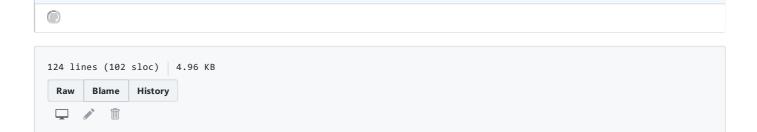


Fetching contributors...

## Cplusplus\_Ders\_Notlari / covariant\_return\_type.md



## eş değişken geri dönüş değeri türü (covariant return type)

Türemiş bir sınıfın taban sınıfının bir sanal işlevini ezecek (override) bir işlevinin, taban sınıf işleviyle hem aynı imzaya hem de aynı geri dönüş türüne sahip olması gerekir:

```
class Car {
public:
    //...
    virtual double getCylinderVolume()const = 0;
};

class Mercedes : public Car {
public:
    //...
    virtual float getCylinderVolume()const override;
};
```

Yukarıdaki kodda Mercedes sınıfı taban sınıfı olan Car sınıfının saf sanal getCylinderVolume işlevini ezerken (ovverride) farklı bir geri dönüş değeri bildirmiş. C++ dilinin kurallarına göre bu durum bir sentaks hatası oluşturuyor. Ancak bu durumun "eşdeğişken geri dönüş türü" (covariant return type) denen bir istisnası var. B bir sınıf türü olmak üzere, taban sınıfın sanal işlevinin geri dönüş değeri B\* türünden ise, türemiş sınıfın bu işlevi ezecek işlevinin geri dönüş değeri, eğer D sınıfı B sınıfından public türetmesiyle elde edilmiş bir sınıf ise, D\* türünden olabilir. Aynı istisna referans semantiği için de söz konusu: B bir sınıf türü olmak üzere, taban sınıfın sanal işlevinin geri dönüş değeri B& türünden ise türemiş sınıfın bu işlevi ezecek işlevinin geri dönüş değeri, eğer D sınıfı B sınıfından public türetmesiyle elde edilmiş bir sınıf ise, D& türünden olabilir. Car sınıfının arayüzünde bulunan sanal kurucu işlev olarak görev yapacak saf sanal (pure virtual) bir clone işlevinin yer aldığını düşünelim:

```
class Car {
public:
    //...
    virtual Car *clone() = 0;
};

class Mercedes : public Car {
public:
    //...
    virtual Mercedes *clone()override; //covariant return type
};

void carGame(Car *p)
{
    Car *pNewCar = p->clone();
    //
}
```

Mercedes sınıfı Car sınıfının clone işlevini ezerken işlevin geri dönüş türünü Mercedes \* olarak değiştirdiğini görüyorsunuz. Bu kod geçerli, çünkü her Mercedes nesnesi aynı zamanda bir Car nesnesi. Peki neden böyle bir istisnaya duyulmuş? Eşdeğişken geri dönüş türü, türemiş sınıf nesnelerinin türemiş sınıf arayüzüyle işlendiği durumlar için önemli bir avantaj sağlıyor:

```
Mercedes *getNewMercedes();
int main()
{
   Mercedes *pm1 = getNewMercedes();
   Mercedes *pm2 = pm1->clone();
   //...
}
```

Yukarıdaki kodda bildirilen getNewMercedes isimli global işlev dinamik bir Mercedes nesnesinin adresini döndürüyor. main işlevi içinde bu işlevin geri dönüş değerinin doğrudan Mercedes türünden bir göstericiye ilk değer verdiğini görüyorsunuz. Eğer Mercedes sınıfının clone işlevinin geri dönüş türü Car \* olsaydı, yani eşdeğişken geri dönüş türü istisnası olmasaydı bu kod geçersiz olacaktı. Kodun geçerliliğini sağlamak için static\_cast tür dönüştürme işlecini kullanmamız gerekecekti:

```
int main()
{
   Mercedes *pm1 = getNewMercedes();
   Mercede *pm2 = static_cast<Mercedes *>(pm1->clone());
   //...
}
```

Bir örnek de FactoryMethod tasarım kalıbının gerçekleştiriminden verelim: Aşağıdaki kodda Car sınıfının ara yüzünde fabrika metodu olarak görev yapacak getServiceStation isimli bir saf sanal işlev yer alıyor. Car sınıfından kalıtım yoluyla elde edilecek somut (concrete) sınıflar bu işlevi ezerek kendilerine uygun somut bir ServiceStatiton nesnesini referans semantiği ile geri döndürecekler:

```
class ServiceStation {
///...
};
class Car {
public:
//...
virtual ServiceStation &getServiceStation() = 0;
class Mercedes;
class MercedesServiceStation : public ServiceStation {
//...
};
class Mercedes : public Car {
public:
MercedesServiceStation &getServiceStation()override;
//...
};
```

Burada dikkat edilmesi gereken bir nokta daha var: eşdeğişken geri dönüş türünün kullanılabilmesi için Mercedes sınıfının getServiceStation işlevinin bildiriminden önce MercedesServiceStation sınıfının sadece bildirilmiş olması (forward declaration) yeterli değil, sınıf tanımlanmış olmalı. Derleyici MercedesServiceStation referansını (ya da adresini) ServiceStation referansına (ya da adresine) dönüştürebilmek için MercedesServiceStation sınıfının belleğe yerleşim verilerine erişmek zorunda.

Eşdeğişken geri dönüş türünün avantajı bize her zaman uygun bir soyutlama (abstraction) seviyesi sağlaması. Eğer Car sınıfı türünden nesneler ile çalışıyor isek soyut (abstract) bir ServiceStation elde edeceğiz. Eğer spesifik bir Car türü, örneğin Mercedes türü ile çalışıyor isek somut bir MercedesServiceStation nesnesi elde edeceğiz:

```
Car *getNewCar();
Mercedes *getNewMercedes();

int main()
{
   Car *pcar = getNewCar();
   ServiceStation &rss = pcar->getServiceStation();
   Mercedes *pm = getNewMercedes();
   MercedesServiceStation &rmss = pm->getServiceStation();
   //...
}
```

```
© 2020 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About
```