**Static ve Dinamik Tür(Static and Dynamic Type)**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#static-ve-dinamik-t%C3%BCrstatic-and-dynamic-type)

Referans ve nesne türlerinden bahsederken bu konuyu da ele almamızda yarar var diye düşünüyorum. Aslında static ve dinamik tip(typing), programlama dillerini birbirlerinden ayıran belli başlı tercihlerden biridir. Her ne kadar burada “type” yazım olarak anlaşılsa da tür/tip olarak düşünülmesi daha doğru olacaktır. Bazı bölümlerde yazım olarak dile getirsem de dinamik veyahut statik tip olarak anlaşılması gerektiğini düşünebilirsiniz. Nedenini ilerleyen bölümlerde daha iyi anlayacaksınız.

**Tür Kontrolü(type checking)**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#t%C3%BCr-kontrol%C3%BCtype-checking)

Static ve dinamik olarak yaptığımız bu ayrımın asıl sebebi tür denetimi, ve bu denetimin nerede gerçekleştiği ile alakalıdır. Öncelikli olarak şu tanımı yapmamızda yarar var. Dinamik tipte yazılan diller **çalışma zamanında(run-time)** *tür denetimi* gerçekleştirirken, statik tipte yazılan diller **derleme zamanında(compile-time)** *tür denetimi* gerçekleştirir. Tür denetimi derleme zamanında ise statik kontrol(static checking), çalışma zamanında dinamik kontrol(dynamic checking) olarak gerçekleşebilir. Anlaşılacağı üzere, dinamik tip dillerde yazılmış kodlar, düzgün çalışmasını engelleyecek hata içeriyor olsa bile derlenecek demektir. Yani derleme hatası diye bir şey almazsınız. Çünkü bütün işlem çalışma zamanında(run-time) gerçekleşir ve hata alınacaksa da çalışma zamanında alınır. Groovy, Python, Ruby, Php, javascript dinamik tip dillere birer örnektir. Fakat, statik tip bir dilde (Java, C, C++, FORTRAN, Pascal ve Scala gibi) yazılmış bir kod, hata içeriyorsa, hatalar düzeltilene kadar derlenemez. Yani değişken türü, çalışma zamanı yerine derleme zamanında biliniyorsa, dil statik tipde yazılmış demektir. Burada, bu iki gruba birer örnek vermek istiyorum.

*Bu arada tür denetimi, programın tür güvenli(type-safe) olduğundan emin olmakla ilgilidir ve tür hataları olasılığını en aza indirir. Hatta TypeScript dilinin de çıkış amacı bu yüzdendir. Amaç javascript dilini tür güvenli hale getirmek.*

**Static Tip Dile Örnek**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#static-tip-dile-%C3%B6rnek)

Aşağıdaki örnek java’da yazılmış bir kod bloğudur. Görüleceği üzere a değişkeninin tipi önceden deklare ediliyor. Yani bir **integer**. Ama sonrasında bu değişkene bir **string** atanmak isteniyor. Haliyle program çalışma zamanı öncesinde, yani derleme zamanında bir hata verir ve bu hatayı düzeltmeniz istenir. Bu yüzden referans türleri derleme zamanı kararlarında, nesne türleri ise çalışma zamanı kararlarında göz önüne alınır. Özetle, java ve java gibi static diller, çalışma zamanına(run-time) geçmeden heap alanında hangi nesneyi referans aldığını bilemez.

|  |  |
| --- | --- |
| 1  2  3 | int a;  a = 2;  a = "Hello"; // causes an compilation error |

[İlgili koda bu link üzerinden de ulaşabilirsiniz](http://www.pythontutor.com/java.html#code=public%20class%20YourClassNameHere%20%7B%0A%20%20%20%20public%20static%20void%20main%28String%5B%5D%20args%29%20%7B%0A%20%20%20%20%20%20%20%20int%20a%3B%0A%20%20%20%20%20%20%20%20a%20%3D%202%3B%0A%20%20%20%20%20%20%20%20a%20%3D%20%22Hello%22%3B%20//%20causes%20an%20compilation%20error%0A%20%20%20%20%7D%0A%7D&cumulative=false&heapPrimitives=nevernest&mode=edit&origin=opt-frontend.js&py=java&rawInputLstJSON=%5B%5D&textReferences=false)

Statik tip dillerde, bir değişken bir türle bildirildikten sonra, hiçbir zaman farklı türdeki başka bir değişkene atanamaz. Şayet bunu yaparsanız derleme zamanında bir tür hatası ile karşılaşırsınız. Alacağınız hata şu şekildedir. **Error: incompatible types: java.lang.String cannot be converted to int.** Ama şimdi göreceğimiz dinamik tip dillerde böyle bir şey söz konusu değildir.

