1. Cevap C. Verilen kod örneğinde 2 satırda derleme hatası vardır. main() metodunda Movie sınıfından bir nesne üreterek bir üst sınıfın name isimli özelliğine erişilmek istenmektedir. Burada name değişkeni protected olmalıdır ki, alt sınıftan erişilebilsin. İkinci derleme hatası ise, Movie sınıfı kurucu metodunda üst sınıfın kurucusuna çağrı yapılmamasından kaynaklanacaktır. Çünkü, Cinema sınfının argümansız kurucu metodu yoktur. Bu nedenle Movie sınıfı kurucu metoduna super() kurucu metot çağrısı eklenmelidir.
2. Cevap D. Soyut bir interface'in metoduna public erişim belirleyicisi uygulanabilir. final, static ya da protected olamaz. Çünkü, bu interface'den implemente edilen sınıflar farklı paketlerde yer alabilir. interface'ler şablondur. Bellekte bir tane olmayacaklar için final ya da static olamazlar.
   Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html
   https://www.tutorialspoint.com/can-we-declare-the-variables-of-a-java-interface-private-and-protected
3. Cevap C. Verilen kod örneğinde ilk playMusic() metodu derlenir ve main() metodunda üretilen bir nesne referansı ile çağrılarak çalıştırılabilir. Ancak ikinci playMusic() metodunun dönüş int olarak verilmiştir. Herhangi bir return yapmayan bu metottan dolayı kod derlenmez. İkinci metodun dönüş tipi olan int ifadesine yerine void yazılarak ya da metoda herhangi int verisi dönen bir return ifadesi eklenirse kod derlenir hale gelecektir. Ayrıca sınıfın bir nesne referansı ile çağrı yapıldığı için bu metodun static olmasına gerek yoktur. static ifadesi silinebilir.
4. Cevap A. Kalıtımın genel amacı üst ve alt sınıflar oluşturarak kodu yapısal hale getirmektedir. Bu bağlamda oluşturulan üst sınıflar genel özellikleri barındırırken bu üst sınıftan genişletilen alt sınıflar ve varsa bu alt sınıfların alt sınıfları üst sınıfın tüm özellikleri yanı sıra kendine has özellikleri taşır. Böyle kodun yeniden kullanılabilirliği ve okunurluğu artar. Tekrar eden kodların yazılması önlenir. Bu nedenle A seçeneği doğrudur. Kalıtımda yapılandırmaya bağlı olarak bir kod daha basit bir hale ya da daha karmaşık bir hale gelebilir. Bu tamamen yapılandırmayla ilgili bir durumdur ve B seçeneği doğru olmayabilir. Object sınıfından gelen nesneler için metotlar kalıtılabilir ancak ilkel tipler için kalıtım söz konusu olmadığından dolayı C seçeneği yanlıştır. Kalıtımda kendilerini referans olan yöntemlere doğru implementasyon yapılmaz. Kalıtımın yönü tam tersidir. D seçeneği yanlıştır.
   Kaynak: https://www.geeksforgeeks.org/inheritance-in-java/
5. Cevap A. Verilenlerden Pet interface'i ve Pet interface'inden implemente edilen Canine sınıfı türü dönüş değeri olabilir. Ayrıca, java.lang paketinde yer alan Object sınıfı türüne izin verilir. Çünkü, tüm sınıflar Object sınıfından kalıtılır. Bu nedenle dönüş tipi olarak verilemeyecek ifade Class ifadesidir.
   Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
6. Cevap B. Java'da interface kavramı şablon veya arayüz anlamında kullanılmaktadır. Bir kalıp içerisinde tanımlanan metot ve özelliklerin, buradan implemente edilen soyut ve soyut olmayan sınıflarda uygulanması amacıyla interface'ler yazılım tasarımında kullanılmaktadır. Bu kapsamda büyük ekiplerle çalışılan durumlarda interface tasarımları ile diğer geliştiriciler hazır olmayan metot ve sınıfların tasarımı konusunda anlaştıklarında bunlara uygun interface'ler sayesinde kendi uygulamalarını rahatça geliştirebilirler. Bu nedenle cevap B seçeneğidir.
