________________________________
 4 void ok::degerAta(float a, float b, float c, float d)
 5 {
 6 // Duzlemde tanimlanan iki nokta
 7 // arasinda bir ok cizer.
 8
 9         A.degerAta( a,          b,          c, d );
10         okGenis = A.uzunluk() / 10.0;
11         B.degerAta( c-okGenis, d-okGenis, c, d );
12         C.degerAta( c-okGenis, d+okGenis, c, d );
13 }
14
15 //_____________________________________________________________
16 void ok::bas()
17 {
18 // Oku olusturan noktalarin koordinatlarini dondurur
19
20     printf("Ana cizgi:\n");
21     A.bas();
22     printf("Ust cizgi:\n");
23     B.bas();
24     printf("Alt cizgi:\n");
25     C.bas();
26 }
27
28 //_____________________________________________________________
29 float ok::uzunluk()
30 {
31 // Cizginin uzunlugunu dondurur.
32
33     return (  A.uzunluk() +
34               B.uzunluk() +
35               C.uzunluk()    );
36 }
37
```

Örnek #1

Ok class' ı – Kullanıcı uygulaması

- Class' ların **başlık** (header, *.h) ve **tariflerini** (implementation, *.cxx) yazdıktan sonra bu class **kütüphanesini** (library) **kendi** programımızda kullanacağız:
  - anaProgram.cxx

- (0,5) ve (10,5) noktaları arasında bir **ok çizmek**, oku oluşturan **noktaları okumak** ve oku oluşturan **çizgilerin toplam uzunluğunu** hesaplamak istiyoruz.

- **BenimGuzelOkum** nesnesinin üye işlevlerine (member function) nokta (".") işlemcisi (ya da operatörü) ile ulaşılıyor

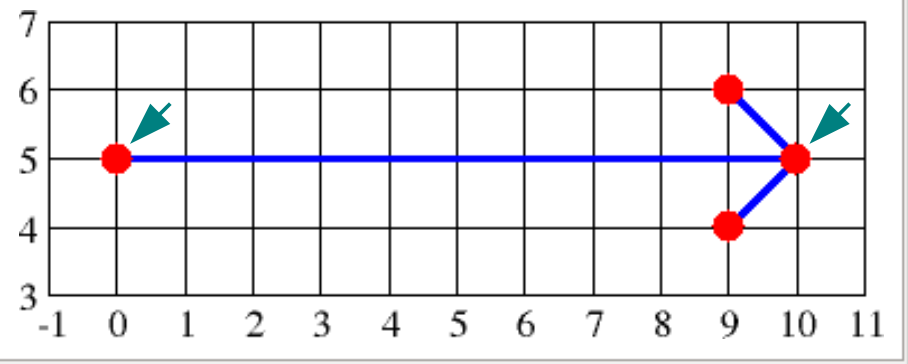
- BenimGuzelOkum nesnesi, **yığın** bellekte da (stack) oluşturuluyor

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy

anaProgram.cxx | anaProgram.cxx | cizgi.cxx

```
1 #include "ok.h"
2
3 //_____________________________________________
4 int main () {
5
6     // Ok nesnesi "BenimGuzelOkum"u yarat
7     ok BenimGuzelOkum;
8
9     // Okun hangi iki nokta arasinda olacagi
10    // Uye islevlere (member function ya da method)
11    // nokta (".") isareti ile ulasiliyor:
12    // nesne.islev(imza) biçiminde.
13    BenimGuzelOkum.degerAta(0.0, 5.0, 10.0, 5.0);
14
15    //Oku olusturan noktalari goster
16    BenimGuzelOkum.bas();
17
18    // Oku olusturan cizgilerin toplam uzunlugunu goster
19    printf("Uzunluk = %f \n", BenimGuzelOkum.uzunluk());
20
21    // Programdan basari ile cik
22    return 0;
23 }
24
```



Ln 12, Col 36 INS

Örnek #1

Ok class' ı – İşaretçi ile

- ***Aynı*** kütüphane (*ok* class'ı) ve ***aynı*** kullanıcı isteği
- Bu kullanımda "*ok*" ***nesnesi***, ***"new"*** işlemcisi (ya da operatörü) ile bir ***işaretçi*** (pointer) olarak yaratılıyor (14. satır)
- "*BenimGüzelOkum*" artık bir ***işaretçi*** olduğu için, ***üye işlevlerine*** (member function) ulaşmak için nokta (***"."***) ***yerine*** *işaretçi işlemcisi* (***"->"***) ***kullanılıyor*** (19., 22. ve 25. satırlar)
- Kullanımdan sonra nesne ***siliniyor*** ve işgal ettiği ***hafıza özgür*** bırakılıyor (30. satır); bu işlemin gerçekleşmemesi durumunda **hafıza kaçağı** (memory leak) meydana gelir.
- BenimGüzelOkum nesnesi **öbek**' bellekte (heap) oluşturuluyor

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy

anaProgram.cxx | cizgi.cxx | cizgi.h

```
1 #include "ok.h"
2
3 //_____________________________________________________
4 int main () {
5
6     // Bir "ok" nesnesine isaret eden ve ismi
7     // "BenimGuzeOkum" olan bir isaretci (pointer)
8     // yarat. Bu isaretcinin isaret ettigi hafiza
9     // bolgesi, ok class'inda tanimlanan degiskenlerin
10    // toplam buyuklugu kadardir ve bu degisken icin
11    // ozel olarak ayrilmistir; kullanimi sona erdikten
12    // sonra tekrar "serbest" birakilmalidir ya da
13    // yaratilan nesne "yok edilmelidir".
14    ok *BenimGuzelOkum = new ok;
15
16    // Okun hangi iki nokta arasinda olacagi
17    // Nokta (".") yerine isaretci ("->") islemcisi
18    // kullanildigina dikkat edin !!!
19    BenimGuzelOkum->degerAta(0.0, 5.0, 10.0, 5.0);
20
21    // Oku olusturan noktalari goster
22    BenimGuzelOkum->bas();
23
24    // Oku olusturan cizgilerin toplam uzunlugunu goster
25    printf("Uzunluk = %f \n", BenimGuzelOkum->uzunluk());
26
27    // Nesne, kullanimi sonunda yok ediliyor; kullandigi
28    // fiziksel hafiza serbest birakiliyor ya da isletim
29    // sisteminin kullanimina geri donuyor.
30    delete BenimGuzelOkum;
31
32    // Programdan basari ile cik
33    return 0;
34 }
35
```

Ln 35, Col 1 INS

Örnek #1

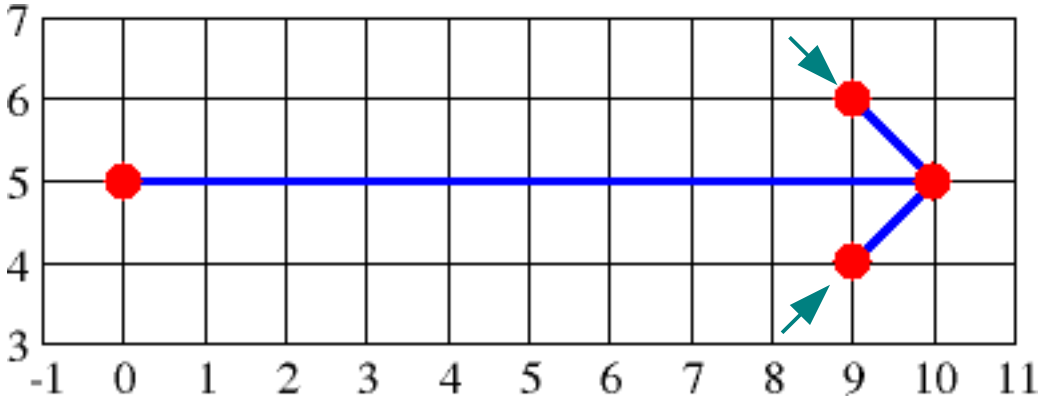
Ok - Çıktı

- ***Kullanıcı programını*** ve iki class' tan oluşan ***kütüphaneyi***:

  g++ anaProgram.cxx cizgi.cxx ok.cxx -o anaProgram

  ile ***derleyip çalıştır***dığımızda...

```
oc@olmak2:~/Documents/HEP_Okulu/workDir/ok$ ./anaProgram
Ana cizgi:
1. Nokta = (0.000000, 5.000000)
2. Nokta = (10.000000, 5.000000)
Ust cizgi:
1. Nokta = (9.000000, 4.000000)
2. Nokta = (10.000000, 5.000000)
Alt cizgi:
1. Nokta = (9.000000, 6.000000)
2. Nokta = (10.000000, 5.000000)
Uzunluk = 12.828426
oc@olmak2:~/Documents/HEP_Okulu/workDir/ok$ _
```



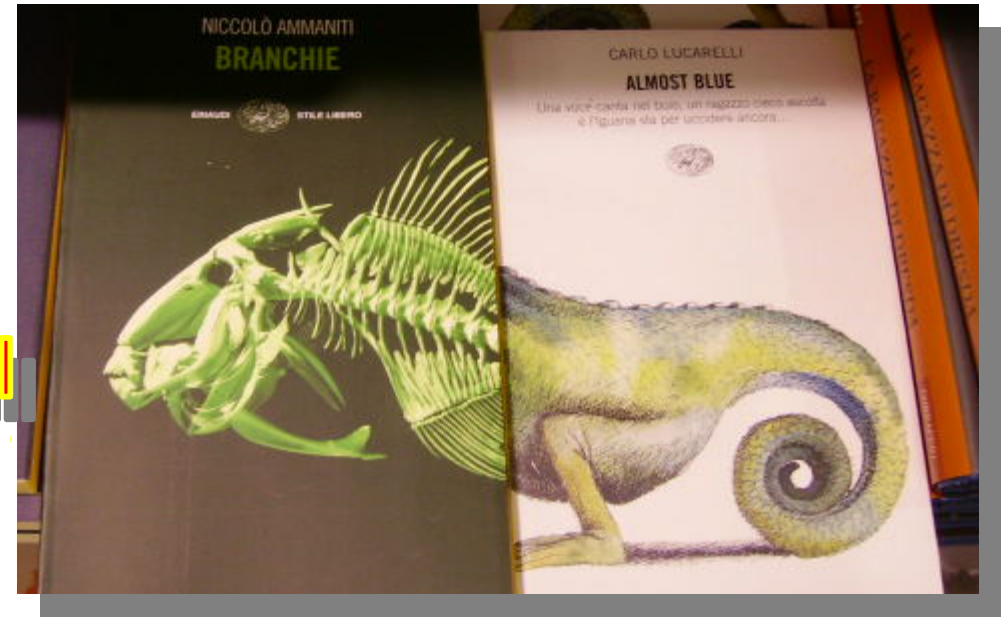
Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

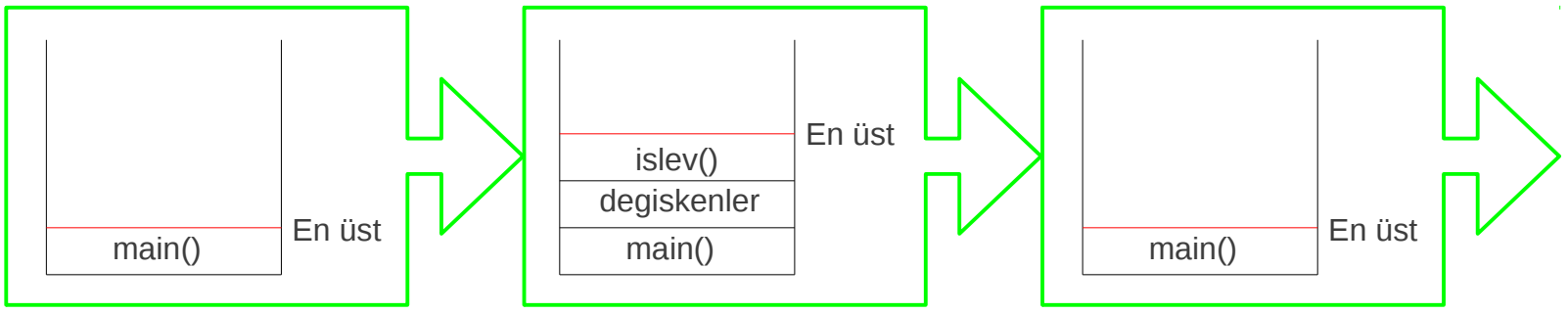
- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması



Kaynak (Code), Öbek (Heap) ve Yığın (Stack)

Yazılımların kullandığı hafıza bölgeleri

- Bir program hafızaya yüklendiğinde, **üç hafıza bölgesi** (segment) kullanılır: text ya da code (**kaynak**), heap (**öbek**) ve stack (**yığın**):
  - **Kaynak bölgesi** derlenmiş ve çalıştırılabilir olan programın makine ifadesinin işgal ettiği yerdir. Program çalıştığında atılacak adımların bir sıralaması niteliğindedir. Tüm kullanıcı ve sistem işlevlerinin bulunduğu alandır.
  - **Öbek** ve **yığın** ise derleyici tarafından veri saklamak için kullanılabilecek sistem hafızasıdır ve anlam bakımından farklı görünmeseler de aralarında işlev bakımından büyük farklar vardır:
    - **Yığın (stack)**, bir sıralı (procedural) programda tanımlanan sıradan değişkenlerin işgal ettiği ve programdan ya da o değişkenin tanımlandığı kapsamdan çıkıldığında **kendiliğinden salıverilen**, son-giren-ilk-çıkar (**LIFO**) yapısında bir bellektir. Denetimi derleyiciye bırakılmıştır.
    - **Öbek (heap)** ise veri saklamak için daha uygun olan, **kullanıcının denetiminde** ayrılan ve **el ile salıverilen** bir bellek alanıdır. Nesne yönelimli programlamada sıkça kullanılır. Öbek'te ayrılan bellek salıverilmediğinde, bellek kaçağı denen soruna yol açar.
- Bir nesneyi **heap'te** ya da **stack'ta** yaratmak ile bunların getirileri ve götürüleri, ileriki uygulamalarda daha açık hale gelecektir



Yığın çalışma yapısı

Kütüphane Derlemek ve Kullanmak

Çizgi ve Ok class' larını libCizgiVeOk.so kütüphanesine derlemek ve kullanmak

- Her class tek başına "**-c**" g++ anahtarı ile **derlenir** ve o class'a ait object (**nesne**) kütüğü oluşturulur (***.o**)
- Kütüphaneye (***.so** kütüğü) konmak istenen nesneler (***.o** kütükleri) "**-shared**" g++ anahtarı ile derlenir. Artık **libCizgiVeOk.so** kütüphanesi oluşmuştur.
- Herkesin kullanımına açmak için ***/usr/lib*** gibi **ortak erişime açık** bir yere yüklenir.
- Kullanıcı programlarında, "**-lCizgiVeOk**" anahtarı ile kullanılır.