**Statik Tip Dillerin Avantajları**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#statik-tip-dillerin-avantajlar%C4%B1)

* Geliştirme sürecinin erken aşamasında büyük oranda hata yakalanır.(Yani derleme zamanında)
* Statik tip diller genellikle daha hızlı çalışan derlenmiş kodla sonuçlanır, çünkü derleyici kullanımda olan tüm veri türlerini bildiğinde, optimize edilmiş makine kodu üretebilir. Sonuç itibariyle daha hızlı ve / veya daha az bellek kullanılır.

Static tip dillerin örneklerini bu [linkten](https://en.wikipedia.org/wiki/Category:Statically_typed_programming_languages) görebilirsiniz.

**Dinamik Tip Dile Örnek**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#dinamik-tip-dile-%C3%B6rnek)

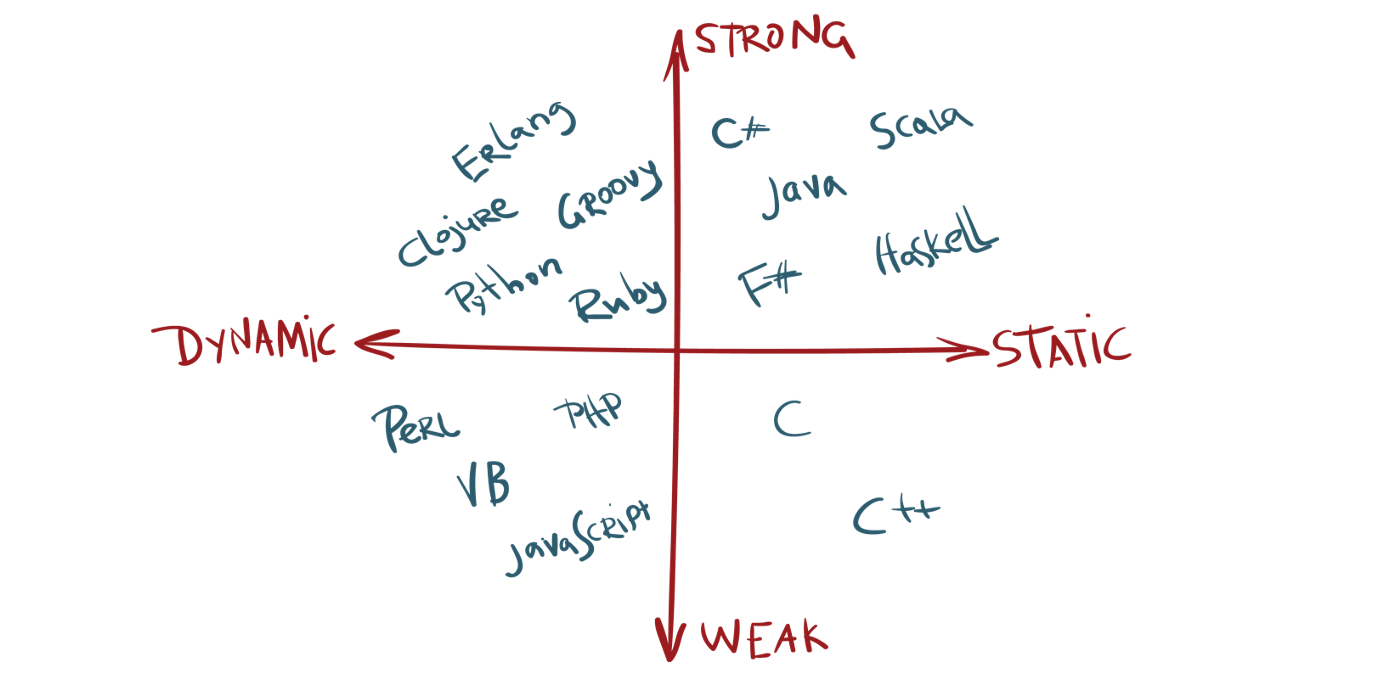
Bu örnek ise dinamik tipte yazılmış bir dil olan python’da oluşturulmuştur.

