Java & Spring Bootcamp

Mustafa Güneş

Hakkımda

- Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği (2017)
- Yapı Kredi Bankası (Teknik Mimari)
- Scotty
- Craftbase (Trendyol Go)
- Midas

- https://twitter.com/mgunes17
- https://www.linkedin.com/in/mgunes17/

Takvim

- Başlangıç 18-19 Aralık 2021
- 1-2 Ocak ders yok
- Bitiş 5-6 Şubat 2022

- 10.00 - 11.00
- 11.00 - 11.15 mola
- 11.15 - 12.15
- 12.15 - 13.00 mola
- 13.00 - 14.00
- 14.00 - 14.10 mola
- 14.10 - 15.00

✴ (Küçük sapmalar olasılık dahilindedir.)

Kullanılacak Araçlar

- Intellij Idea 2021.3
  - Öğrenci maili / 30 günlük deneme sürümü / Community Edition
  - Eclipse, Netbeans, ...
  - Lombok ve Sonarlint eklentileri kurulması faydalı olur.
- openjdk 17.0.1 2021-10-19
  - 8+ büyük oranda yeterli olacaktır.
- Apache Maven 3.8.3
- Docker version 20.10.8

Eğitim Hakkında - 1

- Saçma soru yoktur, soru sormamak saçmadır.
- Anlatımlar ağırlık olarak kod üzerinden olacak.
- Kod yazmak ifadesi her zaman test yazmayı da kapsar.
- Güncel teknolojiler, pragmatik kod örnekleri ile anlatılacak. Amacımız problem çözmek.
- Bir konuya başlayınca baştan sona bitirilmeyecek. Gerektiği yerde gerektiği kadar öğreneceğiz. (Öğrenmek = neyi nerede nasıl neden kullanıyoruz)
- Kitaplardan özetler incelenecek (Ağırlıklı olarak Effective Java ve Clean Code). Çalışan kod yazmak ilk amacımız olsa da fırsatlarımız dahilinde kodu temiz ve verimli bir hale getireceğiz.
- Teorik kavramlar araştırma ödevi olarak verilecek, derslerde tartışılacak.
- Geride kaldığınızı, koptuğunuzu hissettiğiniz zamanlarda bana ulaşın. Buradaki herkes birebir aynı noktada değil, bitirirken de birebir aynı noktada olmayacak. Kendiniz için maksimum verimi almaya odaklanın.

Eğitim Hakkında - 2

- Eğitim sonunda ayrıca proje isteniyor olsa da aralarda da kodlama ödevleri olacak.
- Sunumlar derse yön vermek için var. Ayrıca kodlama detayına giremeyeceğimiz ama bilinmesi faydalı konular sunumlardan anlatılacak.
- Soru sormak için beklemeyin, araya girin sorun veya yazın. Sorular teknik bir konu hakkında olabileceği gibi sektör hakkında da olabilir.
- Bu bootcamp formal/akademik/sistematik bir eğitim değil. Bir grup Yazılım Mühendisinin potansiyel çalışma arkadaşlarıyla birlikte geliştiği bir süreç olacak.
- Interaktif eğitimin avantajlarından faydalanalım. Konuşalım, tartışalım, fikir yürütelim. Video eğitimlerin, kitapların, google bilgilerinin ötesine geçmek için çaba harcayalım.
- Not almanız tavsiye olunur. Not alma alışkanlığınız yoksa başlamak için güzel bir fırsat.

İçerik (Güncellenecek)

- **1. Hafta, Temel Konular;** Java, Object-Oriented Concepts, Veri Yapıları, Lombok, Java Streams, Lambda Functions
- **2. Hafta, Spring Framework;** Dependency Injection, Spring Core, Spring Boot, Spring Cloud
- **3.Hafta, Rest Mimarisi;** Rest Servisler, Alternatif İletişim Yöntemleri
- **4. Hafta, Spring Boot Data;** NoSql Solutions, İlişkisel veri tabanları, Transaction Yönetimi
- **5. Hafta, Security;** Spring Security, JWT
- **6. Hafta, Test Yazım Teknikleri;** Xunit patterns, mock-stub-fake, unit test, integration test, contract test
- **7. Hafta, Uygulama;** Spring Boot, Docker, Postgresql, Github actions