   Kaynak: https://www.tutorialspoint.com/what-is-the-purpose-of-interfaces-in-java
   https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html
   http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch17/interface.htm
   https://www.injavawetrust.com/pure-java-03-interface-declaration-3/
   http://www.kurumsaljava.com/download/68/
7. Cevap B. Derleyici işlemleri yaparken, subclass türünü superclass türüne dönüştürür ve yapılabilecek işlemleri superclass' ın özellikleriyle sınırlandırır. main() metodunda Car türünden

bir nesne oluşturulurken alt sınıf olan ElectricCar sınıfının kurucu metodu ile bellekte alan açılmaktadır. ElectricCar sınıfında yer alan drive() metodunun çağrılması ile ekran çıktısı "Driving electric car" şeklinde olacaktır.
Kaynak: https://www.injavawetrust.com/pure-java-11-object-orientation-02-polymorphism-polimorfizm/
https://www.mobilhanem.com/java-polymorphism/

8. Cevap D. Java bir sınıfı birden fazla sınıfın extend etmesine izin vermez. Fakat bir sınıf birden fazla interface ile implemente edilebilir. Bu nedenle A, B ve C seçenekleri yanlış; D seçeneği doğrudur.
Kaynak: https://www.geeksforgeeks.org/java-and-multiple-inheritance/
https://docs.oracle.com/javase/tutorial/java/IandI/multipleinheritance.html
9. Cevap C. Television sınıfındaki watch() metodunun final olmasından dolayı bu metot override edilemez. Metot overriding yapılacak alt sınıftaki metodun erişim belirleyicisi üst sınıfta yer alan metodun erişim belirleyicisinden daha dar olamaz. Üst sınıfın watch() metodu protected olduğuna göre alt sınıftaki watch() metodu public olabilir. Ayrıca metotların dönüş tipleri aynı olmalıdır. Burada ya her iki metodun dönüş tipi de Object yapılır ya da void yapılarak dönüş tipi iptal edilir. Object yapılırsa return ifadeleri eklenir. Bu şekilde toplamda 3 değişiklik yapılarak kod derlenir hale getirilir.
Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/override.html
https://www.javatpoint.com/method-overriding-in-java
10. Cevap C. Metot overriding'de metotların dönüş tipleri kovaryant olmalıdır. Alt sınıftaki metodun erişim belirleyicisi üst sınıftaki metodun erişim belirleyicisinden daha geniş olmak zorundadır.
Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/override.html
http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch15/overriding.htm
https://omerozkan.net/java-override-anotasyonu/
https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/overriding-(ge%C3%A7ersiz-k%C4%B1lmak)
11. Cevap C. Üst sınıf olan Computer sınıfının process() metodunun final olmasından dolayı Laptop sınıfında override edilemez. Bu nedenle bu satırda derleme hatası vardır ve kod derlenmez. Eğer final değiştiricisi silinirse ekran çıktısı 3 olacaktır.
Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/override.html
12. Cevap A. School sınıfında yer alan getNumberOfStudentsPerClassroom() metodu alt sınıf olan HighSchool sınıfında override edilmektedir. main() metodunda üretilen school nesnesine HighSchool olarak bellekte yer açılır. Dolayısıyla HighSchool sınıfında override edilen getNumberOfStudentsPerClassroom() metodu 2 değeri ile döner ve ekrana yazılır.
13. Cevap B. Verilenlerden static belirleyicisi bir interface metoduna uygulanabilir. static olarak implemente edilen metot, interface içinde metot imzası ile değil gövdesi implemente edilir. Diğer seçeneklerde yer alan final, private ve protected belirleyicileri interface metotlarına uygulanamaz.
Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/interfaceDef.html
http://tutorials.jenkov.com/java/interfaces.html#interface-static-methods
14. SORU HATALIDIR. Verilen kod derlenir ancak geçerli bir main() metodu olmadığı için uygulamaya giriş noktası yoktur. Verilen main() metoduna komut satırı argümanı eklenirse C seçeneğindeki "Sprinting!" ifadesi ekrana basılır. Çünkü, Sprint sınıfından bir örnekleme yapılmaktadır ve override edilen metot çağrılmaktadır.

15. Cevap B. Bir interface bir ya da daha fazla interface'i extend edebilir ancak implemente edemez. Bir sınıf da bir ya da daha fazla interface'i implemente edebilir. Bir sınıf başka bir sınıfı da genişletebilir.Bu bilgilere göre A, C ve D seçenekleri doğrudur. B seçeneği doğru değildir.
    Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/usinginterface.html
    https://www.geeksforgeeks.org/interfaces-and-inheritance-in-java/
16. Cevap D. Verilen kodda Rocket sınıfı Ship sınıfını genişletmektedir. printDetails() metodunda super anahtar sözcüğü ile üst sınıfın özelliklerine erişilmeye çalışılmaktadır. Ancak, super.height ile erişilmek istenen değişkenin private olması nedeniyle sadece kendi sınıfından erişime izin verir. Bu ifadeden dolayı kod derlenmez. Burada çözüm olarak height özelliği veren getHeight() metodu kullanılabilir ya da bu özelliğin erişim belirleyicisini protected, package-private ya da public yaparak kodu derlenir hale getirilebilir. Kod derlenir hale geldiğinde çıktısı "3,5" olacaktır.
    Kaynak: https://www.geeksforgeeks.org/super-keyword/
    https://www.geeksforgeeks.org/variable-scope-in-java/
17. Cevap D. Soyut bir sınıf hem soyut hem somut metotlar içerebilir. Ancak interface'ler sadece soyut metotlar içerebilir. Java 8 ile beraber interface'ler gövdesi olan default ve static metotlar içerebilir hale getirilmiştir. Soruda default ve static metotların hariç tutulması nedeniyle doğru cevap D seçeneği olmaktadır. İlk boşluğa soyut sınıf, ikinci boşluğa interface gelecektir. Somut sınıflar ise soyut metotlar içermezler.
    Kaynak: https://www.baeldung.com/java-static-default-methods
    https://www.geeksforgeeks.org/difference-between-abstract-class-and-concrete-class-in-java/
    https://beginnersbook.com/2017/10/java-8-interface-changes-default-method-and-static-method/
    https://www.injavawetrust.com/pure-java-03-interface-declaration-3/
18. Cevap C. IsoscelesRightTriangle sınıfı soyut bir sınıftır ve soyutlar new anahtar sözcüğü ile referans üretmezler. Bu nedenle g3 numaralı satırdan dolayı kod derlenmez.
19. Cevap D. Verilen kod örneğinde Saxophone sınıfı WoodWind soyut sınıfını genişletmekte ve Horn interface'ini implemente etmektedir. Dolayısıyla hem Horn interface'inde imzası verilen play() metodu implemente edilmeli hem de WoodWind soyut sınıfının play() metodu override edilmelidir. Metot imzalarının aynı olması nedeniyle dönüş türleri aynı olmadığı sürece Saxophone sınıfındaki metodun dönüş türüne ne yazılırsa yazılsın kod derlenmez. Dönüş tipleri aynı olduğu durumda kod derlenir hale gelecektir. Doğru cevap D seçeneğidir.
20. Cevap C. Bir sınıf, bir interface'i implements anahtar sözcüğü ile uygular; bir soyut sınıfı extends anahtar sözcüğü ile genişletir. Boşluklara sırasıyla implements ve extends ifadeleri geleceği için doğru cevap C seçeneğidir.
    Kaynak: https://howtodoinjava.com/oops/extends-vs-implements/
21. Cevap A. Encyclopedia sınıfında kurucu metot, üst sınıfın kurucu metoduna çağrı yapmaktadır. Book sınıfındaki material özelliği "papyrus" değeri yüklenmektedir. main() metodunda Encyclopedia sınıfının getMaterial() metodu çağrılmaktadır. Bu metot içerisinde de super sözcüğü ile üst sınıfın material özelliğindeki değer getirilmektedir. "cellulose" değerini alabilmek için this.material şeklinde değer getirilmelidir.
22. Cevap B. unknownBunny referansı Bunny sınıfından ya da üst bir sınıftan nesne referansı ise A ve C seçenekleri doğru olacaktır. Ayrıca unknownBunny referansı bir interface, sınıf ya da soyut sınıftan da olabilir. Ancak, Bunny sınıfı referansına casting olmadan atanabiliyorsa kesinlikle Bunny sınıfının alt sınıfı değildir. Bu nedenle B seçeneğinde verilen bilgi yanlıştır.
23. Cevap D. abstract bir metodun erişim belirleyicisi public ya da protected olabilir. Diğer seçeneklerde verilenlerden default sözcüğü sadece interface metotları için geçerlidir. final

sözcüğü ile yapılırsa override edilemez, private yapılırsa farklı sınıftan erişilemeyeceği için bu soyut sınıfı genişleten somut sınıfta override edilemez. Bu nedenle final, private ve default sözcükleri soyut bir metot deklare edilirken uygulanamaz.
Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html
https://www.geeksforgeeks.org/abstract-methods-in-java-with-examples/

24. Cevap D. Mars sınıfının deklare edildiği satırda, Planet soyut sınıfı implements sözcüğü ile uygulanmaya; Sphere interface'i extends sözcüğü ile genişletilmeye çalışılmaktadır. Buradaki mantıksal hatadan dolayı kod derlenmez. Burada sınıf deklarasyonunu "public class Mars extends Planet implements Sphere { ... } " olarak düzeltirsek kod derlenir hale gelir ve "Mars" şeklinde ekran çıktısı üretir.
25. Cevap B. Alt sınıf referansı, üst sınıf referansına doğrudan atanabilir. Ancak tersi durumda, üst sınıf referansı alt sınıf referansına atanırken casting yapılması gerekecektir. Bu nedenle A seçeneği yanlış, B seçeneği doğrudur. Bir interface bir sınıftan miras almaz. Bu nedenle C seçeneği yanlıştır. Bir interface'i implemente eden bir sınıf referansı, bu interface'in bir referansına atanırken casting'e gerek duymaz. Bu nedenle D seçeneği yanlıştır.
Kaynak: http://web.deu.edu.tr/doc/oreily/java/langref/ch03_02.htm
https://docs.oracle.com/javase/tutorial/java/IandI/interfaceAsType.html
https://www.tutorialspoint.com/can-we-cast-an-object-reference-to-an-interface-reference-in-java-if-so-when
26. Cevap B. Bir interface içinde tanımlanan değişken otomatik olarak public, final ve static belirleyicileri ile tanımlanır. Ancak abstract olamaz.
Kaynak: http://knowledgehills.com/java/java-interface-variables-default-methods.htm
27. Cevap C. new BlueCar() yapılarak bu sınıfın kurucu metodundan bir örnekleme yapılacaktır. BlueCar sınıfı kurucu metodu super() çağrısı ile üst sınıfın kurucu metodunu çağıracaktır. Car sınıfının kurucu metodundan önce static initialization bloğu çalışarak 1 çıktısını üretir. main() metodunda BlueCar nesnesi yaratıldığında Car sınıfı ilk kez yüklenmiş olur ve önce instance initialization bloğu ve peşinden kurucu metodun çalışmasıyla 3 ve 2 değerleri basılır. BlueCar örneği yaratılırken önce instance initialization bloğu icra edilir, sonra kurucu metoda devam eder ve sırayla 4 ve 5 değerleri basılır. Ekran çıktısı "13245" şeklinde olacaktır.
28. Cevap C. Override ve overload metotların isimleri aynı olmalıdır.
Kaynak: https://www.programcreek.com/2009/02/overriding-and-overloading-in-java-with-examples/
29. Cevap A. main() metodunda Equipment sınıfıından bir nesneye SoccerBall sınıfından üretilen bir nesne atanmaktadır. SoccerBall sınıfının get() metodu çağrıldığında this ifadesi ile mevcut sınıfın örneği alınır. SoccerBall sınıfı kurucu metodu super() ifadesi ile üst sınıfın kurucu metoduna 5 değerini göndermektedir ve size değeri 5 olarak ayarlanır. Böylece üst sınıfın özelliği set edilmektedir. main() metodunda sonraki satırda da bu değer ekrana basılır.
30. Cevap C. Verilen tanımlar sırasıyla değişken gizleme ve metot gizleme tanımlarıdır. Boşluklara gizleme(hiding) yazılmalıdır.
Kaynak: https://www.baeldung.com/java-variable-method-hiding
31. Cevap B. x1 numaralı satırdan dolayı kod derlenmez. Rectangle sınıfındaki getEqualSides() metodu static olarak implemente edildiği için override edilemez. Square sınıfı soyut bir sınıftan türetildiği için üst sınıftan override ettiği metot static olduğu için bu noktada da metot static olmalıdır. Bu sağlanırsa 4 değeri ekran çıktısı olarak basılır.
