



# C++ - Modül 03

## Kalıtım

*Özet:*

*Bu doküman C++ modüllerinden Modül 03'ün alıştırmalarını içermektedir.*

*Sürüm: 7.1*

# İçindekiler

<b>I</b>	<b>Giriş</b>	<b>2</b>
<b>II</b>	<b>Genel kurallar</b>	<b>3</b>
<b>III</b>	<b>Egzersiz 00: Aaaaand... AÇILIN!</b>	<b>6</b>
<b>IV</b>	<b>Alıştırma 01: Serena, aşkım!</b>	<b>8</b>
<b>V</b>	<b>Egzersiz 02: Tekrarlı çalışma</b>	<b>9</b>
<b>VI</b>	<b>Alıştırma 03: şimdi tuhaf oldu!</b>	<b>10</b>
<b>VII</b>	<b>Sunum ve akran değerlendirmesi</b>	<b>12</b>

# Bölüm I Giriş

*C++, Bjarne Stroustrup tarafından C programlama dilinin ya da "C with Classes" (Sınıflı C) dilinin bir uzantısı olarak yaratılmış genel amaçlı bir programlama dilidir (kaynak: [Wikipedia](#)).*

Bu modüllerin amacı sizi **Nesne Yönelimli Programlama** ile tanıştırmaktır. Bu, C++ yolculuğunuzun başlangıç noktası olacaktır. OOP öğrenmek için birçok dil önerilmektedir. Eski dostunuz C'den türetildiği için C++'ı seçmeye karar verdik. Bu karmaşık bir dil olduğundan ve işleri basit tutmak için kodunuz C++98 standardına uygun olacaktır.

Modern C++'ın pek çok açıdan çok farklı olduğunun farkındayız. Dolayısıyla, yetkin bir C++ geliştiricisi olmak istiyorsanız, 42 Common Core'dan sonra daha ileri gitmek size kalmış!

# Bölüm II Genel

## kurallar

### Derleme

- Kodunuzu c++ ve -Wall -Wextra -Werror bayrakları ile derleyin
- Eğer -std=c++98 bayrağını eklerseniz kodunuz yine de derlenmelidir

### Biçimlendirme ve adlandırma kuralları

- Alıştırma dizinleri şu şekilde adlandırılacaktır: ex00, ex01, ... , exn
- Dosyalarınızı, sınıflarınızı, işlevlerinizi, üye işlevlerinizi ve niteliklerinizi yönergelerde belirtildiği şekilde adlandırın.
- Sınıf adlarını **UpperCamelCase** biçiminde yazın. Sınıf kodu içeren dosyalar her zaman sınıf adına göre adlandırılacaktır. Örneğin: ClassName.hpp/ClassName.h, .cpp veya .hpp. Bu durumda, tuğla duvar anlamına gelen "BrickWall" sınıfının tanımını içeren başlık dosyanız varsa, adı BrickWall.hpp olacaktır.
- Aksi belirtilmedikçe, her çıktı mesajı bir yeni satır karakteriyle sonlandırılmalı ve standart görüntülenmelidir.
- *Güle güle Norminette!* C++ modüllerinde herhangi bir kodlama stili zorunlu değildir. En sevdiğinizi takip edebilirsiniz. Ancak, akran değerlendiricilerinizin anlayamadığı bir kodun not veremeyecekleri bir kod olduğunu unutmayın. Temiz ve okunabilir bir kod yazmak için elinizden geleni yapın.

### İzinli/Yasak

Artık C'de kodlama yapmıyorsunuz. C++ zamanı! Bu nedenle:

- Standart kütüphanedeki neredeyse her şeyi kullanmanıza izin verilir. Bu nedenle, zaten bildiklerinize bağlı kalmak yerine, alışkın olduğunuz C işlevlerinin C++-ish sürümlerini mümkün olduğunca kullanmak akıllıca olacaktır.
- Ancak, başka herhangi bir harici kütüphane kullanamazsınız. Bu, C++11 (ve türetilmiş formlar) ve Boost kütüphanelerinin yasak olduğu anlamına gelir. Aşağıdaki fonksiyonlar da yasaktır: \*printf(), \*alloc() ve free(). Eğer bunları kullanırsanız, notunuz 0 olacaktır ve hepsi bu kadar.

- Aksi açıkça belirtilmedikçe, using namespace <ns\_name> ve arkadaş anahtar kelimeleri yasaktır. Aksi takdirde notunuz -42 olacaktır.
- **STL'yi sadece Modül 08 ve 09'da kullanmanıza izin verilmektedir.** Bunun anlamı: o zamana kadar **Kapsayıcı** (vektör/liste/harita/ve benzeri) ve **Algoritma** (<algorithm> başlığını içermesi gereken herhangi bir şey) kullanmayacaksınız. Aksi takdirde notunuz -42 olacaktır.