1. Bölüm

1.1 Java

Tarihine Hızlı Bir Bakış

- Sun Microsystems, 1982, Silikon Vadisi
- Oak ismi (Meşe ağacı), elektronik cihazlar için (başlıca interaktif televizyonlar); proje istenen başarıya ulaşamıyor.
- Web'in gelişimi, Java ismine evrilmesi (kahve), 1995, James Gosling ve ekibi
- İlk kararlı sürüm 1999, son kararlı sürüm 2021 (Java 17)
- Esinlendikleri; Smalltalk, C/C++, ...
- Java Community Process (JCP), 1998; standartların oluşmasına, dilin/teknolojinin gelişimine yön veren ekip
- Oracle'ın Sun Microsystems' ı satın alması

  https://webrazzi.com/2009/04/21/oracle-suni-satin-aldi/

- Mottosu: "Write once, run everywhere"

1.1 Java

Genel Özellikleri

- “Java: an Overview”,James Gosling, 1995
  - https://www.researchgate.net/publication/345758345_Java_an_Overview_the_original_Java_whitepaper
  - Simple, Object-Oriented, Distributed, Robust, Secure, Architecture Neutral, Portable, Interpreted, High Performance, Multithreaded, Dynamic
- **Compiled languages:** Yazılan kod “execute” edilebilen bir hale (makine dili) çevrilir. C/C++, GO, ...
  - **Interpreted Languages:** Yazılan kod çalışma anında “gerektikçe” makine koduyla eşleştirilir. PHP, Ruby, Python, ...
  - Just in time compilers
  - https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/
- Java Virtual Machine (JVM), byte codes
  - Java kodu “compile” olur, compile çıktısı olan byte kodlar JVM tarafından “interpret” edilir. (Aynı kod JVM olan herhangi bir ortamda çalışabilir. Spesifik ihtiyaçlar için mimari detaylara özel kod yazılması gibi durumlar dışında)
- Object Oriented, Java 8+ için fonksiyonel programlama desteği
  - Structred, edger Dijkstra
- **Statically typed languages:** değişkenlerin veri tipleri derleme anında bilinir, hatalar daha erken tespit edilebilir; Java, C/C++, Go, Scala, ...
  - **Dynamically typed languages:** değişkenlerin veri tipleri çalışma anında (runtime) bilinir; Ruby, Python, Javascript, ...
  - https://stackoverflow.com/questions/1517582/what-is-the-difference-between-statically-typed-and-dynamically-typed-languages

1.2 Fundamentals

Object Oriented Concepts

- Abstraction
- Class, Object
- Class = data + behaviour
- Encapsulation
- Access Modifiers
- Inheritance, Polymorphism
- Abstract Class, Interface
  - Composition
- Overloading, Overriding
  - Method signature

Data Structure

- Array
- List
  - ArrayList, LinkedList
- Map <key ,value>
- Set <key>
  - Key -> to remove duplication , retrieve o(1)
- Java Collections

1. Bölüm Sonu

Neler Yaptık ?

- Java Programlama Dili hakkında genel bilgiler
- Object Oriented Kavramlar
- Array, Map, Set, List veri yapıları

2. Bölüm

2.1 Lombok

Sık Kullanılanlar

- Getter
- Setter
- ToString
- NoArgsConstructor
- AllArgsConstructor
- RequiredArgsConstructor
- Data
- EqualsAndHashCode
- Builder

- Maven ile bağlılıkların yönetimi
- pom.xml' in incelenmesi
- mvn clean, mvn install, mvn clean install, mvn install -DskipTests=true
- .gitignore içerisinde neler var
- Semantic version
  - major.minor.bugfix
- Spring initialize
  - https://start.spring.io/

2.2 Effective Java'dan Alıntılar

Effective Java'nın Yaklaşımı

- Effective Java, Third Edition, Joshua Bloch
  - https://www.amazon.com.tr/Effective-Java-Joshua-Bloch/dp/0134685997
- Bir programlama dilini öğrenirken 3 ayrı şekilde uzmanlaşın.
  - **Grammer:** Dilin yapısı nasıl, syntax, OOP ve Fonksiyonel paradigmaları nasıl sağlıyor ?
  - **Vocabulary:** Core libraries, Data Structures, problem çözme şekli nasıl ?
  - **Usage:** Dilin yukarıdaki iki madde kapsamında sağladıkları günlük hayatında nasıl işine yarayacak, pragmatik olan nedir, community neyi nasıl sahiplenmiş ?
- Verilen kuralların/tavsiyelerin gözardı edileceği durumlar ile karşılaşabilirsiniz. Benimseyin ama çok sıkı sahiplenmeyin.
- Amaç performanslı kod yazmaya yönlendirmek yerine "clear, correct, usable, robust, flexible, maintainable" kod yazmaya yönlendirmek.
- İçerik, birbirilerine referans veren ama ayrı ayrı da okunabilecek 90 item.