32. Cevap C. Somut(concrete) bir sınıf soyut(abstract) bir metot içermez. Bu nedenle öncelikle Rotrcraft sınıfı derlenmez. Rotorcraft sınıfı somut olarak kalmaya devam ederse, bu sınıftaki fly() metodunu somut hale getirip bir gövde implemente edilirse bu defa da main() metodu

içinde Helicopter nesnesine Rotorcraft sınıfından bir örnek atanmak istendiğinde ClassCastException alınır.
Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/ClassCastException.html

33. Cevap B. Java'da bir sınıf referansı üst sınıf referansına doğrudan atanabilir ancak tam tersi durumda casting yapılması gerekmektedir. Bu nedenle ilk boşluğa üst sınıf(superclass), ikinci boşluğa(subclass) yazılmalıdır.
Kaynak: https://www.tutorialspoint.com/what-happens-when-a-subclass-object-is-assigned-to-a-superclass-object-in-java
34. Cevap C. Somut(concrete) sınıf, soyut sınıfları genişleten ve interface'leri uygulayan ve kendisi soyut olmayan sınıftır. Verilen tanım somut(concrete) sınıf tanımıdır.
Kaynak: https://www.geeksforgeeks.org/concrete-class-in-java/
35. Cevap D. Verilen kod örneğinde CanFly interface'inde verilen fly() metodunun gövdesi olamaz. Java 8 ile beraber interface'lerde sadece default ve static metotların gövdesi olabilir. Bird sınıfı final yapılmıştır ve kalıtım yoluyla alt sınıflar oluşmasına izin verilmez. Bird sınıfında verilen fly() metodu int tipinde dönüş yapmalıdır ve return ifadesi olmadığı için burada da bir derleme hatası olacaktır. Toplamda 3 tane derleme hatası vardır.
36. Cevap B. Soyut sınıflarda default metot tanımlanamaz. Buna göre interface ve soyut sınıflar arasında ortak olmayan özellik her ikisinde de default metot olmamasıdır.
Kaynak: https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html
37. Cevap C. Performance sınıfında tanımlanan talk() metodu String türünde bir diziyi argüman olarak aldığı için SpeakDialogue ya da SingMonologue sınıflarında yer alan talk() metotlarını override etmemektedir. Bu interface'leri uygulayabilmesi için bu iki interface'den birinin talk() metodu String türünden parametre alacak şekilde güncellenmelidir.
38. Cevap A. Sanal(virtual) metot, bir metodun nasıl icra edileceğinin derleme zamanında değil, çalışma zamanında belirlendiği metottur. Bir metot final olursa, sabit(constant)tir ve değiştirilemez. private erişim belirleyicisi ile kapsüllenirse başka bir sınıftan erişilemez. static yapılırsa sınıf metodu haline gelir. static metotlar, override edilmez, gizlenir(hide). Bu nedenle sanal(virtual) metot denildiğinde, non-final, non-static ve non-private metot ifadesi akla gelir. Bu nedenle, seçeneklerden verilenlerden protected instance metotlar sanal metottur.
39. Cevap B. Bir sınıf, başka bir sınıfı extends anahtar kelimesi genişletir. Bir interface, başka bir interface'i extends anahtar kelimesi genişletir.
40. Cevap A. Verilen kod örneğinde main() metodu için InfiniteMath sınıfından oluşturulan nesne referansı iki üst sınıfı olan Math sınıfı nesne referansına atanmaktadır. math nesnesinin secret özelliğinde double türünden final olarak yüklenen değer 2'dir ve ekran çıktısı 2.0 olur.
41. Cevap D. Soruda verilen metodu override edebilecek metot için öncelikle erişim belirleyicisine(access modifier) bakılır. Sorudaki metot protected olduğu için override edecek metot en az protected olabilir, package-private ya da private olamaz. A ve C seçenekleri yanlıştır. B seçeneğindeki metot FileNotFoundException istisnasının üst sınıfından bir hata fırlattığı için override koşuluna uymamaktadır. FileNotFoundException istisnasının ya aynısını fırlatır ya da herhangi bir istisna fırlatmaz. B seçeneği de yanlıştır. Override eden metodun final olması durumu override koşullarını etkilemediği için doğru cevap D seçeneğidir.
