1. Cevap D. Verilen kod örneğinde try bloğunda bir Exception fırlatılmaktadır. finally bloğu implemente edilmedeği için kod derlenmez. Try bloğu varsa finally bloğunun mutlaka çalışması gerekir. Bu nedenle finally implemente edilmelidir.
   Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html
   https://docs.oracle.com/javase/tutorial/essential/exceptions/finally.html
2. Cevap B. Genel olarak try bloğunda yakalanan exception catch bloğunda handle edilir. En son olarak da exception finally bloğuna aktarılır. finally bloğu icra edilir.
   a. try-catch ifadesinde akış veya try-catch-finally ifadesinde akış.
      1) try bloğunda exception yakalanır ve catch bloğunda handle edilir.
      2) try bloğunda yakalanan exception catch bloğunda handle edilmez.
      3) try bloğunda exception yakalanmaz.
   b. try-finally ifadesinde akış.
      1) try bloğunda exception yakalanır.
      2) try bloğunda exception yakalanmaz.

   Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/handling.html
   https://www.geeksforgeeks.org/flow-control-in-try-catch-finally-in-java/
3. Cevap D. Throwable sınıfı Object sınıfının alt sınıfıdır. Error ve Exception sınıfları Throwable sınıfının alt sınıflarıdır. RuntimeException sınıfı da Exception sınıfının alt sınıfıdır. Bu nedenle doğru gösterim D seçeneğindeki gibidir.
   Kaynak: https://docs.oracle.com/javase/7/docs/api/java/lang/package-tree.html
4. Cevap A. B seçeneğinde verilen türden bir JAVA API'sinde exception sınıfı yoktur. Exception ve RuntimeException sınıfları Java uygulamalarında yakalanabilirken, Error sınıfından bir exception'ın yakalanması önerilmez. Bu nedenle doğru cevap A seçeneğidir.
   Kaynak: https://docs.oracle.com/javase/7/docs/api/java/lang/package-tree.html
   https://docs.oracle.com/javase/8/docs/api/java/lang/Error.html
5. Cevap D. Verilen kod derlenmez. Çünkü, try bloğunda deklare edilen score isimli değişkene catch bloklarından ve finally bloklarından erişilemez. score değişkenini try bloğu dışında deklare edersek sonuç A seçeneğindeki gibi "123" şeklinde olacaktır.
6. Cevap B.
   Kaynak: https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/
   https://www.codejava.net/java-core/exception/java-checked-and-unchecked-exceptions
7. Cevap A. Bir metot için exception deklarasyonu yapılacağında metot imzasından throws anahtar sözcüğü kullanılır. Kod içerisinde exception fırlatılmak istenildiğinde throw anahtar sözcüğü kullanılır.
   Kaynak: https://www.geeksforgeeks.org/throw-throws-java/
   https://www.javatpoint.com/difference-between-throw-and-throws-in-java
   https://www.tutorialspoint.com/difference-between-throw-and-throws-in-java
   https://www.baeldung.com/java-throw-throws
8. Cevap B. IOException sınıfı, Exception sınıfının subclass'dır. Bu nedenle öncelikle IOException için bir catch bloğu implemente edilmelidir. Bu yüzden doğru cevap B seçeneğidir.
   Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/catch.html
   https://docs.oracle.com/javase/8/docs/api/java/lang/package-tree.html
9. Cevap D. Verilen kod örneğinin catch bloğunda throw sözcüğü ile bir hata fırlatılması düşünülmektedir. Ancak t değişkeni deklare edilmediği için kod derlenmez. Söz dizimi hatasına rağmen çalıştırmak istersek "throw t;" satırında java.lang.Error sınıfından hata fırlatır.

