Pytania na sprawdzian

1. Wyjaśnij rolę typów danych w językach programowania.
2. Opisz różnicę między typami wartościowymi i referencyjnymi – wymień po 5 przykładów typów, z każdej z kategorii.
3. Jaka jest różnica między int a Int32?
4. Wyjaśnij jak działa „var” w C#. Kiedy dozwolone jest korzystanie z tego zapisu?
5. Opisz różnicę między float, double i decimal.
6. Czym są typy wyliczeniowe? Kiedy warto z nich skorzystać, jakie są zasady definiowania typów wyliczeniowych?
7. Opisz krotki i ich cechy.
8. Opisz typy nullable – jak działają, kiedy z nich korzystać.
9. Czym się różni stos i sterta? Jakie dane przechowują te mechanizmy w .NET.
10. Czym jest i jak działa Garbage Collector?
11. Opisz typ object oraz jego potencjalne zastosowania.
12. Czym jest boxing i unboxing?
13. Jak działają operatory: ?, ??. Podaj przykład kodu z ich wykorzystaniem.
14. Opisz rolę przestrzeni nazw – czym są, do czego służą?
15. Opisz operatory @ i $ przy tworzeniu zmiennej typu string.
16. Czym jest i jak działa dynamic?
17. Opisz pętle w C#.
18. Jak działa instrukcja switch? Jak zaimplementować możliwość użycia kilku casów – jeden pod drugim / or.
19. Opisz działanie słowa kluczowego when w instrukcji switch.
20. Czym jest zmienna?
21. Czym jest obiekt?
22. Jak działają parametry domyślne metod w C#?
23. Jak działa słowo kluczowe default?
24. Opisz słowa kluczowe: ref, out, in.
25. Czym jest przeciążanie metod?
26. Czym jest klasa?
27. Opisz rolę oraz dostępne modyfikatory dostępu w C#.
28. Czym są pola i właściwości w klasach?
29. Czym jest (*jak działa?*) konstruktor klasy?
30. Wyjaśnij cel tworzenia prywatnych konstruktorów.
31. Słowo kluczowe this – jak działa, kiedy używamy?
32. Opisz istotę klas wewnętrznych.
33. Czym jest klasa statyczna? Kiedy się z nich korzysta? Jakie są zasady jej projektowania?
34. Czym są statyczne konstruktory, właściwości statyczne i metody statyczne? Jaka jest różnica między nimi, a odpowiednikami instancyjnymi?

35. Czym są klasy cząstkowe? Kiedy się z nich korzysta, w jaki sposób?
36. Czym są oraz jak tworzyć metody rozszerzeń?
37. Czym jest enkapsulacja (*hermetyzacja*) w rozumieniu programowania obiektowego?
38. Czym są interfejsy? Kiedy warto z nich skorzystać? Jak się je implementuje?
39. Zasady projektowania interfejsów.
40. Czym jest dziedziczenie? Jak działa? Kiedy warto sięgnąć po ten mechanizm?
41. Opisz rolę i słowa kluczowego base.
42. Czym jest polimorfizm? Jak działa?
43. Czym jest przesłanianie (*shadowing*)? Jak działa?
44. Czym jest (*i kiedy jest możliwe*), jak działa nadpisywanie metod?
45. Podaj sposobu rzutowania obiektów – kiedy można rzutować obiekt, jakie są sposoby i czym się od siebie różnią? – *zwykłe rzutowanie, as, is*.
46. Czym są i jak działają klasy abstrakcyjne? Podaj sens ich istnienia.
47. Czym są metody abstrakcyjne?
48. Czym są (*i jak działają*) metody i klasy zapieczętowane (*sealed*).
49. Opisz cztery podstawowe metody obecne w każdym obiekcie – *Equals(), GetHashCode(), ToString(), GetType()*. Której z metod nie da się nadpisać? Dlaczego?
50. Opisz sens stosowania interfejsu IEquatable<T>, dlaczego samo Equals() to mało?
51. Czym są struktury w C#?
52. Kiedy używać struktury, a kiedy klasy?
53. Czym są rekordy w C#?
54. Podaj sposoby definiowania rekordów.
55. Kiedy korzystać z rekordów, a kiedy klas?
56. Opisz różnicę między rekordami referencyjnymi a wartościowymi.
57. Czym są typy anonimowe? Kiedy warto z nich skorzystać, a kiedy jest to niemożliwe?
58. Podaj rolę interfejsu IDisposable. Jakie możliwości dostarcza jego implementacja?
59. Kiedy korzystamy (kiedy można) z bloku using? (using var a = new ...())
60. Czym są wyjątki?
61. W jaki sposób obsłużyć wyjątki?
62. Opisz w jaki sposób działa blok finally.
63. W jaki sposób obsłużyć różne typy wyjątków? – wiele bloków catch.
64. Jaki jest wpływ występowania wyjątków na wydajność programu?
65. Opisz obiekt Exception – wymień i opisz jego najważniejsze właściwości (*Message, StackTrace, InnerException*).
66. Czym jest owijanie wyjątków? Kiedy się je stosuje?
67. W jaki sposób rzucić wyjątek? – użycie słowa kluczowego throw.