Kaynak: https://www.tutorialspoint.com/Rules-for-Java-method-overriding
42. Cevap C. Kodda boş bırakılan yere Wolf sınıfından bir nesne parametre olarak verilemez. Zoologist sınıfında yer alan setAnimal() metodo Dog sınıfı türünden bir nesne referansını parametre olarak alır. Husky sınıfı, Dog sınıfının bir alt sınıfı olduğundan uygulanabilir. Dog sınıfından bir nesne referansı da verilebilir, null parametre de verilebilir. Canine sınıfı Dog sınıfının üst sınıfı olduğundan dolayı Canine sınıfının bir nesne referansını verebilmek için Dog sınıfına casting yapmak gerekecektir.

43. Cevap A. Bir interface metodu, Java 8 ile beraber static veya default olabilir ve gövdeye sahip olabilir. Aynı zamanda abstract deyimi ile soyut metot olarak da deklare edilebilir. Ancak, final olamaz.
    Kaynak: https://www.baeldung.com/java-static-default-methods
44. Cevap A. Soyut sınıflarla oluşturulan kod parçası sorunsuz şekilde çalışır ve çalışma zamanında "Let's start the party!" yazar.
45. Cevap D. Farklı parametre listesi metotların implemente edilmesi metot overloading ile ilgilidir. İkinci kısım metot overriding ile ilgili gibi görünse de, metot overriding'de metotların dönüş tipleri aynı olmak zorunda değildir. Birbirinin yerine geçebilen tipler yeterlidir. Bu nedenle cevap D seçeneği olacaktır. Kaynakta, overriding kuralları yer almaktadır.
    Kaynak: https://www.tutorialspoint.com/java/java_overriding.htm
46. Cevap B. Üst sınıf, argümansız bir yapıcı metot içermese de alt sınıf deklarasyonu yapılabilir. A seçeneği yanlıştır. Üst sınıf, argümansız bir yapıcı metot içermiyorsa, alt sınıf en az bir tane kurucu metot içermelidir. B seçeneği doğrudur. Üst sınıfın argümansız kurucu metot içermesi alt sınıfın herhangi bir kurucu metot tanımlamasını zorunlu kılmadığı için hem C hem D seçenekleri yanlıştır.
    Kaynak: https://www.geeksforgeeks.org/constructors-in-java/
47. Cevap D. Nesne tipi var olan nesnenin özellikleriyle ilgilenirken, referans tipler ise nesneyi çağıran tarafından nasıl kullanılacağını belirler.
    Kaynak: https://www.coursera.org/lecture/object-oriented-java/core-reference-vs-object-type-h75Dw
    https://www.oreilly.com/library/view/java-8-pocket/9781491901083/ch04.html
48. SORU HATALIDIR. Verilenlerden sadece Long dönüş tipi yazılabilir. Çünkü, Long sınıfı Number sınıfının bir alt sınıfıdır. Integer yazılamaz. Çünkü, StringInstrument sınıfındaki metodu override edebilmesi için kapsayabileceği bir türde olmalıdır. Ancak soruda verilen kod örneği olduğu gibi yazıldığından derlenmez. Çünkü, return ifadesinde yer alan sayının wrapper sınıfı Integer'dır ve Long'a cast edilmesi gerekir. Bu yapılmadığı için kod derlenmez.
49. Cevap B. Java 8 ile beraber interface'lere getirilen default metot kavramı, interface'lere geriye dönük uyumluluk kazandırmıştır. C ve D seçeneklerinde verilen bir interface'i sınıf gibi kullanma amacına yönelik olarak somut metot oluşturma ve sınıf seviyesinde metot tanımlanmasına izin verme amaçları static metotlarla sağlanmıştır.
    Kaynak: https://docs.oracle.com/javase/specs/jls/se8/html/jls-13.html#d5e19889
    https://beginnersbook.com/2017/10/java-8-interface-changes-default-method-and-static-method/
    https://www.geeksforgeeks.org/static-method-in-interface-in-java/
    https://www.journaldev.com/2389/java-8-features-with-examples
50. Cevap C. Verilen kodda Machine sınıfındaki turnOn() metodu EOFException fırlatmaktadır. IOException bu exception sınıfının bir üst sınıfı olduğuna göre alt sınıftaki override edilen metotta ya EOFException fırlatılmalıdır ya da Machine sınıfındaki metot IOException fırlatmalıdır. Overiding kuralına uymayan bu implementasyon geçersiz olduğundan dolayı kod derlenmez.
    Kaynak: https://www.geeksforgeeks.org/overriding-in-java/