10. Cevap C. p3 numaralı satırdan dolayı kod derlenmez. Çünkü, openDrawbridge metodu Exception sınıfından bir exception fırlatmaktadır. Bu nedenle p3 numaralı satırdaki kod ya try-catch bloğuna alınmalıdır ya da main() metodu "throws Exception" ifadesi ile imzalanarak exception fırlatacağı bildirilmelidir.
    Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html
11. Cevap B. Verilenlerden NullPointerException ve ArithmeticException her ikisi de RuntimeException'ın alt sınıflarıdır ve bu unchecked exceptionların fırlatıldıkları yerde deklare edilmesi ya da handle edilmesi gerekmez. Ancak Exception, bir checked exceptiondır ve fırlatıldığı yerde deklare edilip handle edilmelidir.
    Java'da sıradışı exceptionlar ikiye ayrılır: Checked Exceptions ve Unchecked Exceptions.
    Exception sınıfının alt sınıfı olup RuntimeException olmayan sınıflari Checked Exception olarak adlandırılır.
    RuntimeException ve alt sınıfları ise Unchecked Exception olarak adlandırılır. Error sınıfı da bir Unchecked Exception'dır.
    - Checked Exceptionlar ya yakalanmalı ya da metot arayüzünde throws ile fırlatıldığı belirtilmelidir. Eğer bir kod parçası checked exception fırlatıyorsa bu durumda iki seçenek vardır, ya try-catch ile yakalanır ya da yakalanmayıp arayüzde throws ile fırlatıldığı belirtilir. Bu durum derleme zamanında kontrol edilir ve derleme hatası varsa bildirilir.
    - Unchecked Exceptionlar yakalanmak ya da metot arayüzünde throws ile fırlatıldığı belirtilmek zorunda değildir. Çünkü, uchecked exceptionlar programcı hatasıdır ve düzeltilmesi gerekir. Tüm unchecked exceptionlar java.lang.RuntimeException sınıfının ya da alt sınıfın nesneleridir. RuntimeException sınıfının ya da alt sınıflarının nesnelerinin fırlatılması derleme zamanında kontrol edilmez ya da belirtilmediğinde de hata oluşmaz.
    - RuntimeException sınıfının ya da alt sınıflarının nesnelerinin derleme zamanında kontrol edilmez ve yakalanmadığı ya da belirtilmediğinde de hata oluşmaz. Bu yüzden bu tür sıradışı durumların superclass'ına RuntimeException adı verilmiştir.
    - Checked exceptionlar umulan ya da düzeltilebilecek durumları ifade eder. Örneğin, FileNotFoundException. Bu yüzden çalışma zamanında yakalanarak düzeltilmelidir. Ama unchecked exceptionlar programcı hatasıdır ve çalışma zamanında oluşması beklenmemelidir. Örneğin, NullPointerException. Eğer bir unchecked exception çalışma zamanında oluşuyorsa, kodda düzeltilmelidir. Sonuç olarak bir yazılım, çalışma zamanında hiçbir unchecked exception fırlatmayacak hale getirilmelidir.

    Kaynak: https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/
    https://www.javaturk.org/download/java/oofp_with_java/7.-Bolum-Siradisi-Durum-Yonetimi-Exception-Handling.pdf
12. Cevap A. Kod örneğinde try bloğundaki ilk satırın icra ile "1" değeri basılır. Hemen ardından ClassCastException türü bir exception fırlatılmaktadır. İlk catch bloğu, ArrayIndexOutOfBoundsException için yazılmıştır. ArrayIndexOutOfBoundsException, ClassCastException sınıfının superclass'ı değildir. Bu nedenle il catch bloğu icra edilmez. ClassCastException sınıfı, Throwable sınıfının alt sınıfı olduğundan ikinci catch bloğu icra edilir ve "3" değeri basılır. Daha sonra finally bloğu icra edilir ve "4" değeri basılır. try-catch-finally bloklarının sonrasında da main() metodu sonra kodu ile "5" değeri basılarak program sonlandırılır.
    Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/package-tree.html
13. Cevap C. finally bloğunda exception fırlatılabilir ve bu durumda her satır icra edilmez. Bu nedenle A seçeneğindeki her satırın çalışması garantisi ifadesi ve D seçeneğindeki exception fırlatılamaması bilgileri yanlıştır. finally bloğu, ilgilisi olan catch bloğundan bağımsız olarak icra

edilir. Bu nedenle B seçeneği de yanlıştır. finally blokları için süslü parantezler zorunludur. Bu nedenle doğru cevap C'dir.
Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/finally.html

