Instuderingsfrågor inför slutprov i Programmering 2

Provet omfattar kursboken: *Programmering 2 med C#*, kap 2-4, **sid 69-188**.

Svar till alla frågor finns i kursboken *Programmering 2 med C#*.

Svar till vissa frågor finns i kursboken *Programmering 1+ med C#*.

Kap 2 Objektorienterad programmering (OOP), sid 69-128

1. Kan man skriva i C# ett program utan att deklarera en klass?
2. Vad är definitionen på *program* i C#? (Se ***Programmering 1+ med C#***).
3. Är det korrekt att säga att pepparkakor är *klasser* och pepparkaksformen *objekt*?
4. Kan man via *abstraktion* komma från objekt till klass eller är det tvärtom?
5. Om pennor är objekt var kan man hitta klassen *penna*?
6. Av vilka två huvudingredienser består en klass i regel?
7. Anta att *Tal* är en klass. Är *addition()* en metod eller en datamedlem i klassen *Tal*?
8. Anta att *Bil* är en klass. Är *Motor* en metod eller en datamedlem i klassen *Bil*?
9. Är `+` en metod, en datamedlem eller en operator i klassen *String*? Vad heter den?
10. Med vilken fråga hittar man datamedlemmarna i en klass?
11. Med vilken fråga hittar man metoderna i en klass?
12. Är *objekt* och *instans* synonymer?
13. Vad är definitionen på en *klass* i C#? Finns andra definitioner? Om ja, nämn dem alla.
14. Vilka tre steg måste man ta när man använder en klass?
15. Om man deklarerat klassen `Employee`, kan man skapa ett objekt av den med koden `new Employee`?
16. … Eller måste man skriva `new Employee()`?
17. Är parentesen `()` obligatorisk i koden `new Employee()`?
18. Vad gör parentesen `()` i koden `new Employee()`?
19. Vad heter de variabler som kan tilldelas objekt?
20. Vad är skillnaden mellan datamedlemmar och lokala variabler?
21. Vad betyder *punktnotation*?
22. Kan man skriva `static` framför namnet på en klass?
23. Är `static` en åtkomst- eller en allokeringsmodifierare?
24. Vad betyder *allokering* i datasammanhang?
25. Vad innebär en *statisk datamedlem*?
26. Vad händer med default konstruktorn om vi själva definierar en egen konstruktor?

27. Hur kommer man åt en statisk datamedlem, med klass- eller objektnamnet?
28. Vilka är synonymer bland termerna *klassvariabel*, *instansvariabel* och *statisk datamedlem*?
29. Vilka är de tre hörnstenarna inom objektorienterad programmering?
30. Kan man skriva `private` framför namnet på en klass?
31. Är `private` en allokerings- eller en åtkomstmodifierare?
32. Brukar man deklarera datamedlemmarna eller metoderna som `private` i OOP?
33. Är innehållet i en klass by default `private` eller `public`?
34. Vilka är de tre viktigaste egenskaperna hos klassens konstruktor?
35. Finns klassens konstruktor även om vi inte definierar den explicit i klassen?
36. Varför och när behövs klassens konstruktor?
37. Kan man ha flera konstruktorer i en klass? Om ja, vad är meningen med detta?
38. Vad är *default konstruktorn*?
39. Hur många parametrar har default konstruktorn?
40. Vad är konstruktorns huvuduppgift?
41. Skapas i satsen `Circle[] c = new Circle[5];` fem objekt av typ `Circle`?
42. Är det korrekt att prata om *array av objekt*? Om nej, med vilket begrepp borde den ersättas?
43. Om en bil är ett objekt av klassen *Bil* och dess motor ett objekt av klassen *Motor* vilken relation råder mellan dem?
44. Om man vill skriva klassen *Anställd* måste man, för att modellera anställdens arbetstid, bygga in den som en klass *Tid* eller *Datum* i klassen *Anställd*?
45. Om du skulle skriva klassen *Cylinder* skulle du relatera till den klassen *Cirkel* som komposition eller som arv?
46. Pekar i ett klassdiagram pilen från subklassen till superklassen eller tvärtom?
47. Använder programmet `Composition` (sid 106-107) komposition av objekt eller av klasser?
48. När en subklass ärver en superklass ärver den superklassens alla datamedlemmar och metoder?
49. När en subklass ärver en superklass ärver den automatiskt även superklassens konstruktor?
50. Skriv huvudet till subklassen *Anställd* som ärver superklassen *Person*.
51. Skriv huvudet till subklassens konstruktor i fråga 50 om superklassen har datamedlemmarna *förnamn, efternamn, födelsedatum* och subklassen lägger till datamedlemmen *anställningsdatum*.
52. Vad är `base()` i huvudet till konstruktorn av en subklass?
53. Vilka två koder etablerar en arvrelation mellan en sub- och en superklass?
54. Kan det förekomma en kombination av komposition och arv i ett C#-program? Hitta exempel för det i bokens programexempel.
55. Varför är polymorfism en speciell form av överlagring av metoder?
56. Vad skiljer polymorfism från vanlig överlagring av metoder?
57. Vad händer med default konstruktorn om vi själva definierar en egen konstruktor?