|  |  |
| --- | --- |
| 1  2 | data = 10;  data = "Hello World!"; |

Çalışma zamanında bir değişkenin türü kontrol edilirse bu, dinamik tip bir dil olarak adlandırılır. Dinamik olarak yazılan dillerde, değişkenlerin veri türlerini bildirmenize gerek yoktur. Yani java’da olduğu gibi referans türü diye bir şey yoktur. Yukarıdaki ifade de, öncesinde bir sayı atanandan aynı değişkene, bu sefer **string** bir değer atanıyor. Program başarıyla hatasız yürütülecektir. Bu, dinamik tip programlama dillerinin karakteristiğidir. Yani bütün olay çalışma zamanında gerçekleşir.

Dinamik tip dillerde değişkenler nesnelere, çalışma zamanında atama ifadeleriyle bağlanır ve programın yürütülmesi sırasında aynı değişkenleri farklı türdeki nesnelere bağlamak mümkündür. Bir sonraki bölüm zaten statik ve dinamik bağlama konusu olacak. Burada hangi durumlarda statik bağlama, hangi durumlarda dinamik bağlama tercih edilir, ona bakacağız.

*Yukarıdakilere ek olarak iki farklı gruptan daha bahsetmek istiyorum. Aslında dinamik ve statik tip diller de kendi içerisinde 2 farklı gruba ayrılır. Bunlar****strong-typed****ve****weakly-typed****dillerdir. Yalnız şunu belirtmekte yarar var. Statik olup,****strong-typed****veyahut****weakly-typed****olan da olabilir. Aynı şekilde dinamik olup****strong-typed****veyahut****weakly-typed****olan da olabilir. Aşağıdaki şekilde bu gruplamayı rahatlıkla görebilirsiniz.*



Bir dil şartnamesi, tür denetimi kurallarını(typing rules) şiddetle uyguluyorsa, süreç **“strong-typed”** olarak adlandırılır, uygulamıyorsa **“weakly-typed”** olarak adlandırılır.

**Dinamik Tip Dillerin Avantajları**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#dinamik-tip-dillerin-avantajlar%C4%B1)

* Ayrı bir derleme adımının olmaması, kod değişikliklerinizi test edebilmeniz için derleyicinin bitmesini beklemeniz gerekmediği anlamına gelir. Bu, hata ayıklama döngüsünü çok daha kısa ve daha az hantal yapar. Yani derleme zamanının olmaması, daha hızlı geri dönüş anlamına gelir
* Dinamik tür denetimli dillerin uygulamaları, genellikle her çalışma zamanı(run-time) nesnesini tür bilgilerini içeren bir tür etiketiyle(yani bir türe bir referans şeklinde) ilişkilendirir. Bu çalışma zamanı tür bilgileri(RTTI), dinamik dağıtım(dynamic dispatch), geç bağlama(late binding), aşağı akış(downcasting), yansıma(reflection) ve benzer özellikleri uygulamak için de kullanılabilir.

Dinamik tip dillerin örneklerini bu [linkten](https://en.wikipedia.org/wiki/Category:Dynamically_typed_programming_languages) görebilirsiniz.