14. Cevap C. Kodda verilen IOException sınıfı, FileNotFoundException sınıfının superclass'ı olduğu için öncelikle FileNotFoundException için catch bloğu yazılmalıdır. Önce IOException için catch bloğu yazıldığı için kod derlenmez. Hata düzeltildiğinde kodun çıktısı "XY" şeklinde olacaktır.
Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/catch.html
15. Cevap C. try bloğu için bir catch bloğu ya da finally bloğu yazılmalıdır. Hem catch hem finally bloğu yazılmazsa kod derlenmez. Bazı exception yakalama durumlarında her ikisinin yazılmasını gerekir. Bu nedenle doğru cevap C seçeneğidir.
Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/summary.html
https://docs.oracle.com/javase/tutorial/essential/exceptions/try.html
16. Cevap B. A seçeneği exception esas rolünü ortaya koymaktadır. Beklenenden farklı bir durumun yakalanması amacıyla exception handling kullanılmaktadır. Exception fırlatan bir uygulamanın sonlanması önlenebilir. Bu nedenle B seçeneğindeki bilgi yanlıştır ve bu seçenek doğru cevaptır. C seçeneğindeki bilgi doğrudur. Fırlatılacak hatanın bir if-else ifadesi gibi bir kontrol yapısıyla kontrol edilerek bazı exceptionların programatik olarak fırlatılmasından kaçınılabilir. Genellikle NullPointerException, FileNotFoundException gibi exceptionlar için sık sık kullanılan bir yöntemdir. A seçeneğine benzer olarak D seçeneği de exception handling'in esas amaçlarından birini ifade etmektedir. Bir uygulama beklenmeyen problemi yakalayarak handle eder. D seçeneğindeki bilgi doğrudur.
Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html
17. Cevap D. Verilen kodda catch bloğu süslü parantezlerle değil, parantezlerle deklare edildiği için söz dizimi hatası vardır. Seçeneklerde verilenlerden farklı bir durumdan dolayı derlenmez.
18. Cevap B. Printer interface'inde soyut metot olarak tanımlanan printData() metodu PrintException fırlatmaktadır. Bu nedenle bu interface'i implemente edecek olan sınıfta bu metodun override edildiği yerde daha üst bir exception sınıfı ya da farklı bir exception yazılamayacağından dolayı PrintException fırlatan bir metot override edebilir. A seçeneğindeki metoda izin verilir, B seçeneğindeki metoda izin verilmez. C seçeneğindeki metot, Printer interface'inin printData() metodunu override etmez ancak bu metoda izin verilebilir. Bu nedenle doğru seçenek B'dir.
19. Cevap D. RuntimeException, Exception sınıfının; Exception sınıfı ise Throwable sınıfının subclass'ıdır. Throwable sınıfı, java.lang paketinde yer almaktadır. Bu paket her java dosyasında default olarak import edildiği için herhangi bir import işlemi gerekmez.
20. Cevap C. Verilen kod örneği c3 numaralı satırdan dolayı derlenmez. catch bloğu yazılırken yazım hatası vardır. catch bloğunda parantezler içinde yakalanacak exception parametre olarak gösterilmelidir.
21. Cevap B. Checked exceptionlar handle edilmeli ve bildirilmelidirler. Aksi durumda kod derlenmeyecektir. Unchecked exceptionların handle edilmesi isteğe bağlı olarak yakalanabilirler. java.lang.Error sınıfı hiçbir zaman uygulama tarafından handle edilmemelidir. Bu nedenle doğru cevap B seçeneğidir.
Kaynak: https://www.codejava.net/java-core/exception/java-checked-and-unchecked-exceptions
22. Cevap B. Verilen kod q2 numaralı satırdan dolayı derlenmez. CastleUnderSiegeException ya da daha üst başka bir sınıf, openDrawbridge() metodu imzasında bildirilmediği için bu satırdan dolayı derlenmez. Hata giderilirse, bu defa da main() metodunda exceptionların deklare

edilmesi gerekecektir. Bu deklarasyonun yapılması sonrası CastleUnderSiegeException fırlatılarak uygulama sonlanır.