```
> g++ -c -fPIC cizgi.cxx
> g++ -c -fPIC ok.cxx
> g++ -shared ok.o cizgi.o -o libCizgiVeOk.so
> sudo cp libCizgiVeOk.so /usr/lib/
> g++ anaProgram.cxx -o anaProgram -lCizgiVeOk
> ./anaProgram
Ana cizgi:
1. Nokta = (0.000000, 5.000000)
2. Nokta = (10.000000, 5.000000)
Ust cizgi:
1. Nokta = (9.000000, 4.000000)
2. Nokta = (10.000000, 5.000000)
Alt cizgi:
1. Nokta = (9.000000, 6.000000)
2. Nokta = (10.000000, 5.000000)
Uzunluk = 12.828426
> _
```

- Kullanıcının kütüphaneyi kullanmak için class kaynaklarına (source code) ihtiyaç duymayacağına ***fakat*** kütüphanede hangi işlevlerin (method) varolduğunu bilmeye ihtiyaç duyacağına dikkat edin:
  - Belgelemeye ihtiyaç var

Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

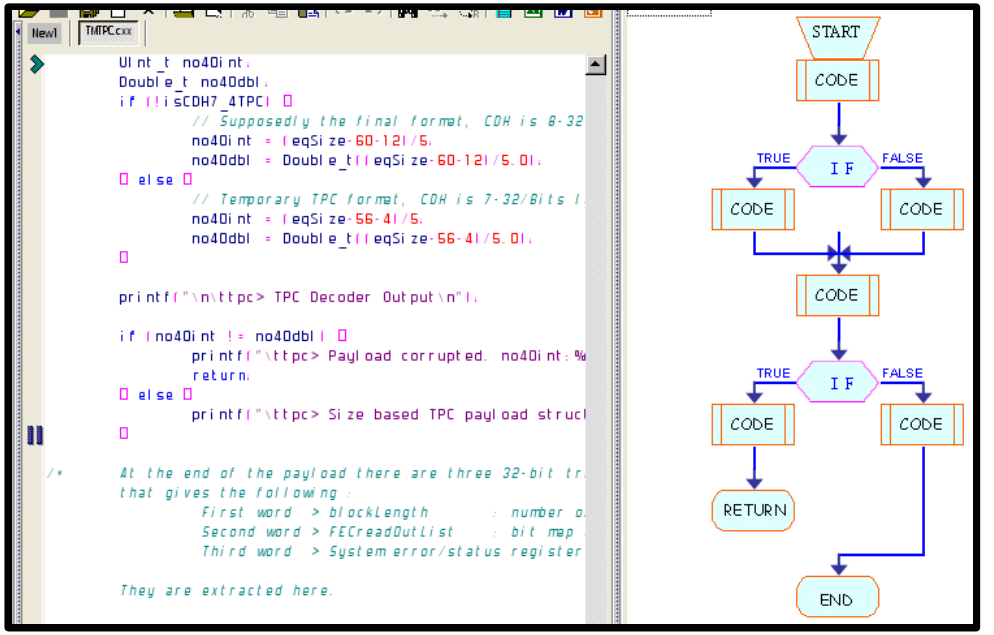
- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

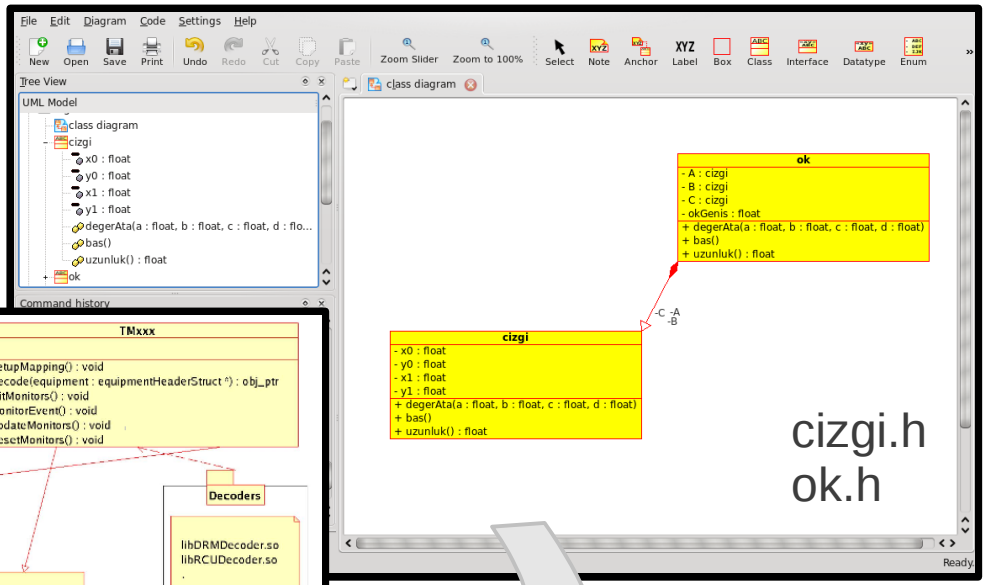
- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması

Düşünmek Derken...

Akış ve UML çizelgeleri

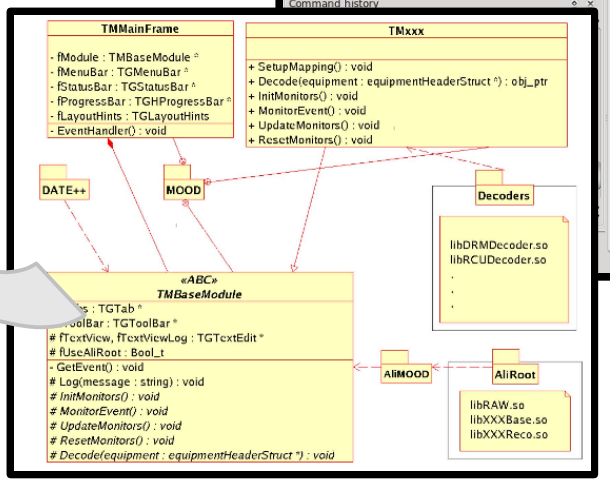
- Sıralı programlamada **akış** çizelgesi
- Nesne yönelimli programlamada **UML** (unified modelling language) diyagramı
  - **UML** içindeki class için **akış** çizelgesi
  - **UML 'den class hiyerarşisi** ve
  - **Class hiyerarşisi** ' nden **UML** üretmek mümkündür






```
class Dortgen: public Co
  public:
    int Alan ()
      { return (genislik
  };