### Birkaç tasarım gereksinimi

- C++'da da bellek sızıntısı meydana gelir. Bellek ayırdığınızda (new anahtar sözcüğü), **bellek sızıntılarından** kaçınmalısınız.
- Modül 02'den Modül 09'a kadar, **aksi açıkça belirtilmediği sürece**, dersleriniz **Ortodoks Kanonik Formunda** tasarlanmalıdır.
- Bir başlık dosyasına konulan herhangi bir işlev uygulaması (işlev şablonları hariç) alıştırma için 0 anlamına gelir.
- Başlıklarınızın her birini diğerlerinden bağımsız olarak kullanabilmelisiniz. Bu nedenle, ihtiyaç duydukları tüm bağımlılıkları içermelidirler. Ancak, **include korumaları** ekleyerek çifte içirme sorunundan kaçınmalısınız. Aksi takdirde notunuz 0 olacaktır.

### Beni oku

- Gerekirse bazı ek dosyalar ekleyebilirsiniz (örneğin, kodunuzu bölmek için). Bu ödevler bir program tarafından doğrulanmadığından, zorunlu dosyaları teslim ettiğiniz sürece bunu yapmaktan çekinmeyin.
- Bazen bir alıştırmanın yönergeleri kısa görünebilir ancak örnekler, açıkça yazılmayan gereksinimleri gösterebilir.
- Başlamadan önce her modülü tamamen okuyun! Gerçekten okuyun.
- Odin tarafından, Thor tarafından! Beynini kullan!!!



C++ projeleri için Makefile ile ilgili olarak, C'deki aynı kurallar geçerlidir (Makefile ile ilgili Norm bölümüne bakın).




Çok sayıda sınıf uygulamanız gerekecek. En sevdiğiniz metin düzenleyicinizi komut dosyası haline getiremediğiniz sürece bu sıkıcı görünebilir.



Egzersizleri tamamlamanız için size belirli bir miktar özgürlük verilir. Ancak, zorunlu kurallara uyun ve tembellik etmeyin. Pek çok faydalı bilgiyi kaçırsınız! Teorik kavramlar hakkında okumaktan çekinmeyin.

## Bölüm III

### Egzersiz 00: Aaaaand... AÇILIN!

	Egzersiz : 00
Aaaaand... AÇILIN!	
Giriş dizini : <i>ex00/</i>	
Teslim edilecek dosyalar : Makefile, main.cpp, ClapTrap.{h, hpp}, ClapTrap.cpp	
Yasak fonksiyonlar : Hiçbiri	

İlk olarak, bir sınıf uygulamak zorundasınız! Ne kadar orijinal!

**ClapTrap** olarak adlandırılacak ve parantez içinde belirtilen değerlere ilklendirilmiş aşağıdaki özel sahip olacaktır:

- Bir kurucuya parametre olarak aktarılan ad
- Hit puanları (10), ClapTrap'in sağlığını temsil eder
- Enerji puanları (10)
- Saldırı hasarı (0)

ClapTrap'in daha gerçekçi görünmesi için aşağıdaki genel üye işlevleri ekleyin:

- `void attack(const std::string& target);`
- `void takeDamage(unsigned int amount);`
- `void beRepaired(unsigned int amount);`

ClapTrap saldırıdığı anda, hedefinin <saldırı hasarı> can puanı kaybetmesine neden olur. ClapTrap kendini onardığında, <miktar> can puanını geri kazanır. Saldırmak ve onarmak 1'er enerji puanına mal olur. Elbette, ClapTrap'in hiç can puanı veya enerji puanı kalmamışsa hiçbir şey yapamaz. Ancak, bu alıştırmalar bir giriş niteliğinde olduğundan, ClapTrap örnekleri birbirleriyle doğrudan etkileşime girmemelidir ve parametreler başka bir ClapTrap örneğine atıfta bulunmayacaktır.

Tüm bu üye fonksiyonlarda, ne olduğunu açıklamak için bir mesaj yazdırmanız gerekir. Örneğin, `attack()` fonksiyonu aşağıdaki gibi bir şey görüntüleyebilir (tabii ki köşeli parantezler olmadan):

`ClapTrap <isim> <hedef> saldırarak <hasar> puanı hasar veriyor!`


Kurucular ve yıkıcılar da bir mesaj göstermelidir, böylece eş-değerlendiricileriniz bunların çağrıldığını kolayca görebilir.

Kodunuzun beklendiği gibi çalıştığından emin olmak için kendi testlerinizi uygulayın ve teslim edin.



## Bölüm IV

### Egzersiz 01: Serena, aşkım!

	Egzersiz : 01
	Serena, aşkım!
	Giriş dizini : <i>ex01/</i>
	Teslim edilecek dosyalar : Önceki alıştırmadan dosyalar+ ScavTrap.{h, hpp}, ScavTrap.cpp
	Yasak fonksiyonlar : Hiçbiri

Asla yeterince ClapTrap'e sahip olamayacağınız için, şimdi türetilmiş bir robot yaratacaksınız. Bu robotun adı **ScavTrap** olacak ve yapıcılar ile yıkıcısını ClapTrap'ten miras alacak. Ancak, yapıcılar, yıkıcısı ve `attack()` farklı mesajlar yazdıracaktır. Sonuçta, ClapTrap'ler bireyselliklerinin farkındadırlar.