23. Cevap A. Bir try-catch yapısında birden fazla catch bloğu implemente edilirse ilk eşleşmedeki catch bloğu icra edilecektir.
    Kaynak: https://docs.oracle.com/javase/8/docs/technotes/guides/language/catch-multiple.html
24. Cevap C. main() metodunda yer alan try bloğunun ilk satırındaki koddan dolayı kod derlenmez. main() metodu içinde kurulan try-catch bloğu NullPointerException yakalamak için kurulmuştur. Ancak, compute() metodunun Exception fırlatacak olması nedeniyle main() metodu içindeki catch bloğu ile bu exception yakalanmaz.
25. BOŞ.
26. Cevap B. Bir program sonsuz bir döngüde takılı kaldığında StackOverflowError fırlatırken, Var olmayan bir nesneye referansına başvuru yapıldığında NullPointerException fırlatır. Bu nedenle doğru cevap B seçeneğidir.
    Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/StackOverflowError.html
    https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html
27. Cevap C. Checked exceptionların handle edilmesi ve bildirilmesi gerekir. Bu nedenle A seçeneği doğrudur. Çağırana potansiyel sorunları bildirmek için kullanıldıkları için B seçeneği de doğrudur. Checked exceptionlar, çağırana hatasını düzeltmesini gösterdiği için D seçeneği de doğrudur. Checked exceptionlar, farklı bir exception alınmasını sağlamayabileceği için C seçeneği, checked exceptionların bir kullanılma sebebi değildir.
    Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html
    https://howtodoinjava.com/java/exception-handling/checked-vs-unchecked-exceptions-in-java/
28. Cevap D. Kodda verilen try-catch-finally yapısında catch ve finally blokları arasında sıralama hatası vardır. Bu yüzden kod derlenmez. Sıralama düzeltilirse B seçeneğindeki ifade ekran çıktısı olacaktır.
    Kaynak: https://www.geeksforgeeks.org/flow-control-in-try-catch-finally-in-java/
29. Cevap A. Bir try ifadesinde finally bloğu ya 1 tane olur ya da olmaz. catch bloğu ise ya hiç olmaz ya da birden fazla olabilir. Bu nedenle doğru seçenek A'dır.
    Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/try.html
30. Cevap D. Verilen kod derlenir. Fakat getDuckies() metodunda age değişkeni 5'tir. Ancak count değişkeni 0 olduğundan dolayı çalışma zamanında ArithmeticException hatası fırlatır. Bu hatanın fırlatılma nedeni 5/0 işleminde sıfıra bölmeye çalışılmasından kaynaklanmaktadır.
    Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/ArithmeticException.html
31. Cevap B. try ifadesinde hem catch bloğu hem finally bloğu bir exception fırlatıyorsa çağıran finally bloğundaki exceptionı alır, catch bloğundaki exception ihmal edilir.
32. Cevap A. m1 numaralı satırdan dolayı kod derlenmez. Çünkü, metot imzasında bir exception fırlatılacağı throws anahtar sözcüğü ile bildirilir. Hata giderilirse roar() metoduna 2 değeri gönderileceği için if kontrol yapısındaki ifade true döner ve IllegalArgumentException fırlatır.
    Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/declaring.html
33. Cevap A. Verilen kod parçası ClassCastException fırlatır. Object sınıfından olan exception nesne referansı alt sınıfı olan RuntimeException atanırken ClassCastException hatası fırlatacaktır.
    Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/ClassCastException.html
34. Cevap C. Tüm exceptionlar Throwable sınıfından türediği için tüm exception tiplerini handle edecek sınıf Throwable sınıfıdır.

Kaynak: https://docs.oracle.com/javase/7/docs/api/java/lang/package-tree.html
https://www.geeksforgeeks.org/exceptions-in-java/