class Ucgen: public CokG
  public:
```

UML' den üretilmiş kaynak.

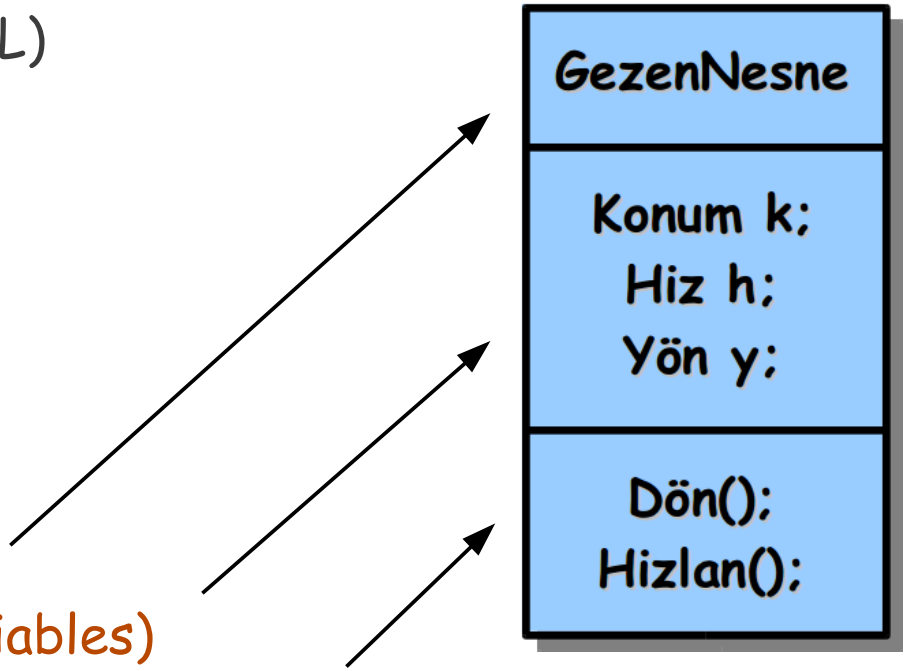


Daha Karışık

Örnek #2

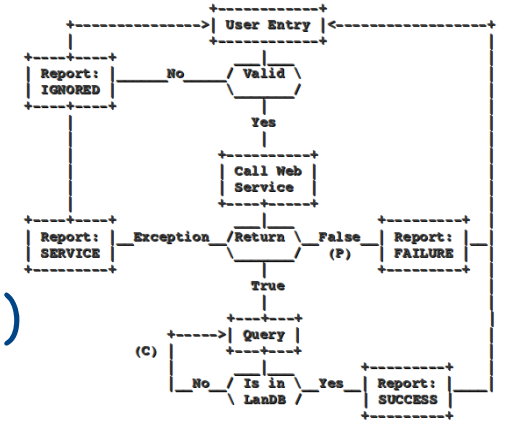
Yeni veri çeşidi yazmak ve UML hiyerarşisi Kurmak

- ***Syntax*** error'u bana ***derleyici*** söyleyebilir (***semantik*** hataları değil)
- ***Mimari*** hataları ***için*** bu ***geçerli değil***, dolayısı ile ~1000 satırı aşkın bir kodun içinde kaybolmamak için bir tasarım ortama ihtiyacım var:
  - Unified Modeling Language (UML)
  - Paradigması/fikri şu:
    - Nesneler olsun (dörtgen)
    - Herbir nesne tanımlansın:
      - ismi (class isim;)
      - ne bildiği (member variables)
      - ne yaptığı (methods, member functions)
    - Hedef, bu nesnelerin işlevsel bir hiyerarşisi olsun



UML Demek...

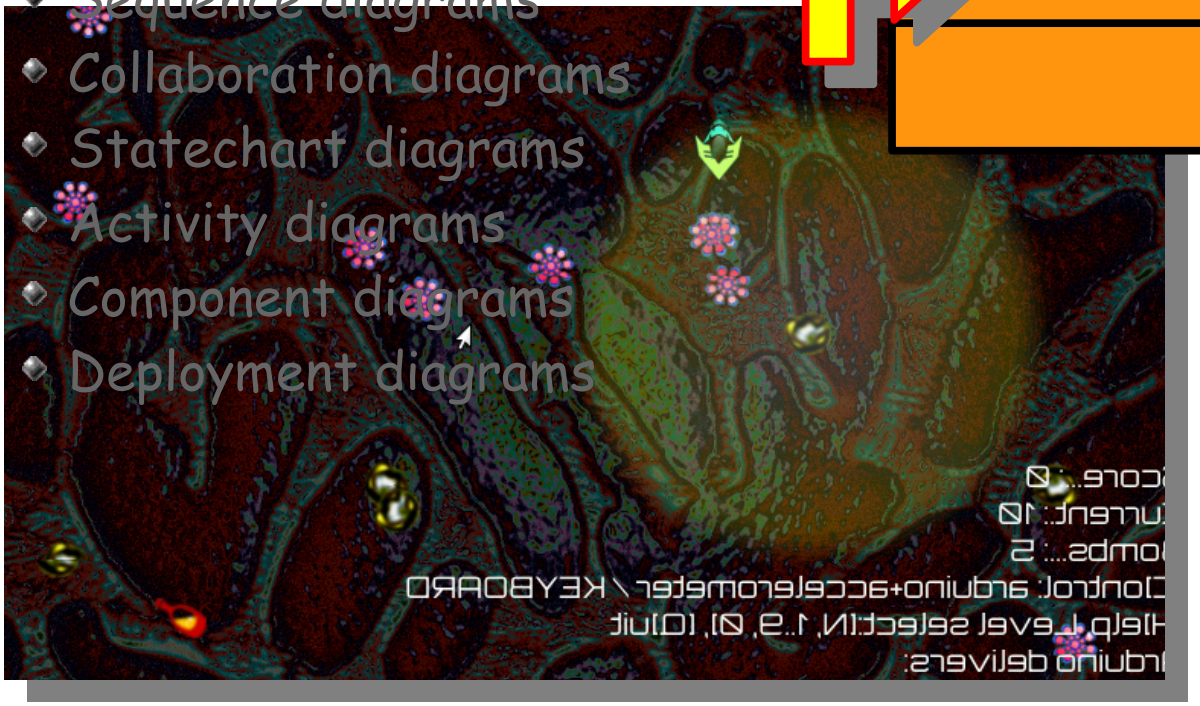
Farklı bakış açıları demektir

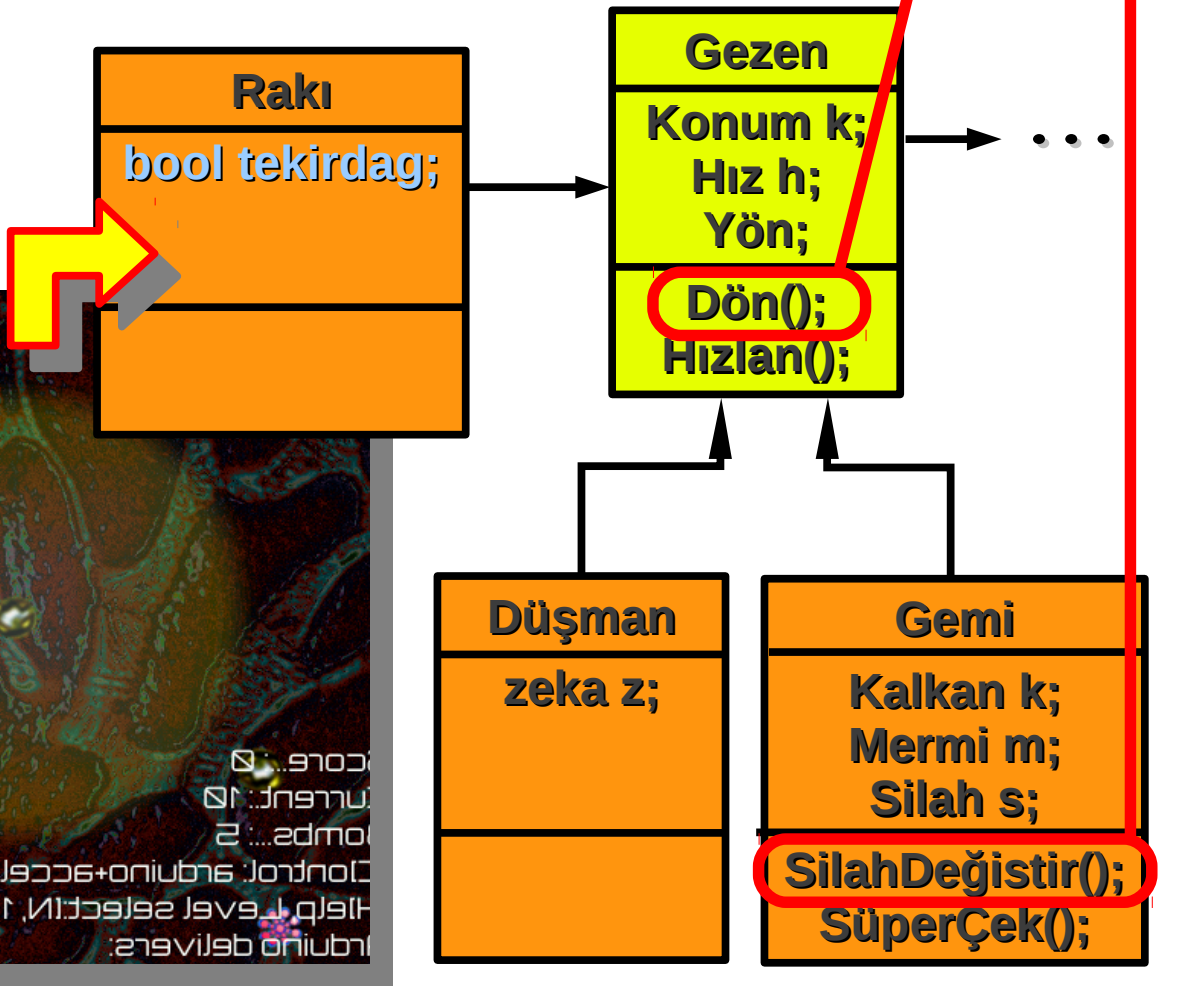


- **Akış** Çizelgesi → **Sıralı** programlama (ForTran, C v.b.)
- **UML** Diyagramı → **Nesne Yönelimli** (C++, java, python v.b.)

UML birkaç farklı algılama yolunu mümkün kılar:

- Use case diagrams
- *Class diagrams*
- Object diagrams
- Sequence diagrams
- Collaboration diagrams
- Statechart diagrams
- Activity diagrams
- Component diagrams
- Deployment diagrams





UML ile Tasarım Alıştırması

Sipariş hizmet sağlayıcı

- Bir ödeme yazılımı (amazon.com'unki gibi)
- Merkez nesnem sipariş
  + verildiği **tarihi** ve şu anki **durumunu** **biliyor**

  + kendisine **vergi** miktarı, toplam **fiyatı** ve toplam **ağırlığı** **sorulabiliyor**

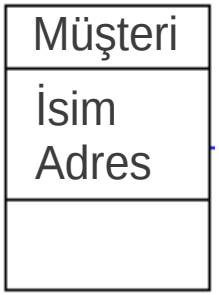
- Bu nesne **bunlar dışında** hiçbirşey **bilmiyor**
  + sınıf tanımı ve uygulaması **kısa** kalıyor, bu iyi bir şey

- Ne sipariş edilmiş ?
  + **Beni ilgilendirmez** !.. (**Bİ**)

UML ile Tasarım Alıştırması

Sipariş hizmet sağlayıcı

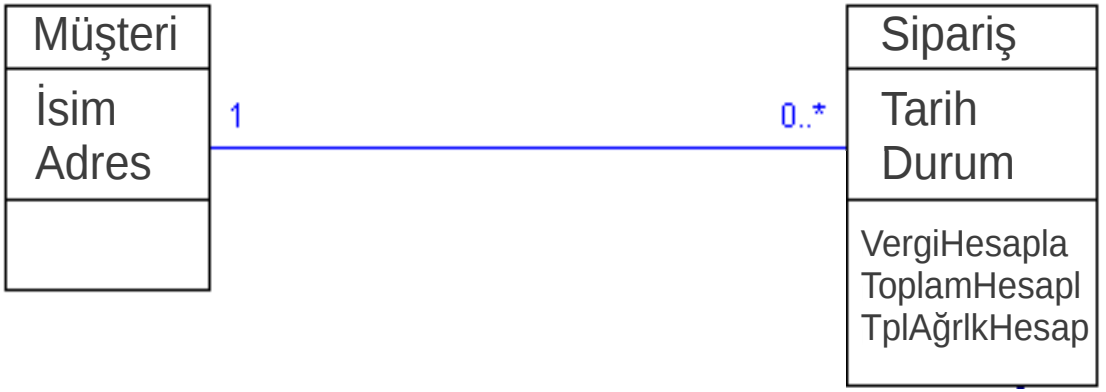
- Ortada bir sipariş varsa bu, bir **alıcıyı** gerekli kılar
- Alıcı kişi/müşteri bir **isme** ve **adrese** sahip



UML ile Tasarım Alıştırması

Sipariş hizmet sağlayıcı

- Ortada bir sipariş varsa bu, bir **alıcıyı** gerekli kılar
- Alıcı kişi/müşteri bir **isme** ve **adrese** **sahip**

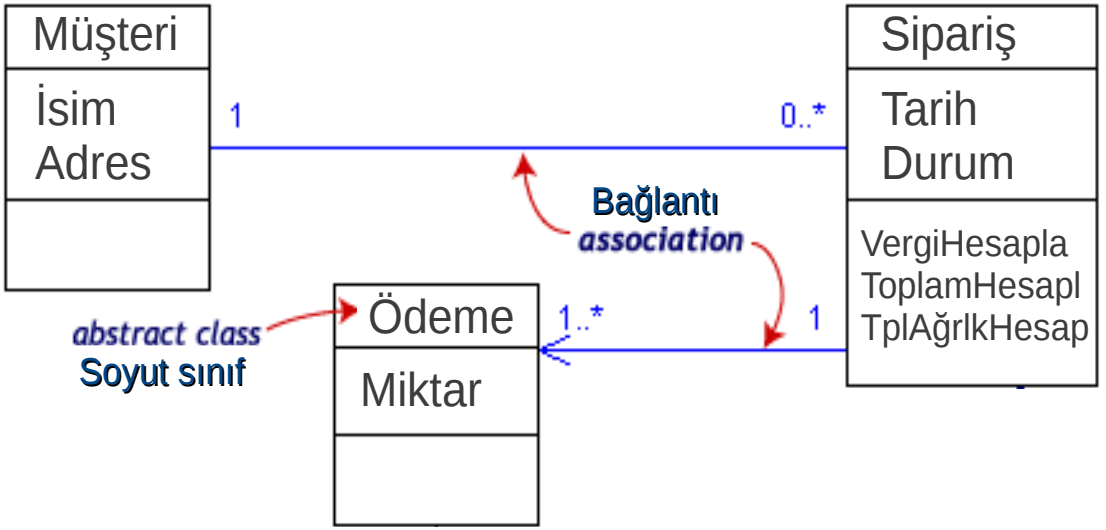


- Aralarındaki **ilişki** düz çizgi (çifttaraflı)