2.2 Effective Java'dan Alıntılar

Item 1 - Consider static factory methods instead of constructors

- Static factory methods != factory method (in design patterns)
- Constructor'lara isim veremiyoruz. Fakat farklı construcxtor'ların farklı amaçları var.
- Aynı parametrelerle farklı işler yaptırmak constructor'lar ile mümkün değil. (Sebep aynı; isim veremiyoruz).
- Belki de gerçekten yeni obje yaratma ihtiyacımız yok. Static factory metodumuz gerekirse constructor kullanabilir.)
- Amacımıza uygun farklı bir değer return ettirmek isteyebiliriz. (Constructor her zaman ait olduğu sınıf tipinde bir obje oluşturur.)
- Dezavantajı; constructor içerisinde önce parent class'a ait constructor çağrılıyor. Bu işleyişten mahrum kalıyoruz.
- Tavsiye isimlendirmeler; from (type conversion with 1 parameter), of (type conversion with multiple parameters), valueOf, getInstance, newInstance, type, getType, newType

2.2 Effective Java'dan Alıntılar

Item 2 - Consider a builder when faced with many constructor parameters

- Kastedilen Design Pattern'larda yer alan Builder Pattern değil.
- Constructor ve Static Factory Method ortak bir dezavantaja sahip; parametre sayısının artınca scale olmazlar. (Kullanım sağa doğru uzar).
- Telescoping constructor (**anti-pattern**); gerekli tüm parametre kombinasyonlarıyla constructor'lar oluşturmak.
  - https://medium.com/@modestofiguereo/design-patterns-2-the-builder-pattern-and-the-telescoping-constructor-anti-pattern-60a33de7522e
- Builder kullanımı, kodu aşağı doğru uzatır.
  - Benzer analojiyi veri tabanı için de kullanabiliriz. Tablonun büyümesini column sayısını artırarak (sağa) değil, row sayısını artırarak(aşağı) görmek isteriz.
- **Avoid** construction with JavaBean style
  - Boş bir constructor çağırıp gerekli parametreleri tek tek atamak
  - https://stackoverflow.com/questions/22472018/why-java-builder-pattern-over-java-bean-pattern
  - Obje inconsistent bir halde kalabilir; immutable veya thread-safe yapmak zor olur.
- Parametre sayısı 4'ten az ise builder kullanımı tercih edilmeyebilir.

2.2 Effective Java'dan Alıntılar

Item 4 - Enforce non-instantiability with a private constructor

- Bazı class'lardan obje oluşturmak anlamsızdır.
  - Utility class'lar, amacı static field ve/veya metotları gruplamak olan class'lar.
- Default constructor'lar bu class'lardan yeni obje oluşturmanın yolunu açar.
- Abstract yapmak doğrudan obje oluşturmayı engeller anca dolaylı yoldan o sınıfı extend etmeyi engellemez.
  - (Bir seçenek class final yapılabilir.)
- Basitçe private bir constructor ekleyip obje oluşturma ihtimalini ortadan kaldırabiliriz.
  - Kodu okuyan yazılımcıya bilgi verme amaçlı olarak private constructor içerisinde Runtime exception fırlatılabilir.
  - (Bunu yapmadığımızda sonarlint uyarı veriyor.)

2.2 Effective Java'dan Alıntılar

Item 40 - Consistently use the Override annotation

- *Koddaki Bigram class'ı örneği*
- Override ettiğini sandığın halde override edemiyor olabilirsin.
- @Override kullan compiler seni uyarsın.
  - Abstract metotlar için anotasyon koymasan bile compiler uyarır. (Abstract metotlar override edilmelidir.)

2.2 Effective Java'dan Alıntılar

Item 11 - Always override HashCode when you override equals

- 2 objenin eşit olması için HashCode'larının eşit olması gerekir.
  - Fakat HashCode'ları eşit olan 2 obje her zaman aynı olmayabilir. (Collision durumları), fakat farklıysa her zaman farklıdır.
- equals metodu override edilirken hangi field'lar kullanıldıysa, hashCode metodu override edilirken de o field'lar kullanılmalıdır.
- Equals ve HashCode'un birlikte ve doğru override edilmesiyle Map, Set gibi Collection'lar doğru çalışır.
- Lombok @EqualsAndHashCode
  - https://projectlombok.org/features/EqualsAndHashCode
- HashCode hesaplanan bir değer olduğu için değişebilir. Immutable class'larda değişmez.
  - Değişmediği durumda hesaplama maliyeti projenizin performans beklentilerini sekteye uğratıyorsa cache'lenebilir.