35. BOŞ
36. Cevap A. ClassCastException sınıfı RuntimeException sınıfını genişletmektedir. ClassCastException sınıfının RuntimeException sınıfının subclass'ı olması nedeniyle ClassCastException yakalamak için inşa edilecek catch bloğu, RuntimeException yakalamak için inşa edilecek catch bloğundan önce implemente edilmelidir.
37. Cevap C. A seçeneği yanlıştır. Bilgisayarın yanmasına karşın exception fırlatmak herhangi bir çözüm üretmez. Derlenmeyen kod çalışmayacağı için exception fırlatılmaz. Bu nedenle B seçeneği de yanlıştır. D seçeneğinde verilen, bir metodun beklenenden erken sonlanması problemi exceptionlarla ilgili bir durum değildir. C seçeneğinde ifade edilen, bir metoda geçersiz veri aktarılması senaryosu exception kullanmak için iyi senaryolardan biridir.
Kaynak: https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html
38. Cevap C. Verilen kod derlenmez. Çünkü Organ sınıfındaki operate() metodu RuntimeException fırlatmaktadır. Heart sınıfındaki operate() metodunun superclass'ındaki operate() metodunu override edebilmesi için üst sınıfın aynı ya da daha üst bir exception'ı throws etmesi gerekir. Bu nedenle doğru cevap C seçeneğidir.
Kaynak: https://www.geeksforgeeks.org/exception-handling-with-method-overriding-in-java/
39. Cevap D. NullPointerException, bir unchecked exception'dır. Metot imzasında bildirilmesi isteğe bağlıdır. Bir try-catch bloğu ile sarılarak handle edilebilir ya da gözardı edilebilir. Bu nedenle D seçeneğinde verilen bilgi doğrudur.
Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html
40. Cevap D. Kod derlenmez. Çünkü zipper() metodundaki catch bloğunda bir RuntimeException fırlatılmak istenirken new anahtar sözcüğü eksiktir. RuntimeException(String) metodu, RuntimeException sınıfının bir kurucu metodu olduğundan dolayı new anahtar sözcüğü ile yeni nesne referansı üretilmelidir.
41. Cevap C. Verilen kod örneğinde try bloğunda bir ClassCastException fırlatılmaktadır. ClassCastException sınıfı, RuntimeException sınıfının subclassıdır. ClassCastException sınıfı, IllegalArgumentException sınıfı ile ilgili olmadığından dolayı finally bloğu icra edilir. finally bloğunda RuntimeException fırlatılarak uygulama sonlanır.
Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/package-tree.html
42. Cevap A. Verilen kod örneğinde Outfielder interface'inde bir metot bildirilmektedir. Burada bildirilen catchBall() metodu OutOfBoundsException fırlatmaktadır. Outfielder metodunu implemente eden bir sınıf yazıldığında, bu sınıfın Outfielder interface'indeki metotları override etmesi gerekecektir. Overriding metotların fırlatacağı exceptionlar superclass'larındaki ya da implemente ettikleri interface'lerdeki metotlardan daha geniş bir exception fırlatamaz. Bu nedenle verilen metot deklarasyonlarından sadece A seçeneğindeki deklarasyonu uygulayabilir.
43. Cevap D. Verilen kod örneğinde dancing() metodundaki catch bloğunda Error sınıfı bildirilmiş ancak değişken adı verilmemiştir. Yani try bloğunda yakalanan exceptionın hangi nesne referansında tutulacağı belirsizdir ve kod derlenmez. Buradaki hata çözülürse, kod IllegalArgumentException fırlatarak sonlanır. Eğer catch bloğunda bu exception ı yakalamak istersen Error sınıfından bir referans yerine IllegalArgumentException sınıfından bir referans

kullanarak dönen hatayı catch bloğunda yakalayabiliriz ve “Unable!” şeklinde ekran çıktısı elde ederiz.
Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/package-tree.html

44. Cevap D. Kod derlenir ancak çalışma zamanında openDrawbridge() metodunun try bloğunda bir Exception fırlatır. Fırlatılan catch bloğunda yakalanamaz. Çünkü, Exception sınıfından ya da daha üst bir sınıftan bir exception için catch bloğu yoktur. Sonrasında ise finally bloğu çalışır. finally bloğundan bir RuntimeException fırlatılır ve konsola “Or maybe this one” ifadesi fırlatılan exception ile beraber yazılır.
45. Cevap C. Hem IllegalArgumentException hem de ClassCastException, RuntimeException sınıfının subclasslarıdır. Bu nedenle her ikisi için istenen sırada catch bloğu yazılabilir.
Kaynak: https://docs.oracle.com/javase/8/docs/api/java/lang/package-tree.html
46. Cevap D. Kod örneğinde verilen RuntimeException bir interface değil sınıftır. Bu nedenle implemente edilmez, extend edilir. Bu hatadan dolayı kod derlenmez. Hata düzeltilirse “Problem?Fixed!” şeklinde ekran çıktısı üretir.
47. Cevap D. Verilen kod flipSwitch() metodundaki try bloğunda throws anahtar sözcüğü ile exception fırlatılmak istenmesinden dolayı derlenmez. Hata çözülürse önce “Flipped!” ifadesi stack trace olarak basılır. Daha sonra da “Circuit Break!” ifadesi ekran çıktısı olarak basılır.
48. BOŞ.
49. Cevap C. Verilen kod z1 numaralı satırdan dolayı derlenmez. Dışta kalan catch bloğunda Exception sınıfından üretilen nesne referansı ile iç kısımdaki catch bloğunda yer alan FileNotFoundException sınıfından üretilen nesne referansları çakışmaktadır. Hata giderilirse içteki catch bloğunda exception yakalanır ve ”Failed” ifadesi ekrana basılır.
50. Verilen kod öncelikle x1 numaralı satırdan dolayı derlenmez. Çünkü, fırlatılan exception, snore() metodunda bildirilmemiştir. Bu hata giderilirse daha sonrasında main() metodunda ya exception fırlatıldığı bildirilmelidir ya da main() metodunda snore() metodu çağrısını try catch blokları arasına almak gerekecektir.