  - + Sipariş sınıf'ı Müşteri cinsinden bir **değişkene** sahip demek
- Sayılar 1 ve "0..*" sırayla şu demek:
  - + (1) Bir **sipariş** sadece **bir alıcıya** sahip olabilir
  - + (0..*) Bir **alıcı** **herhangi bir sayıda** sipariş verebilir

UML ile Tasarım Alıştırması

Sipariş hizmet sağlayıcı

- Her **sipariş'in** bir **bedeli** vardır (Ödeme sınıfı)
- Sipariş'in ne **kadar olduğunu** **sorabilirim** **ama** ödemenin **hangi sipariş için** olduğunu **soramam** (bağlantı tek yönlü)

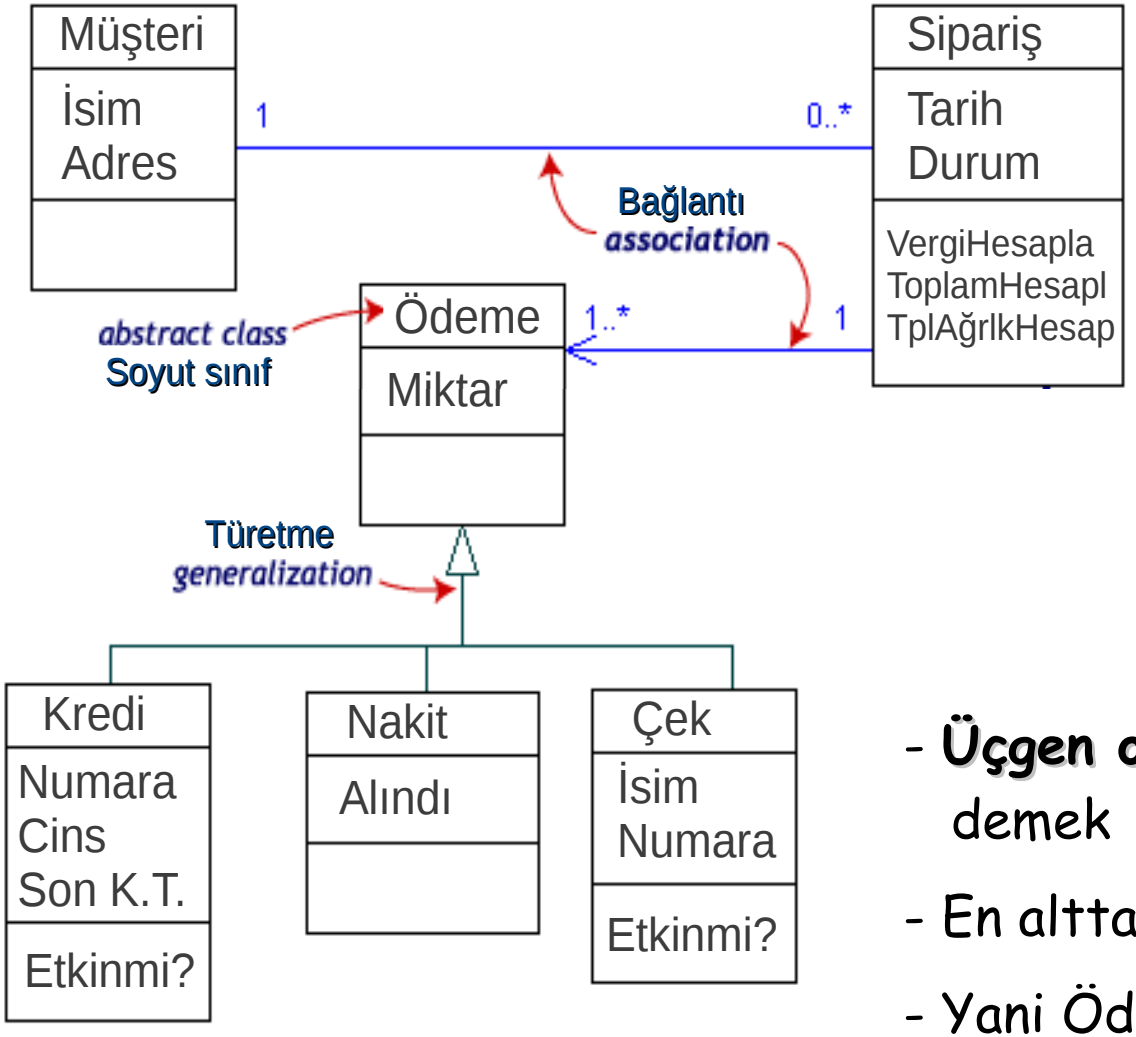


- Her sipariş için **en az bir ödeme** olmalı
- Ödeme nesnesi **sadece** miktarı biliyor
- Kartla mı, peşin mi ?
  - + **Bİ** → program kısa, kolay

UML ile Tasarım Alıştırması

Sipariş hizmet sağlayıcı

- **Kredi**, **Nakit**, **Çek** sınıf'ları **Ödeme'den** türedi
- Her biri **sadece** ilgili ödeme değişkenlerini biliyor ve bunlarla ilgili şeyleri yapabiliyor (Gerisi **BİR**)



- Bir yandan bunu nasıl yazabileceğinizi düşünün
- Üçgen, dörtgen örneğini hatırlayın

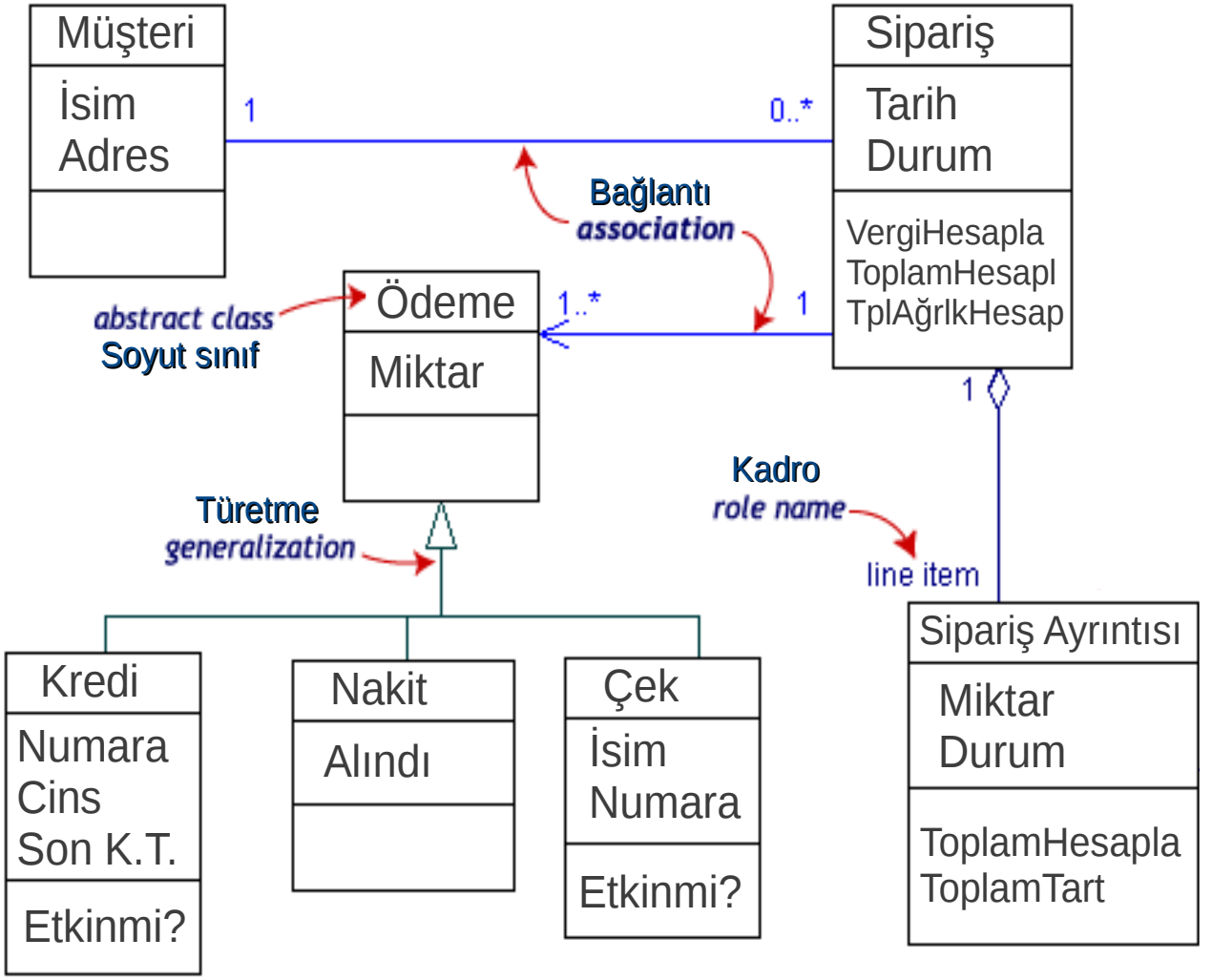
- **Üçgen ok** türemiş olmak (inheritance) demek
- En alttakiler, Ödeme'den türetilmişler
- Yani Ödeme bir ana sınıf (base class)

UML ile Tasarım Alıştırması

Sipariş hizmet sağlayıcı

- Her **bir sipariş** **en az bir sipariş ayrıntısına** sahip
- **Baklava ok** kolleksiyon demek, örneğin **array**, **vector** v.b.
- **Bağlantı** **tek yönlü**, sipariş nesnesi ayrıntısı ile ilgili cevap veriyor **ama**...

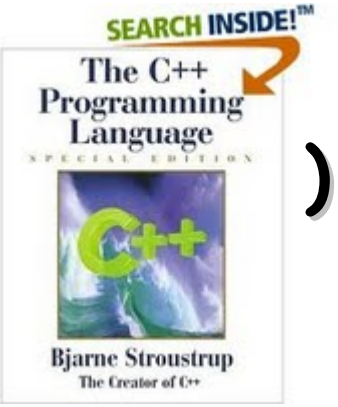
...tersi anlamlı değil (**Bİ**)



- Kac tane ?
  + (bil)
- Kaç lira ?
  + (hesapla)
- Ne kadar ağır ?
  + (hesapla)

UML ile Tasarım Alıştırması

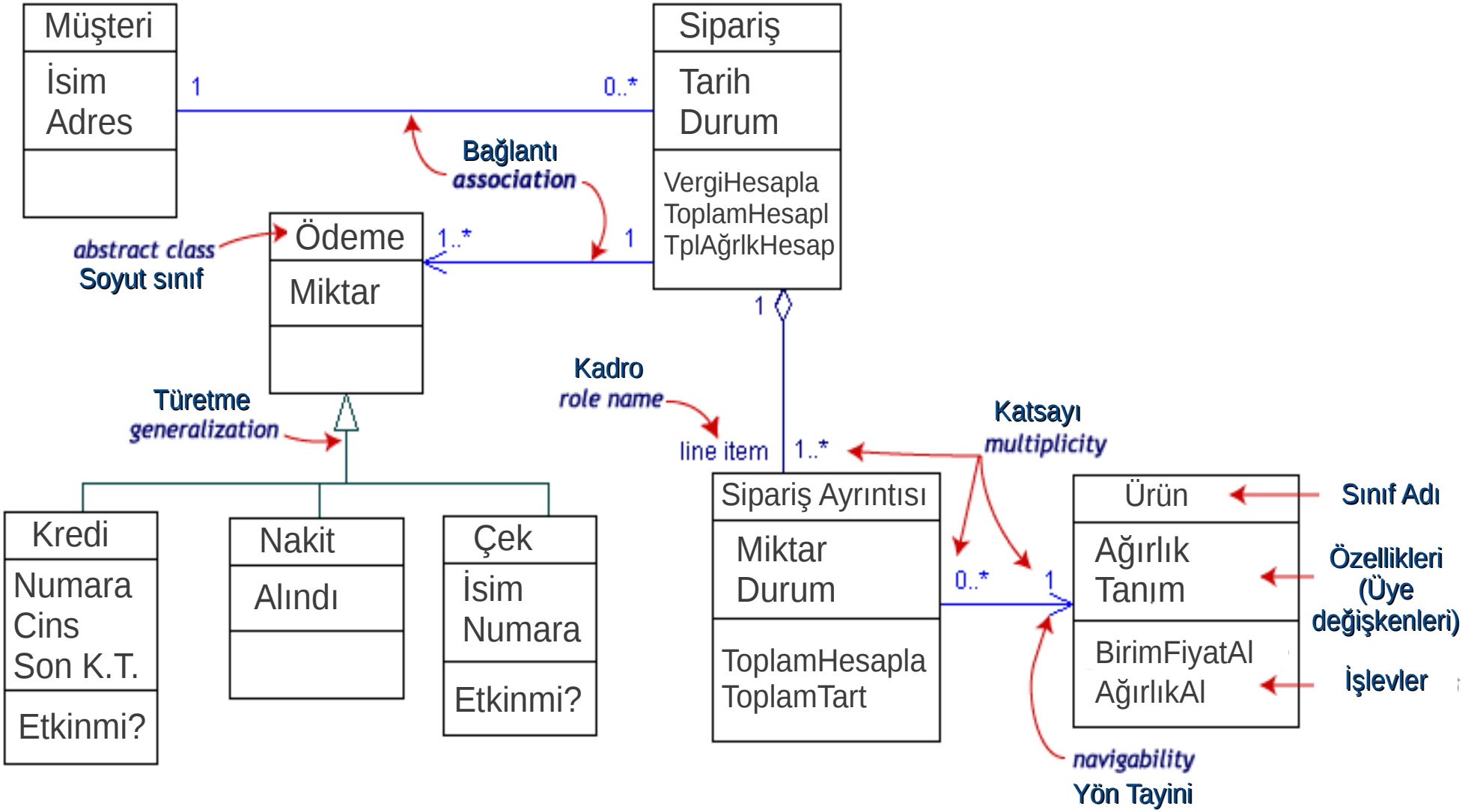
Sipariş hizmet sağlayıcı



-Asıl satın alınabilen **en küçük şey** ise Ürün sınıfı cinsinden **(ör:)**

-**Ağırlığını** ve **tanımı** biliyor

-Ağırlığını ve fiyatını veri tabanından **sorgulayabiliyor**



UML ile Tasarım Alıştırması

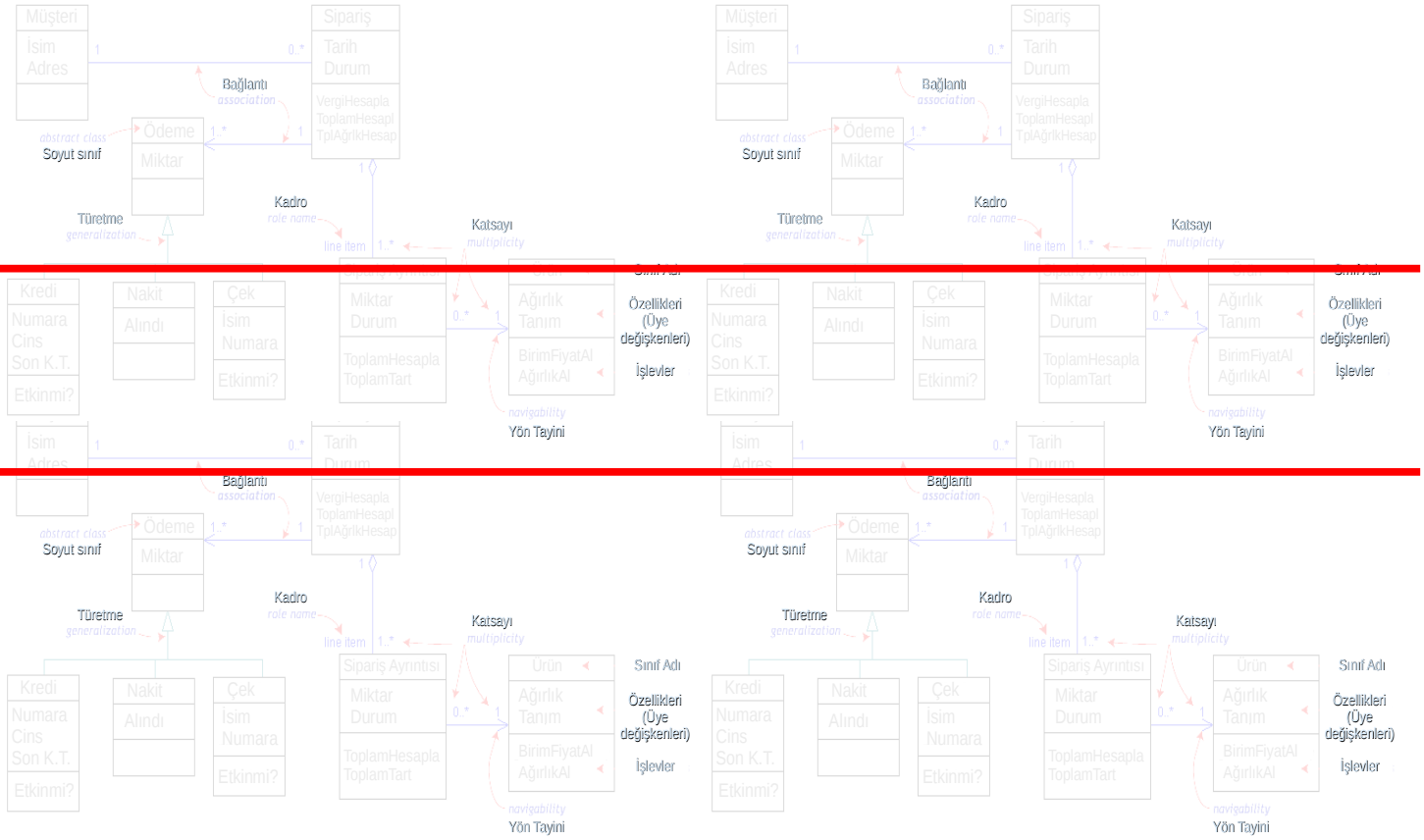
Sipariş hizmet sağlayıcı

- Her yazılmın temelde **3 düzeyi** var:
  - + **Display** (kullanıcı ile etkileşim)
  - + **Processing** (veri işlemek)
  - + **Data** (veriye ulaşmak: kütük/veri tabanı okumak/yazmak)
- **Hangi sınıf** 'ın **hangi düzeyde** olduğu da düşünülmeli

Display

Processing

Data



UML ile Tasarım Alıştırması

C'de olası bir alternatif ?