2.3 Java 8 ve sonrası

Göze çarpan değişiklikler

- Stream API, java.util.Stream
  - Stream = sequence of data items
  - Stream'in farklı bölümleri farklı cpu core'lar üzerinde çalışabilir. (Kodu paralelleştirmek kolaylaştı)
- lambda function, method reference
  - Behaviour parameterization = davranışı parametrik hale getirme (anonymous class'lar yerine)
  - Nasıl ki user input değişecek diye metotlara sabit değer koymayarak parametre alır halde yazıyoruz, value'ler gibi "behavior" lar da değişebilir. Kodumuz daha esnek olur.
  - First class citizen (functional programming style)
- Predicate<P> predicate = Return değeri Boolean olan ve parametre olan P tipinde değer alan bir lambda function veya metot assign edilebilir.
- Optional<T> = to avoid null pointer exceptions
  - Anekdot; Tony Hoare, 1965, invention of null reference, "my billion dollar mistakes"
- default methods in interfaces

2.3 Java 8 ve sonrası

Lambda Functions

- Lambda calculus, Alonzo Church, 1930'lar
- Lambda expression özellikleri
  - İsmi yok (anonim)
  - Function (metot değil, side effect'i yok)
  - Bir değişkene atayabiliriz veya bir metoda parametre olarak geçebiliriz.
  - Kısa ve sadedir.
- (Lambda parameters) -> { lamda body};
- Method reference
  - Daha okunabilir kod
  - Bazı lambda ifadeleri, metot reference olarak yazılabilir.

2.4 Effective Java 'dan Alıntılar

Chapter 7 - Lambdas and Streams

- **Item 42:** Prefer Lambdas to anonymous classes
  - Boilerplate kodu azaltır.
- **Item 43:** Prefer method references to lambdas
  - Shorter + clearer code
- **Item 44:** Favor the use of standard functional interfaces
  - İşini görecek functional interface'ler varsa kendin yazma. Yazarsan da @FunctionalInterface'i ekle
- **Item 45:** Use streams judiciously
  - Bazen basit bir for stream'dan daha anlaşılır olabilir. (Bazen)
- **Item 48:** Use caution when making streams parallel
  - Gerçekten ihtiyacın olana kadar stream'ler için .parallel() kullanma. Kullandığın takdirde çok iyi test.

2. Bölüm Sonu

Neler Yaptık ?

- Spring initializr
- Maven ile proje yönetimi, pom.xml
- Sık kullanılan Lombok anotasyonları
- Java streams, lamda function, method reference
- Effective Java Item 1, 2, 4, 40, 11 ; 42, 43, 44, 45, 48

2. Bölüm Sonu

Ödev 1 /Araştırma Soruları

1. Pass by value, pass by reference kavramları nedir ? Java' ile ilişkili olarak açıklayınız.
2. Immutability nedir, neden önemlidir ? Bir Java sınıfı nasıl immutable yapılır ?
3. Framework ve library arasındaki fark nedir ?
4. Java'da Garbage Collector' un görevi nedir ?
5. Memory leak nedir ? Java'da memory leak oluşması mümkün müdür ?
6. Yeni Java sürümleri ne sıklıkla çıkmaktadır ?
7. Stack ve Heap nedir ? Java'da hangi yapılar stack ile, hangi yapılar heap ile ilişkilidir ?
8. OpenJDK ve OracleJDK arasındaki farklar nelerdir ?
9. @FunctionalInterface anotasyonu nerelerde kullanılabilir, neleri sağlar ?
10. Java'da hangi functional interface'ler yer almaktadır ? Yaptığınız araştırmada en popüler/göze çarpanlar hangileridir ?

2. Bölüm Sonu

Ödev 2 /Kodlama

- https://martinfowler.com/articles/collection-pipeline/
- Beklenenler (yukarıda yer alan yazıdan)
  - Ana başlıkların birkaç cümleyle özetinin çıkarılması
  - Spring initializr ile bir proje oluşturulması; Lombok ve spring-boot-starter-test dependency' lerinin eklenmesi
  - Yazıda yer alan operatörlerin Java'daki karşılıklarının bulunup her biri için birer örnek metot yazılması (bazı operatörlerin birebir karşılığı olmayabilir, küçük dokunuşlar yapmak durumunda kalabilirsiniz)
  - (Bonus) Metotlara Junit ile test yazılması