58. Kan polymorfa metoder förekomma i en och samma klass?
59. Kan överlagrade metoder förekomma i en och samma klass?
60. Har överskuggade metoder samma huvud inkl. parameterlista?
61. Kan man komma åt de privata datamedlemmarna i en superklass från en subklass?
62. Är `protected` en allokerings- eller en åtkomstmodifierare?
63. Är protected mer eller mindre restriktiv än private?
64. Vad gör protected när man skriver den framför en datamedlem?
65. Kan protected skrivas även framför metoders namn?
66. Kan protected skrivas även framför klassers namn?
67. Vad exakt innebär överskuggning av metoder?

Kap 3 Metoder i OOP, sid 129-151

68. Vad är *accessmetoder*? Nämn tre typer av accessmetoder.
69. Varför behöver man accessmetoder?
70. Beskriv sambandet mellan accessmetoder och *Property*.
71. Vad innebär en *statisk metod*?
72. Hur kommer man anropa en statisk metod, med klass- eller objektnamnet?
73. Kan man skicka referensvariabler som parametrar till en metod?
74. Vad innebär *blockstruktur* i C#? Nämn ett exempel.
75. *Algoritmen för platsbyte* kodas i kursboken på sid 154. Är programmet **`MiniSort`** objektorienterat?
76. Varför misslyckas försöket att modularisera **`MiniSort`**?
77. Beskriv med egna ord parameteröverföringsmetoden *värdeanrop* (*Call by value*).
78. Beskriv med egna ord parameteröverföringsmetoden *referensanrop* (*Call by reference*).
79. Vilka ändringar måste göras i koden, för att lyckas med modulariseringen av **`MiniSort`**?
80. Hur måste en parameter i en metod deklareras för att bli en utparameter?
81. Vad är skillnaden mellan en metods returvärde och metodens utparametrar?
82. Varför (och när) behövs utparametrar, när man kan exportera data från en metod via returvärdet?
83. Hur kan man deklarera en array som parameter i en metod?
84. Var någonstans måste storleken på en array anges när den skickas som parameter till en metod?
85. Vilken parameteröverföringsmetod använder C# automatiskt på en array som parameter?

Kap 4 Mer om metoder, sid 153-188

86. Beskriv kort reglerna för livslängden av variabler.
87. Är det korrekt att prata om *globala variabler* i C#? Om nej, med vilket begrepp borde den ersättas?
88. Vad händer med default konstruktorn om vi själva definierar en egen konstruktor?

89. Vad innebär *överskuggning av variabler*? Vad är skillnaden till *överskrivning av variabler*?
90. Är `this` en klass, ett objekt, en datamamedlem, en metod eller en referensvariabel?
91. När behöver man använda `this`?
92. Vad innebär *överlagring av metoder*?
93. Har överlagrade metoder samma *signatur*?
94. Kan överlagrade metoder ha samma antal parametrar?
95. Kan i en metod parametrarnas datatyper vara variabla?
96. Vad heter de metoder där variabler är platshållare för datatyper?
97. Vad innebär *generiska metoder*?
98. Nämn exempel på när man har en fördel av att definiera *generiska metoder*.
99. Skriv ett exempel på ett *lambdauttryck*.
100. Vad står *LINQ* för och vilket språks syntax liknar det?
101. Vad är *delegater*?
102. Vad innebär *metodgrupper*?

Copyright © 2022 Taifun Alishenas