```
typedef struct siparis {
    Tarih d;
    Durum s;
    İsim n;
    Adres a;
    ...
    ...
    ...
    ...
    ...
    ...
    double ToplamHesapla();
    double AğırlıkHesapla();
};
```

Tanımlanmış yeni veri tipi

Kullanım:

```
siparis o;
Double a=o.ToplamHesapla();
```

Sipariş ile ilgili değişkenler
(üye değişkenler yerine)

Sipariş ile ilgili işlevler
(üye işlevler yerine)

UML ile Tasarım Alıştırması

Seçime bağlı ödev

Kısım #1:

- Kendi yapabildiğiniz **her şeyin** algoritması yazılabilir mi ? Cevabınızı bir kaç cümle ile **savunun**.
- Bir önceki sayfadaki alternatifin sorunu nedir ?
  - mimari açıdan değerlendirin

Kısım #2:

- Bir oyun tanımlayın ve mimariini **kurun** (örnekteki oyunu kullanabilirsiniz)
- Mimari'ye yoğunlaşın uygulamaya değil ve yukarıda yaptığınız sınıf diyagramından ya **otomatik** kod üretin ya da **elle** yazın ki...
  - ...aklınızdaki **UML ↔ C++ karşılıklılığı** ile ilgili resim netleşsin

Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

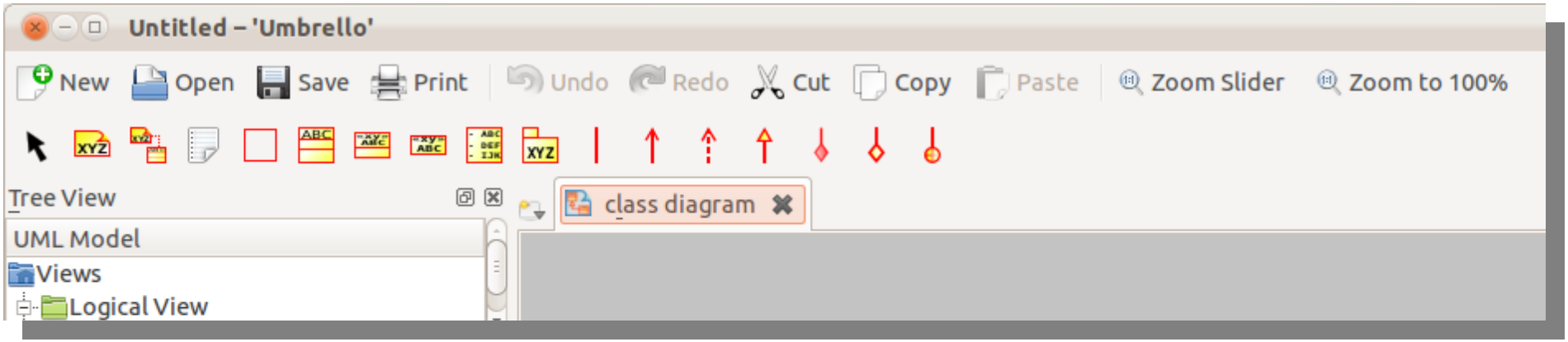
- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması

UML Tasarım Yazılımları

Kod'dan UML ve UML'den kod üreten araçlar

- Umlet, umbrello, gaphor, dia, bouml, uml-utilities, tcm, xfig, **v.b.**
- Chrome web browser'ı üzerinde çalışan **bile** var
- Bunlardan bazıları...
  - + ...sadece çizmek için (xfig gibi)
  - + ...sadece UML'den kod üretiyor
  - + ...sadece kod okuyup UML üretiyor
  - + ...iki yönlü UML<->Kod üretiyor
  - + ...komut satırı'nda bu iki işi yapıyor
- **Umbrello** hızlı bir arama ve denemede en iyisi(?) gibi görünüyor(?)
  - + Subjektif



Umbrello ile Bir Örnek

Kod'dan UML ve UML'den kod üreten araçlar

- **Sıyırduino** örneğini değerlendir
- Önce hareket eden 'şey'in **ast-üst yapısını** kur
- Sonra bunu, **oyun mimariine** koy (bu derste değil)

Umbrello ile Bir Örnek

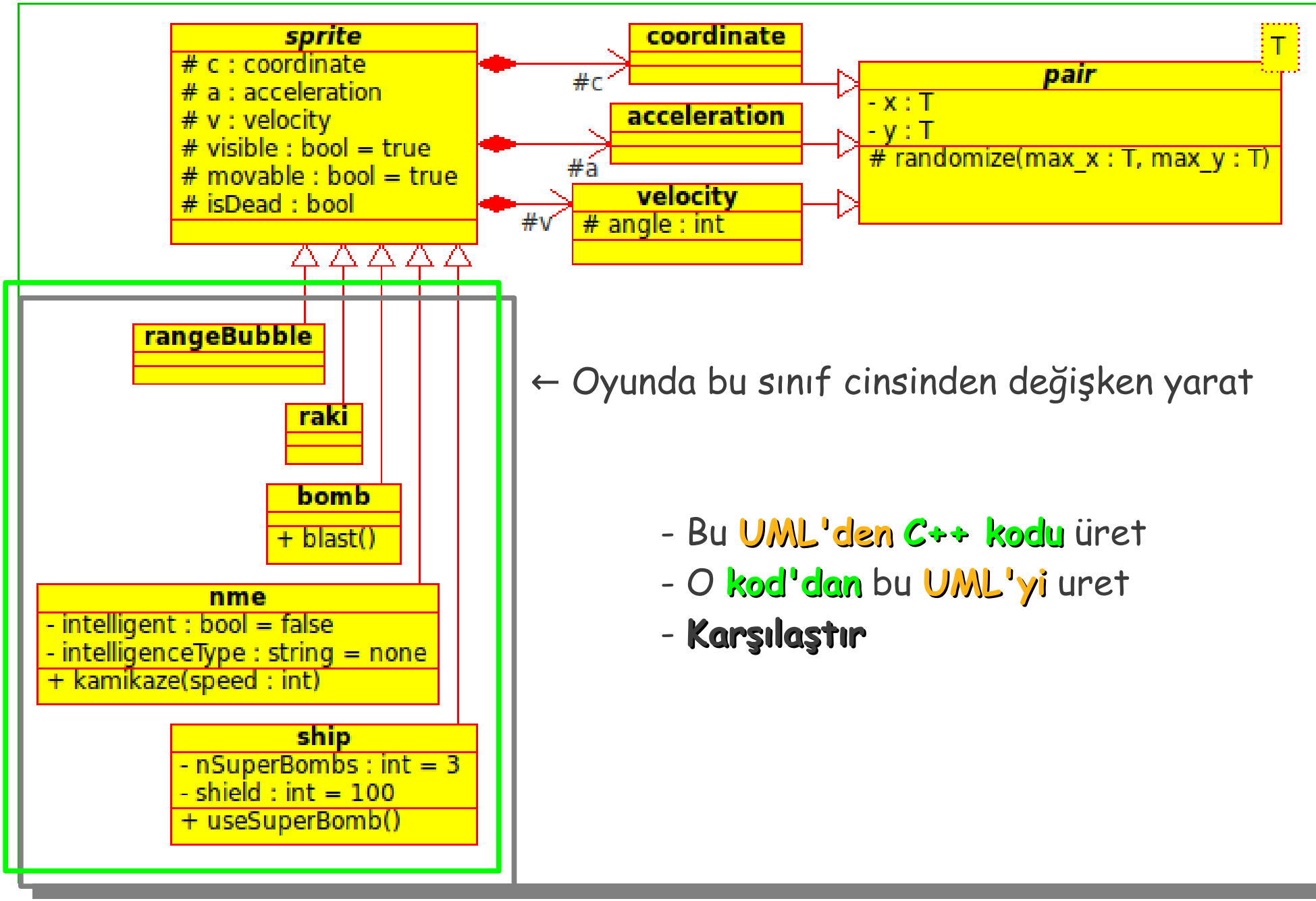
Kod'dan UML ve UML'den kod üreten araçlar

- **Sıyırduino** örneğini değerlendir
- Önce hareket eden 'şey'in **ast-üst yapısını** kur
- Sonra bunu, **oyun mimariine** koy (bu derste değil)

Sıyırduino filmi

Umbrello ile Bir Örnek

Kod'dan UML ve UML'den kod üreten araçlar



← Oyunda bu sınıf cinsinden değişken yarat

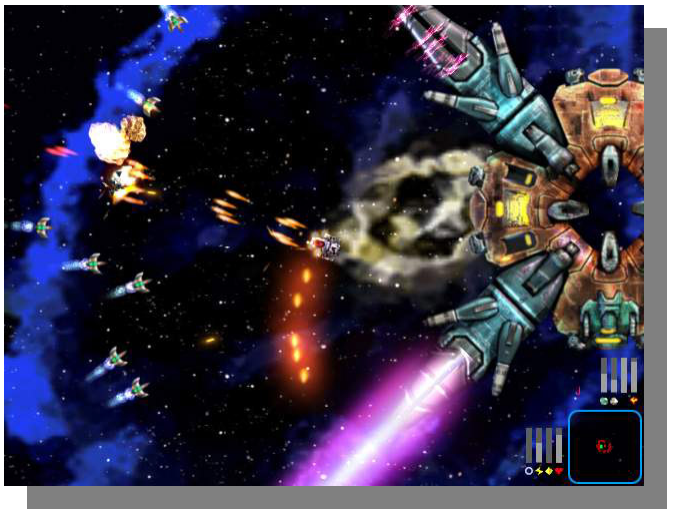
- Bu UML'den C++ kodu üret
- O kod'dan bu UML'yi uret
- **Karşılaştır**

Umbrello ile Bir Örnek

Kod'dan UML ve UML'den kod üreten araçlar

- **Yeni** kullanıcı **veri tipi** = **class** (ya da **struct**)
- Bu tipin değişkenlerini **işleme tabi tutma kuralları** = **operatörler**

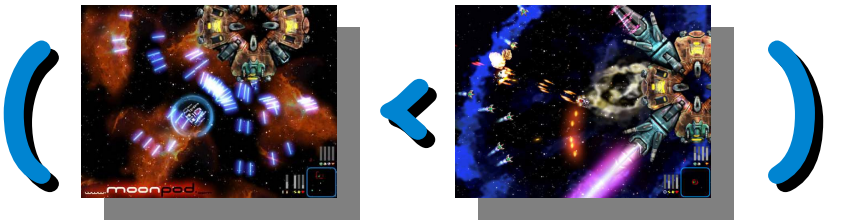
- **Aşina** olduğumuz veri tiplerinden örnek:
  - + int, double, v.b. → veri tipi
  - + topama, çıkarma, v.b. → operatörler
  - + A*B için operatör söyle tanımlı:
    - int **operator***(){return(A*B);}
  - + A<B için "<" operatörü:
    - bool **operator<**(){return(A<B);}



- class **bolumSonuCanavari**;
- class **muonTrack**;
- class **ucgen**;

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama



- Tanımla: **class** bolumSonuCanavari
- **Küçüktür (<)** işlemcisini tanımla böylece
  - + **bolumSonuCanavari** cinsinden olan değişkenleri karşılaştırabilirim
  - + variable, instance, object

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
1 #include<string>   // string icin
2 #include<cstdlib>  // rand icin
3 #include<iostream> // cout icin
4 using namespace std; // Tembelim: 'std::cout' yerine
5                      // kisaca 'cout' yeterli olsun diye
```