**Güçlü Tip Dile Örnek**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#g%C3%BC%C3%A7l%C3%BC-tip-dile-%C3%B6rnek)

**Strong-typed**(güçlü-tip) bir dil, değişkenlerin belirli veri türlerine bağlı olduğu dildir ve türler, denetimin ne zaman gerçekleştiğine bakılmaksızın, bulundukları ifadede beklendiği gibi eşleşmezse **tür hatalarına** neden olur. Python, tıpkı Java gibi **strong-typed**(güçlü-tip) dillerden biridir. Java’nın statik tip bir dil, python’un ise dinamik tip bir dil olduğunu zaten biliyoruz ama burada değinilmek istenen bunlar değil. Aşağıdaki örnekte ne demek istediğimi daha net anlayabilirsiniz. Bu kod python’da hazırlanmıştır.

|  |  |
| --- | --- |
| 1  2 | a = "Hello"  a = a + 10; |

Hata mesajını canlı görüntülemek için koda bu [link](http://www.pythontutor.com/visualize.html#code=a%20%3D%20%22Hello%22%0Aa%20%3D%20a%20%2B%2010%3B&cumulative=false&curInstr=2&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=3&rawInputLstJSON=%5B%5D&textReferences=false) üzerinden de bakabilirsiniz.

Görüleceği üzere a değişkeni, **string** bir değer olmasına rağmen bir sayı ile toplanılmaya çalışılıyor. Sonuç olarak **TypeError: must be str, not int** şeklinde bir hata ile karşılaşıyoruz. Bu bize, her ne kadar dinamik bir tip olsa da çalışma zamanında içsel bir denetimin olduğunu gösterir. Gerçi java bu denetimi daha en başında yani derleme zamanında bize söyler. Bu yüzden;

* Java, static ve strong-typed bir dildir.
* Python ise dinamik ve strong-typed bir dildir.

**Zayıf Tip Dile Örnek**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#zay%C4%B1f-tip-dile-%C3%B6rnek)

Diğer taraftan **weakly-typed**(zayıf tip) bir dil, değişkenlerin belirli bir veri türüne bağlı olmadığı bir dildir; hala bir türü vardır, ancak tür güvenlik kısıtlamaları, güçlü yazılan dillere göre daha düşüktür. Örneğin javascript, tıpkı C programlama dili gibi zayıf yazılan dile bir örnektir. Her ne kadar C static tip bir dil, javascript ise dinamik tip bir dil olmasına rağmen her ikisi de **weakly-typed**(zayıf tip) bir dil olarak geçmektedir. Peki bu ne demek?

|  |  |
| --- | --- |
| 1  2 | var a = "5" + 2;  var b = 2; |

Canlı görüntülemek için koda bu [link](http://www.pythontutor.com/javascript.html#code=var%20a%20%3D%20%225%22%20%2B%202%3B%0Avar%20b%20%3D%202%3B&curInstr=0&mode=display&origin=opt-frontend.js&py=js&rawInputLstJSON=%5B%5D) üzerinden de bakabilirsiniz.

Görüleceği üzere a değişkeni bir **string** ve bir **sayı** ile toplanılmaya çalışılıyor. Sonuç olarak a değişkeni, **strong-typed** bir dil olan python’da olduğu gibi hata vermiyor ve “52” şeklinde bir sonucu karşımıza çıkıyor. Ama benzer şeyi python yapmamıza izin vermemişti. Çünkü tırnak içine aldığımız değeri python **string** olarak algılayıp ona göre bir işlem yapmaya çalışır. Bu yüzden bu tarz dillere **weakly-typed**(zayıf tip) diller denir.

Tam da referans ve nesne tipleri arasındaki farkı öğrenmişken bu tanımların da anlamlarını bilmeniz gerçekten önemlidir.

**Sözlük**[**Permalink**](https://hasancelik.org/java-kalitim-polimorfizm/Java-inheritance3_1/#s%C3%B6zl%C3%BCk)

* Bilgisayar bilimlerinde, dinamik dağıtım([Dynamic dispatch](https://en.wikipedia.org/wiki/Dynamic_dispatch)), hangi polimorfik işlemin çalışma zamanında uygulanacağını seçme işlemidir.
* Sınıf tabanlı programlamada, [downcasting](https://en.wikipedia.org/wiki/Downcasting) veya tür ayrıntılandırma(type refinement), bir temel sınıfın türetilmiş sınıflarından birine referans verme eylemidir.
* Geç bağlanma([late binding](https://en.wikipedia.org/wiki/Late_binding)), dinamik bağlanma veya dinamik bağlantı, bir nesneye çağrılan yöntemin veya bağımsız değişkenlerle çağrılan işlevin çalışma zamanında adıyla arandığı bir bilgisayar programlama mekanizmasıdır.