68. Wyjaśnij różnicę między throw a throw ex.
69. Kiedy program powinien rzucać wyjątek?
70. Czym jest połykanie wyjątków?
71. Opisz sens tworzenia własnych typów wyjątków.
72. Czym są (*dlaczego powstały?*) kolekcje?
73. Czym jest typ generyczny? Jakie są jego zalety i jakie daje możliwości?
74. Czym jest pojemność (*capacity*) listy? Jak działa?
75. Czym jest słownik? Jakie są wymagania co do klucza?
76. Czym jest hashset? Jakie wymagania powinien spełniać obiekt w kolekcji?
77. Opisz interfejs IEnumerable<T>, jakie daje możliwości?
78. Jak działa pętla foreach?
79. Czym jest i jak działa wykonanie odroczone?
80. Opisz różnicę między break i yield break – *specjalna dedykacja dla Filipa*. 🙄
81. Zasady tworzenia metod umożliwiających odroczone wykonanie.
82. Jak stworzyć własną kolekcję w C#? Szczegółowy opis implementacji interfejsu IEnumerable<T>.
83. Jak dodać obsługę indeksatora do własnej kolekcji?
84. Czym są delegaty? Opisz ich rolę i zastosowania.
85. Wymień oraz opisz podstawowe delegaty predefiniowane – Action, Func, Predicate.
86. Jaką rolę pełnią funkcje anonimowe? Kiedy warto z nich skorzystać?
87. Czym są oraz w jaki sposób są konstruowane wyrażenia lambda?
88. Czym są zdarzenia? Jak się je definiuje?
89. Opisz znaczenie i potencjalne zastosowania delegaty EventHandler.
90. Opisz ryzyko wycieków pamięci w C# przy korzystaniu z mechanizmu zdarzeń.
91. Czym są atrybuty? Opisz ich rolę i sposób działania.
92. Jak i kiedy tworzymy własne atrybuty?
93. Opisz istotę refleksji – możliwe zastosowania, ryzyka, ograniczenia związane z refleksją. Jaki powoduje ona narzut obliczeniowy?
94. Czym jest LINQ? (*style, zastosowania, kiedy możliwe jest jego użycie?*)
95. Wymień i opisz podstawowe operatory LINQ – Where, Select, OrderBy, ThenBy, GroupBy, Distinct.
96. Znaczenie operatorów kończących w LINQ – jakie są pułapki związane z ich niezastosowaniem? Na co trzeba uważać?
97. Do czego służy interfejs IQueryable<T> - na jakich źródłach danych operuje, w jaki sposób?
98. Opisz różnicę między IEnumerable<T> i IQueryable<T>.
99. Jak działa tłumaczenie LINQ na SQL? Czym są operacje nietłumaczalne?
100. Czym jest wątek programu?
101. Opisz znaczenie wątku UI.
102. Opisz różnicę między zadaniami ciężkimi obliczeniowymi a zadaniami I/O.

103. Opisz różnicę między czekaniem synchronicznym i asynchronicznym?
104. Czym jest asynchroniczność? Wyjaśnij jak działa.
105. Opisz słowa kluczowe async i await.
106. Czym jest Task?
107. Czym jest async void? Dlaczego nie powinniśmy go stosować, kiedy jest on dopuszczalny?
108. Dlaczego metody asynchroniczne muszą zwrócić wynik opakowany w Task<T>?
109. Wyjaśnij różnicę między Task a ValueTask.
110. Opisz różne strategie używania await.
111. Czym jest odrzucanie tasków? Jak działa?
112. Czym jest kontekst synchronizacji?
113. Opisz strategię anulowania operacji asynchronicznych – CancellationToken.
114. Czym jest przetwarzanie równoległe?
115. Opisz różnicę między równoległością a asynchronicznością.
116. Czym jest oraz jak działa ThreadPool? Jakie niesie ze sobą zalety?
117. Opisz różnicę między new Thread() oraz Task.Run().
118. Czym jest biblioteka Parallel? Jakie metody oferuje?
119. Czym jest Parallel LINQ? Kiedy warto z niego skorzystać?
120. Czym jest stopień równoległości?
121. Opisz rolę synchronizacji kodu równoległego – dlaczego jest ona potrzebna, kiedy z niej skorzystać, kiedy można przesadzić?
122. Wymień sposoby synchronizacji kodu – lock, semafory, kolekcje bezpieczne wątkowo – *nie zapomnij że lock to ryzyko deadlock, kiedy?*.
123. Opisz wady i zalety kolekcji bezpiecznych wątkowo. Kiedy warto z nich skorzystać?
124. Opisz rolę operacji atomowych w kodzie równoległym.
125. Wszystkie konwencje nazewnictwa:
    a. Klasy,
    b. katalogi, przestrzenie nazw,
    c. pola, właściwości,
    d. metody, argumenty metod,
    e. interfejsy, typy generyczne
    f. klasy abstrakcyjne,
    g. zmienne, stałe,
    h. metody asynchroniczne,
    i. itd.