- Kullanacağın komutların tanımlı olduğu basşlıkları ekle

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- bolumSonuCanavari veri tipini tanımla
- eni, boyu, ağırlığı ve mermi sayısı var
- bunları kullanarak 'zorluk' belirle
- böylece '<' işlemi anlam kazansın

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
7 //_____________________________________
8 class bolumSonuCanavari {
9     private:
10        double en, boy, agirlik;
11        int mermi;
12    public:
13        // C'tor
14        bolumSonuCanavari();
15        // Karsilastirma operatorunu tanimliyorum
16        bool operator<(const bolumSonuCanavari);
17        void goster();
18 };
```

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Constructor (**C'tor**) rastgele sayılar atar
- 0 ile 100 arasında değer alır **yeni yaratılan her nesne**
  - variable, object, instance

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
20 //_____|_____________________________________
21 bolumSonuCanavari::bolumSonuCanavari() {
22       // C'tor initiates the instance (variable)
23
24     en = (rand()/(1.0 * RAND_MAX)) * 100.0;
25     boy = (rand()/(1.0 * RAND_MAX)) * 100.0;
26     agirlik = 1.0 + (rand()/(1.0 * RAND_MAX)) * 100.0;
27     mermi = (rand()/(1.0 * RAND_MAX)) * 100.0;
28 }
```

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- bolumSonuCanavari için **küçüktür operatörünü** tanımla
- üye işlevin adı '**operator<**'
- benzer şekilde sunlar da var: **operator>**, **operator/**, **operator***, **operator[]**, **operator+**, **operator-**, v.b.

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
30 //_____|_____________________________________________
31 bool bolumSonuCanavari::operator<(const bolumSonuCanavari bsc) {
32       // Kucuktur operatorunu tanimliyoruz
33
34     double ff0; // Ilk operand icin fitness function
35     double ff1; // Ikinci operand icin fitness function
36
37       // Keyfi karsilastirma kriterim
38     ff0 = (this->en * this->boy) / this->agirlik + this->mermi;
39     ff1 = (bsc.en  * bsc.boy)  / bsc.agirlik  + bsc.mermi;
40
41       // Buyuk mu degil mi
42     return (ff0<ff1);
43 }
```

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Nesne kendisini **göstersin**
- Böylece **sağlamasını** yapabilelim

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
45 //_____________________________________
46 void bolumSonuCanavari::goster() {
47       // Display private members
48     cout<<"En:"<<en<<" - Boy:"<<boy
49         <<" - Agirlik:"<<agirlik
50         <<" - Mermi:"<<mermi
51         <<endl; // C'deki "\n" anlaminda, end line
52                 // ya da bir <cr> yaz (carriage return)
53 }
```

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Kütüphanemiz tamamlanmıştır
- Şimdi iki canavar yaratalım ve karşılaştıralım

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
55 //_____________________|_____________________
56 int main() {
57       // Iki bolum sonu canavari yarat, karsilastir
58       // ve ilk olarak guclu olani goster.
59
60     bolumSonuCanavari canavar0, canavar1; // Create instance
61
62       // Bu karmasik karsilastirma islemini biz tanimladik
63     if (canavar0<canavar1) {
64         canavar1.goster();
65         canavar0.goster();
66     } else {
67         canavar0.goster();
68         canavar1.goster();
69     }
70
71     exit(0); // Kibarca cik programdan
72 }
```

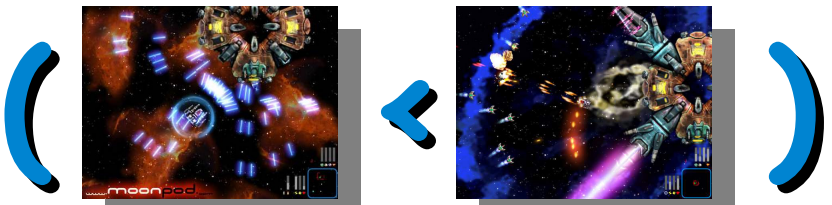
Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Derleyip çalıştırdığımızda:
  - + **ilkinin** zorluk seviyesi **128.168243859** iken
  - + **ikincininki 120.779886749**

```
o@olmak-x200: ~/Documents/odtuYefCalismaGrubu
o@olmak-x200:~/Documents/odtuYefCalismaGrubu$ g++ myType.cxx -o myType
o@olmak-x200:~/Documents/odtuYefCalismaGrubu$ ./myType
En:91.1647 - Boy:19.7551 - Agirlik:34.5223 - Mermi:76
En:84.0188 - Boy:39.4383 - Agirlik:79.3099 - Mermi:79
o@olmak-x200:~/Documents/odtuYefCalismaGrubu$
```

- Yani **karşılaştırma operatörü** istediğim gibi çalıştı



bsc0 **bsc1**

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- **Tanımlamadığımız büyüktür operatörünü (>)** kullanmayı deneyelim

myType.cxx (160 GB Filesystem) - gedit

286.cxx | myType.cxx

```
55 //_____________________________________
56 int main() {
57       // Iki bolum sonu canavari yarat, karsilastir
58       // ve ilk olarak guclu olani goster.
59
60     bolumSonuCanavari canavar0, canavar1; // Create instance
61
62       // Bu karmasik karsilastirma islemini biz tanimladik
63     if (canavar0>canavar1) {
64         canavar1.goster();
65         canavar0.goster();
66     } else {
67         canavar0.goster();
68         canavar1.goster();
69     }
70
71     exit(0); // Kibarca cik programdan
72 }
```

Kendin Tanımla Kendin Kullan

Yeni veri tipi ve ilgili işlemcileri tanımlama

- **Derlemek mümkün değil** çünkü tanımlamadığımız bir işlev
- **Operator (>)** henüz **anlamlı değil**

```
o@olmak-x200: ~/Documents/odtuYefCalismaGrubu
o@olmak-x200:~/Documents/odtuYefCalismaGrubu$ g++ myType.cxx -o myType
myType.cxx: In function ‘int main()’:
myType.cxx:63:18: error: no match for ‘operator>’ in ‘canavar0 > canavar1’
o@olmak-x200:~/Documents/odtuYefCalismaGrubu$
```

return( < ); // OK

return( > ); // ~~OK~~

- **Ödev** olarak **diğer işlevleri** tanımlayabilirsiniz ya da kendiniz bir veri tipi yaratıp alıştırmayı tekrarlayabilirsiniz

Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

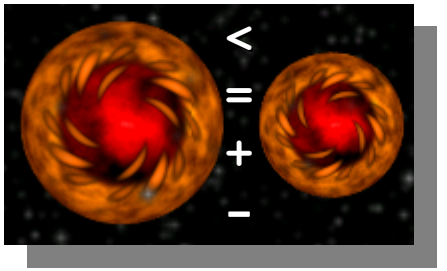
- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

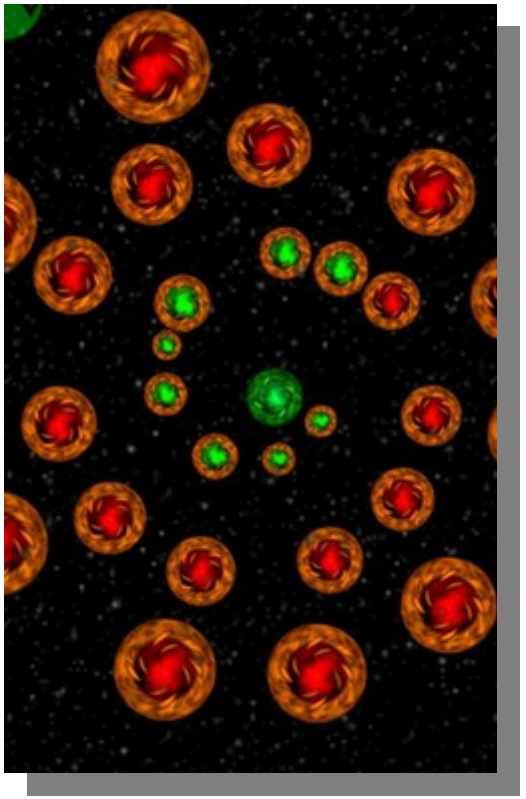
- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması

Geometrik Şekil Hiyerarşisi

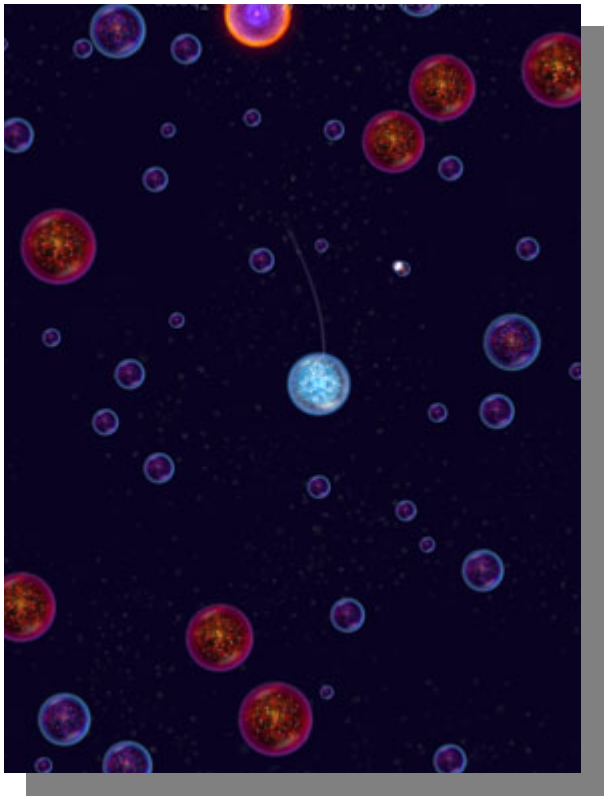
Yeni veri tipi ve ilgili işlemcileri tanımlama

- İçinde küreler olan oyun yazıyorum: ben de **küreyim**
- Kürelerin bazıları **büyük** bazıları **kücük**
- Kendimden büyüğe dokunursam soğuruluyorum: **küçül**
- Kendimden küçüğe dokunursam soğuruyorum: **büyü**
- İlerlemek için dışarıya bişeyler atıp küçülüyorum





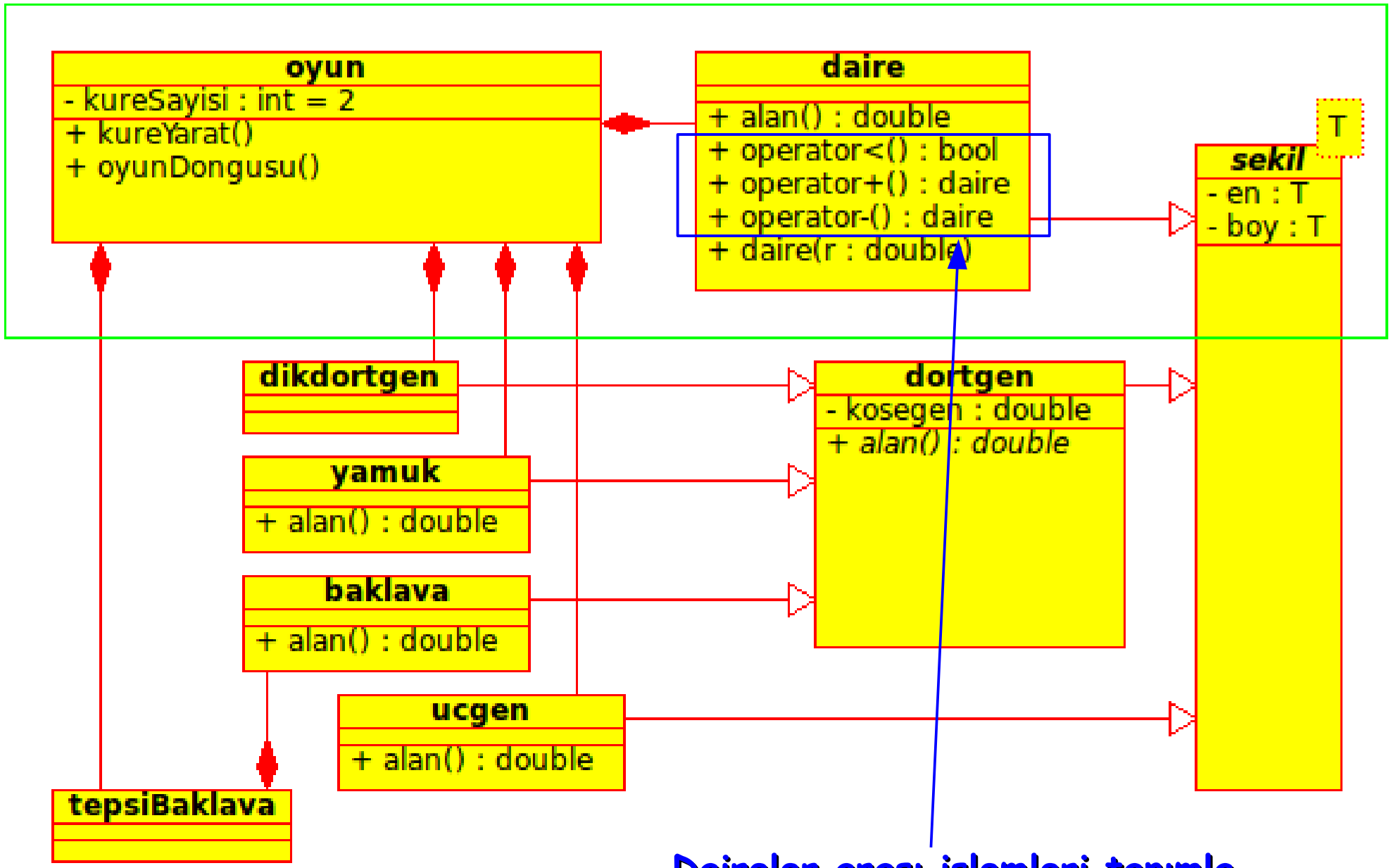
Big bang



Osmos

Geometrik Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

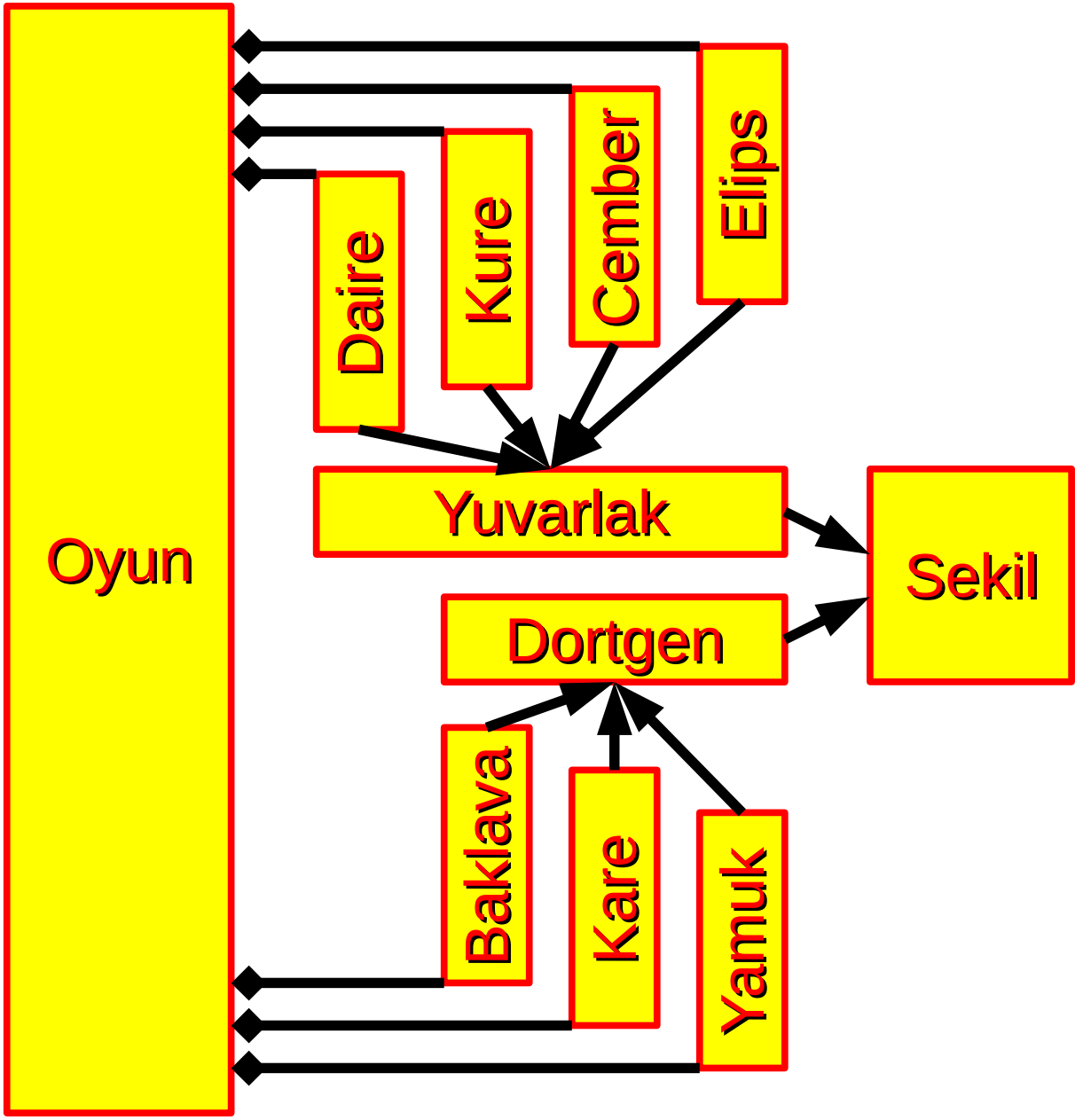


Geometrik Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

-**Ödev**: başka bir mimari düşünün; mesela:

```
int main() {

    // Değişken yarat
    oyun a;

    // Method çağır
    a.oyna(5, 76, 3);

    // Çık
    exit(0);
}
```



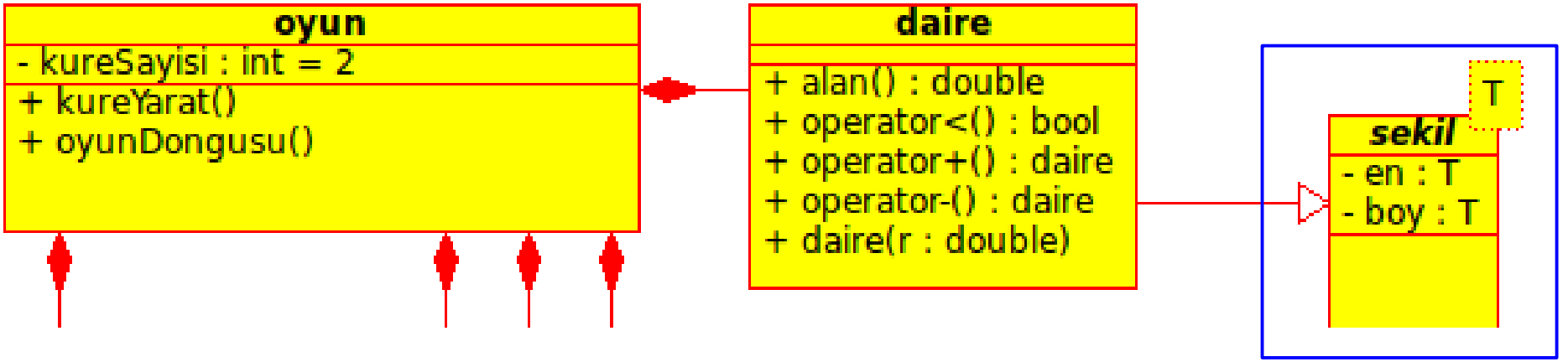
Geometrik Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Diğerlerinin kendisinden türeyeceği **soyut sınıfı** tanımlıyorum

```
1  class sekil {
2
3      protected:
4
5          double en;
6          double boy;
7
8      public:
9
10         // C'tor
11         sekil(double a = 1.0, double b = 1.0);
12
13         // Methods ( virtual means, derived
14         //           classes *can* redefine )
15         virtual double alan();
16         void setEn(double e);
17         void setBoy(double b);
18 };
```

```
1  #include"sekil.h" /* where sekil is defined */
2
3  //____________________________
4  sekil::sekil(double a, double b) {
5      en  = a;
6      boy = b;
7  }
8
9  //____________________________
10 double sekil::alan() {
11     return (en * boy);
12 }
13
14 //____________________________
15 void sekil::setEn(double e) {
16     en = e;
17 }
18
19 //____________________________
20 void sekil::setBoy(double b) {
21     boy = b;
22 }
```

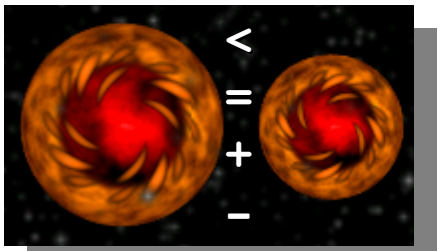


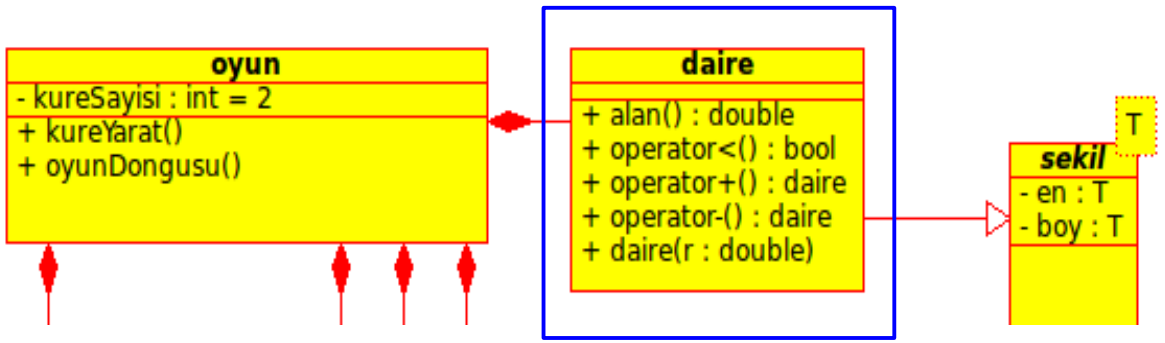
Geo. Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Şekilden türeyen **daire** ve daireler arasında gerçekleştirilebilecek **işlevleri** tanımlıyorum

```
#include"sekil.h"

class daire : public sekil {

    public:

        // C'tor
        daire(double r=1.0);

        // Methodlar
        double alan();
        void setR(double r);

        // Islemleri tanimliyorum
        bool operator<(daire a);
        daire operator+(daire a);
        daire operator-(daire a);
};

```






```
#include"daire.h"

//_____________________________
daire::daire(double r) {
    // C'tor = initializer
    en = boy = 2*r;
}

//_____________________________
void daire::setR(double r) {
    // Method
    en = boy = 2*r;
}

//_____________________________
double daire::alan() {
    // Method
    double r = en/2.0;
    return (3.141592*r*r);
}

//_____________________________
bool daire::operator<(daire a) {
    // Kucuktur islemcisi
    double r  = en/2.0;
    double ra = a.en/2.0;
    return (3.141592*r*r < 3.141592*ra*ra);
}

//_____________________________
daire daire::operator+(daire a) {
    // Toplama islemcisi
    daire sonuc(en/2 + a.en/2);
    return sonuc;
}

//_____________________________
daire daire::operator-(daire a) {
    // Cikarma islemcisi
    daire sonuc(en/2 - a.en/2);
    return sonuc;
}
```

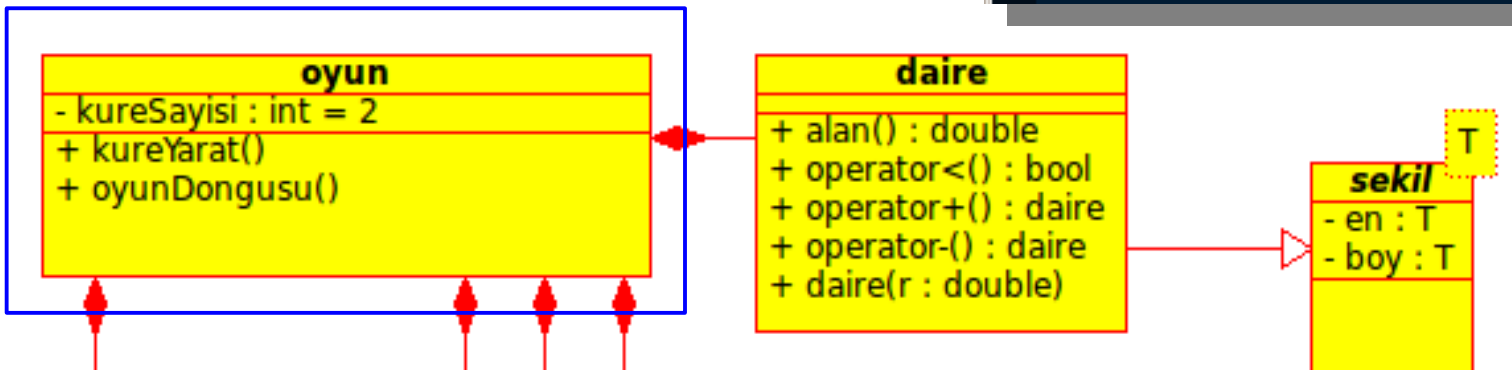
Geometrik Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

- İçinde **daire** cinsinden değişkenlerin yer alacağı **oyun sınıfı** tanımlanıyor
- **Oyun**'da **daire nesneleri** yaratılıp **karşılaştırılacak**