Testlerinizde uygun yapım/yıkım zincirlemesinin gösterilmesi gerektiğini unutmayın. Bir ScavTrap oluşturulduğunda, program bir ClapTrap inşa ederek başlar. Yıkım ters sırada gerçekleşir. Neden?

**ScavTrap**, ClapTrap'ın niteliklerini kullanacaktır (sonuç olarak ClapTrap'ı güncelleyecektir) ve bunları şu şekilde başlatmalıdır:

- Bir kurucuya parametre olarak aktarılan ad
- Hit puanları (100), ClapTrap'ın sağlığını temsil eder
- Enerji puanları (50)
- Saldırı hasarı (20)

ScavTrap ayrıca kendi özel kapasitesine de sahip olacaktır:


```
void guardGate();
```

Bu üye işlevi ScavTrap'in şu anda Kapı bekçisi modunda olduğunu bildiren bir mesaj görüntüler.

Programınıza daha fazla test eklemeyi unutmayın.

## Bölüm V

### Egzersiz 02: Tekrarlı çalışma

	Alıştırma : 02
Tekrarlayan işler	
Dönüş dizini : <i>ex02/</i>	
Teslim edilecek dosyalar : Önceki alışırtırmalardan dosyalar+ FragTrap.{h, hpp}, FragTrap.cpp	
Yasak fonksiyonlar : Hiçbiri	

ClapTrap yapmak muhtemelen sinirlerinizi bozmaya başlamıştır.

Şimdi, ClapTrap'ten miras alan bir **FragTrap** sınıfı uygulayın. ScavTrap'a çok benzer. Ancak, yapım ve yıkım mesajları farklı olmalıdır. Testlerinizde uygun yapım/yıkım zincirleme gösterilmelidir. Bir FragTrap oluşturulduğunda, program bir ClapTrap inşa ederek başlar. Yıkım ters sırada gerçekleşir.

Nitelikler için aynı şeyler, ancak bu sefer farklı değerler:

- Bir kurucuya parametre olarak aktarılan ad
- Hit puanları (100), ClapTrap'ın sağlığını temsil eder
- Enerji puanları (100)
- Saldırı hasarı (30)

FragTrap'ın da özel bir kapasitesi vardır:


```
void highFivesGuys(void);
```

Bu üye işlev, standart çıktıda pozitif bir beşlik isteği görüntüler.

Yine, programınıza daha fazla test ekleyin.

## Bölüm VI

### Alıştırma 03: şimdi tuhaf oldu!

	Egzersiz : 03
İşte şimdi tuhaf oldu!	
Dönüş dizini : <i>ex03/</i>	
Teslim edilecek dosyalar : Önceki alışırtımalardan dosyalar+ DiamondTrap.{h, hpp}, DiamondTrap.cpp	
Yasak fonksiyonlar : Hiçbiri	

Bu alıştırmada bir canavar yaratacaksınız: yarı FragTrap, yarı ScavTrap olan bir ClapTrap. Adı **DiamondTrap** olacak ve hem 'ten hem de ScavTrap'ten miras alacak. Bu çok riskli!

DiamondTrap sınıfı bir name private niteliğine sahip olacaktır. Bu niteliğe ClapTrap temel sınıfındakiyle tam olarak aynı değişken adını verin (burada robotun adından bahsetmiyoruz).

Daha açık olmak gerekirse, işte iki örnek.  
ClapTrap'in değişkeni name ise, DiamondTrap'in değişkenine name adını verin.  
ClapTrap'in değişkeni \_name ise, DiamondTrap'in değişkenine \_name adını verin.

Nitelikleri ve üye işlevleri, ana sınıflarından herhangi birinden seçilecektir:

- Bir kurucuya parametre olarak aktarılan ad
- ClapTrap::name (kurucunun parametresi+ "\_clap\_name" son eki)
- Hit puanları (FragTrap)
- Enerji noktaları (ScavTrap)
- Saldırı hasarı (FragTrap)
- attack() (Scavtrap)

Her iki ana sınıfın özel işlevlerine ek olarak, DiamondTrap kendi özel kapasitesine sahip olacaktır:

```
void whoAmI();
```

Bu üye fonksiyon hem kendi adını hem de ClapTrap adını görüntüler.

Elbette, DiamondTrap'in ClapTrap alt nesnesi bir kez ve yalnızca bir kez oluşturulacaktır. Evet, bir numara var.

Yine, programınıza daha fazla test ekleyin.



Wshadow ve -Wno-shadow derleyici bayraklarını biliyor musunuz?



Bu modülü egzersiz 03'ü yapmadan geçebilirsiniz.

## Bölüm VII

### Sunum ve akran değerlendirmesi

Ödevinizi her zamanki gibi Git deponuzda teslim edin. Savunma sırasında yalnızca deponuzdaki çalışmalar değerlendirilecektir. Doğru olduklarından emin olmak için klasörlerinizin ve dosyalarınızın adlarını iki kez kontrol etmekten çekinmeyin.



????????????? XXXXXXXXXXXX= \$3\$\$cf36316f07f871b4f14926007c37d388