```
#include"daire.h"

class oyun {

    private:

        daire d0, d1, d2;

    public:

        // C'tor
        oyun(double a, double b);

        // Member
        void calistir();
};
```

```
#include<iostream>
#include"oyun.h"
using namespace std;

//_____________________________
oyun::oyun(double a, double b) {
    // Buyukluk ver
    d0.setR(a);
    d1.setR(b);
}

//_____________________________
void oyun::calistir() {
    // Ayni islemleri yap
    cout<<d0.alan()<<endl;  // Alanlarini goster
    cout<<d1.alan()<<endl;

    if (d0<d1) {            // Karsilastir
        cout<<"Sonuc: d0<d1"<<endl;
        d2 = d1-d0;         // Cikart
    } else {
        cout<<"Sonuc: d1<d0"<<endl;
        d2 = d1+d0;         // Topla
    }

    cout<<d2.alan()<<endl;  // Cikarma/Toplama
                            // sonucunu goster
}
```



Geometrik Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

- İki farklı yoğurt yeğiş:
  - + main'i kullan
    - Oyun'a gerek yok
    - Kalabalık, yeniden kullanımı zor
  - + oyun sınıfı'nı kullan
    - Sade, iş oyun sınıfı içine saklı, yeniden kullanımı kolay
    - Fazladan oyun sınıfına ihtiyaç var

```
 - I -
3.14159
28.2743
Sonuc: d0<d1
12.5664
```

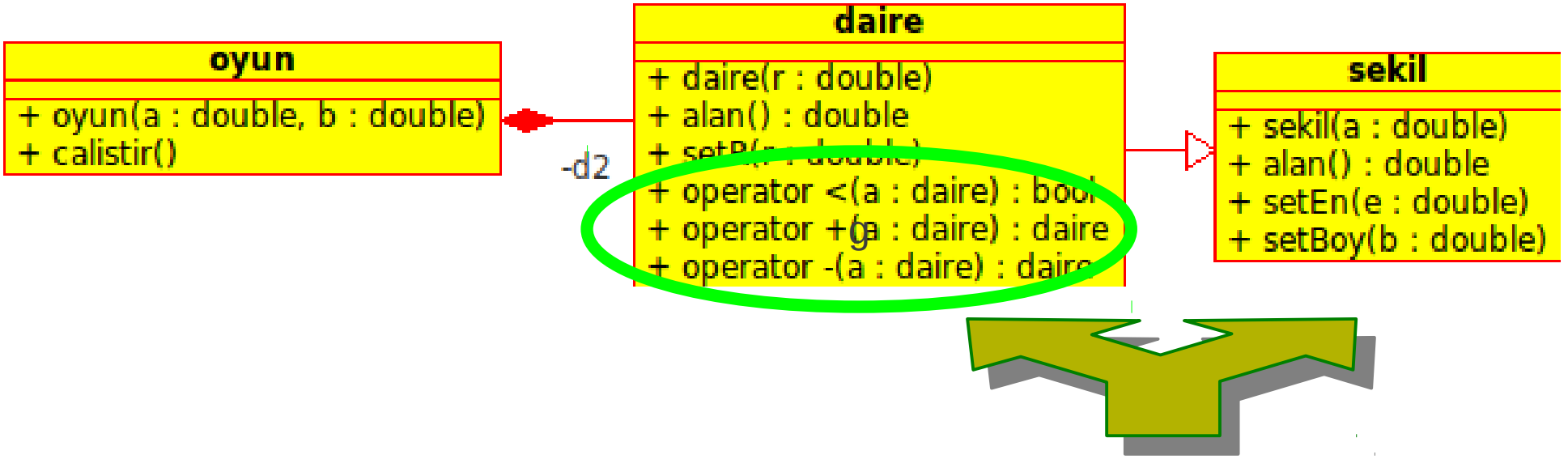
```
 - II -
3.14159
28.2743
Sonuc: d0<d1
12.5664
```

```
#include<iostream>    // std::cout burada tanimli
#include"oyun.h"      // oyun (analizim) veri tipi burada tanimli
using namespace std; // her defasinda 'std::' yazmayayim

int main() {

    double r0(1.0), r1(3.0);// to play with

    // OLASI ANA ROGRAM - I
    //-----------------------------------------
    cout << endl << " - I -" << endl;
    daire d0(r0);           // Daire yarat
    daire d1(r1);
    daire d2;

    cout << d0.alan() << endl;  // Alanlarini goster
    cout << d1.alan() << endl;

    if (d0<d1) {            // Karsilastir
        cout << "Sonuc: d0<d1" << endl;
        d2 = d1-d0;         // Cikart
    } else {
        cout << "Sonuc: d1<d0" << endl;
        d2 = d1+d0;         // Topla
    }

    cout << d2.alan() << endl;  // Cikarma/Toplama
                                // sonucunu goster

    // OLASI ANA ROGRAM - II
    //-----------------------------------------
    cout << endl << " - II -" << endl;
    oyun a(r0, r1);
    a.calistir();

    return 0;               // Kibarca cik
}
```

Geometrik Şekil Hiyerarşisi

Yeni veri tipi ve ilgili işlemcileri tanımlama

- Asağıdaki UML'yi sağlayan bir C++ programı varsaydığımızda, **operatörleri base-class'ta tanımlamak** dairenin implementasyonunu kısaltırdı ***ama*** ... ?



Operatörleri, daire'den alıp şekil sınıfına'ına alırsam ne olur ?

Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** Kütüphanesine **derlemek** ve **kullanmak**

- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması

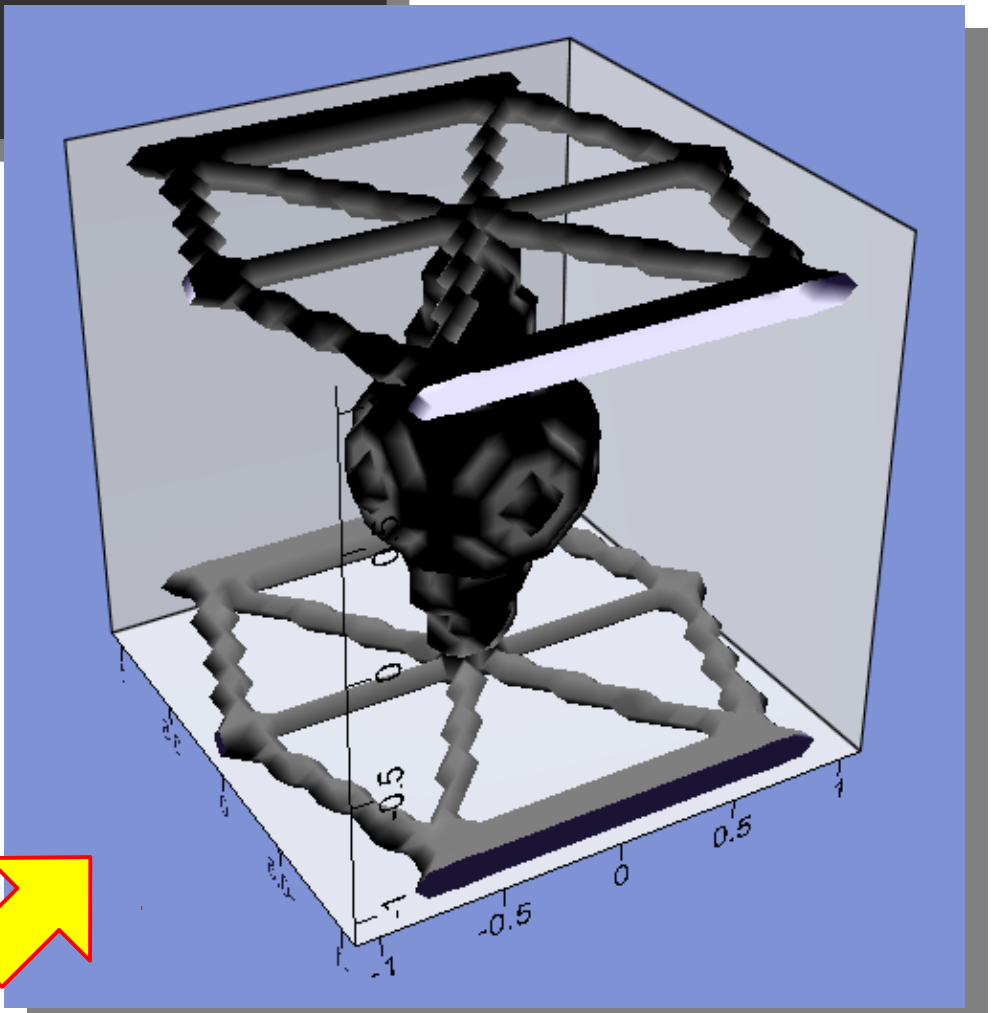
ROOT: Derle ve Merhaba Dünya Alıştırması

Yıldız savaşlarından Tie Fighter Örneği

```
> tar xvfz root_v5.32.00.source.tar.gz
> export ROOTSYS=$HOME/root
> export PATH=$PATH:$ROOTSYS/bin
> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
> cd root
> ./configure
> make
> sudo make install
> _
```

- Çevreselleri **ata**, **derle**, **yükle** ve ROOT kullanıma **hazır**

```
# root -l
> TF3 *tieFighter = new
  TF3("tieFighter","(x^2+y^2+z^2<0.2)+
  ((y^2+z^2<0.08)*(x<0.4)*(x>0))+
  (x^2+4*y^2<(1-TMath::Abs(z))*0.12)+
  ((TMath::Abs(z)<0.95)*(TMath::Abs(z)
  >0.9)*(TMath::Abs(x)
  +TMath::Abs(y)*0.3<1))+
  ((TMath::Abs(z)<1)*(TMath::Abs(z)>0.
  89))*((TMath::Abs(x)<0.7)*(TMath::Ab
  s(y)>0.9)+(TMath::Abs(y)<0.035)+
  (x>y*0.7-0.05)*(x<y*0.7+0.05)+(-
  x>y*0.7-0.05)*(-x<y*0.7+0.05)+
  ((TMath::Abs(x)
  +TMath::Abs(y)*0.3<1.05)*(TMath::Abs
  (x)+TMath::Abs(y)*0.3>0.95)))",-
  1.1,1.1,-1.1,1.1,-1.1,1.1);
> tieFighter->Draw()
```



Algoritmik Düşünce ve C/C++ Hakkında

Programlama etkinliği üzerine bir değerlendirme

- **Dikkat !!** Çoğunlukla yaptığım temel **hatalar**
  - **ForTran** yazıp, adına **C** deyip, **C++** sanmak
  - OO kütüphane kullan **!=** C++ programla

- Amaç
  - Geliştirici ne **ister** ne **yapar** ?
- Programlama **Paradigmaları**
  - **Sıralı**: Temel Programlama İfadeleri
    - Telefon defteri uygulaması
      - **Döngüler** ve **şart koşma**
      - **Rastlantısal sayılar** ile π'nin hesaplanması
      - Farklı dillerdeki **uygulaması** ve **akış çizelgesi**
  - **Nesne yönelimli**: karmaşadaki sadelik
    - **ÇokGen**'den türeyen **ÜçGen** ve **DörtGen**
    - **Çizgiden** Türeyen **Ok**
      - Cizgi class'ı ve uygulaması
      - Ok class'ı ve uygulaması
      - Kullanıcı programındaki **kullanımı**
      - Kullanıcı programında **işaretçi ile** kullanımı
- Kaynak (**code**), Öbek (**heap**) ve Yığın (**stack**)
  - Yazılımların kullandığı **hafıza bölgeleri**
- **Kütüphane derlemek** ve kullanmak
  - Cizgi ve ok class'larını **libCizgiVeOk.so** kütüphanesine **derlemek** ve **kullanmak**

- Düşünmek derken: **Akış** ve **UML çizelgeleri**
  - **UML** ile düşünce/mimari **geliştirmek**
    - Hareketli parçacıklar (**sıyırduino**)
    - Sipariş alan **hizmet sağlayıcı** örneği
- **UML ile C++** Arasındaki İlişki
  - **Yeni** veri tipi ve ilgili **operatörleri** tanımlama
    - iPad'in Osmos'u ya da Android'in Big Bang'i ve **bölüm sonu canavarları** arası karşılaştırma
    - Daire **class'ının** tanımlanması
    - Daireler arası **işlemcilerin** geliştirilmesi
    - Kullanıcı programı
- **ROOT'** tan Bahis...
  - Kurulum ve "**Merhaba dünya !!**" alıştırması