Facit till
Slutprov i Programmering 2

Namn: _____

Klass: _____

Regler:

- Detta prov är ett ***”öppet prov”***, dvs:
- Alla hjälpmedel är tillåtna, förutom att kommunicera med varandra. För att kontrollera det:
- Förbehåller sig läraren rätten att efter provet kalla eleven till samtal för att få förklaringar för vissa utvalda lösningar.
- Flervalsfrågor (Multiple choice questions) med symbolen ring (○) har endast ett korrekt svar. Frågor med symbolen ruta (□) har flera korrekta svar.
- Skriv dina svar tydligt, strukturerat och förståeligt på extra papper och hänvis till uppgifternas nummer.
- Endast kod som uppfyller kraven på ***god programmerings-stil***, ger poäng.
- Det är inte tillåtet att kopiera, spara eller fota provfrågorna.
- Provtiden är obegränsad.

1. Den objektorienterade synen på programmering definierar *program* som:

   - o Problemlösning med hjälp av datorn
   - o Program = algoritm + data
   - ☒ Program = Modell av verkligheten
   - o Användarvänlig applikation (2/0/0)

****

2. En mer detaljerad definition på *C# program* kan formuleras så här:

   - o En samling av metoder av vilka en är **`Main()`**-metoden som exekveringens startpunkt.
   - ☒ En samling av klasser av vilka en innehåller **`Main()`**-metoden som exekveringens startpunkt.
   - o En samling av objekt som kommunicerar med varandra med hjälp av referenser.
   - o Kod med korrekt syntax som kan både kompileras och exekveras. (2/0/0)

****

3. Ange de tre hörnstenarna (viktiga princip) i objektorienterad programmering. (2/0/0)

   1. Inkapsling 2. Arv 3. Polymorfism

****

4. Vad menas med *modularisering* i programmeringssammanhang? (2/0/0)

   Att bryta ned ett stort problem i mindre moduler och bearbeta modulerna en i taget, liknande Lego-principen.
   I OOP: moduler = klasser.
   Att sedan sätta ihop modulerna till det stora programmet.

****

5. Definiera (förklara) begreppen *klass* och *objekt*. (Ersätt dem inte med några synonym.) (2/1/0)

   En klass är kod som beskriver en kategori av saker och ting.
   En klass består av datamedlemmar och metoder.
   En klass används som mall för att skapa objekt av klassen.
   En klass är en ny datatyp som skapas med **`class`**.

****

6. Anta att **`Employee`** är en klass. Vilken *typ av variabel* är **`a`** i följande sats: (1/1/0)

```
Employee a = new Employee();
```

   Referensvariabel.

****

7. Skriv den del av kod som i satsen ovan (fråga 6) skapar ett *objekt* av typ **`Employee`**: (1/1/0)

`new Employee()`

****

8. Till vilket värde initieras datamedlemmar i **`Employee`**-objektet som skapas med satsen i fråga 6 om de är av typ **`String`**? (1/1/0)

Tomma strängen. I kod: `""`

****

9. Anta att **`Circle`** är en klass. Går det att kompilera följande kod: **`Circle c = new Circle;`**

- o Ja
- ☒ Nej

****

10. Vad kallas en referens som inte pekar på något objekt? (1/1/0)

`null`-referens

****

11. Det reserverade ordet **`static`** är en: (2/0/0)

- o Åtkomstmodifierare
- ☒ Allokeringsmodifierare

****

12. Vad är konsekvensen av att skriva **`static`** framför namnet på en datamedlem? (1/1/0)

Statiska datamedlemmar allokeras i klassen, inte i objekten.
Statiska datamedlemmar är klassvariabler, inte instansvariabler.
Konsekvensen är att man utanför klassen kommer åt dem med klassnamnet, inte med objektnamnet.

****

13. Det reserverade ordet **`private`** är en: (2/0/0)

- ☒ Åtkomstmodifierare
- o Allokeringsmodifierare

****

14. Är det en regel eller en rekommendation i objektorienterad programmering att deklarera sina datamedlemmar som **`private`** och sina metoder som **`public`**? (2/0/0)

- o Regel
- ☒ Rekommendation

15. Vilket verktyg har man i C# för att *initiera* klassens privata datamedlemmar när ett objekt skapas?

Klassens konstruktor. (0/1/0)

****

16. Beskriv de tre viktigaste egenskaperna hos en *konstruktor:* (1/2/0)

1. Konstruktorn har samma namn som klassen.

2. Returtypen saknas.
   Konstruktorn kan inte returnera ett värde. Den får inte ens ha returtypen **`void`** framför sitt namn

3. Anropet av konstruktorn sker i samma sats som objektet skapas.

****

17. Skapa två objekt av följande klass, dvs två hundar. Låt den ena skälla (eng. *bark*) och den andra att sätta sig ned, genom att anropa klassens metoder. (2/2/2)

```
class Dog
{
   string name;
   string color;
   int age;
   float size;

   public Dog(string n, string c, int a, float s)
   {
      name = n;
      color = c;
      age = a;
      size = s
   }

   public void Bark()
   {
      Console.WriteLine(name + ": Voff!");
   }

   public void Sit()
   {
      Console.WriteLine(name + " sätter sig ned.");
   }
}
```

```
Dog d1 = new Dog("Rune", brun, 4, 30);
```

```
Dog d2 = new Dog("Happy", svart, 3, 45);
```

```
d1.Bark();
```

```
d2.Sit();
```

****

18. Är datamedlemmarna i klassen **`Dog`** (uppg. 17) privata eller publika? Varför? (0/2/0)

Privata, därför att alla medlemmar i en klass är by default privata.

****

19. Kan man utelämna konstruktorn i klassen **`Dog`** (uppg. 17)? Om nej motivera varför inte? Om ja, skapa ett objekt av denna klass (utan konstruktor). (1/2/1)

Ja, då aktiveras default konstruktorn: `Dog d3 = new Dog();`

****

20. Skriv en klass **`Triangle`** med lämpliga privata datamedlemmar, en egen konstruktor och publika metoder för beräkning av arean och omkretsen. (1/2/2)

Lösning:

```
class Triangle
{
   private int side_a, side_b, side_c, height_b;  // Datamedlemmar

   public Triangle(int a, int b, int c, int h)    // Konstruktorn
   {
      side_a   = a;
      side_b   = b;
      side_c   = c;
      height_b = h;
   }

   public int Area()                              // Metod
   {
      return side_b * height_b/2;
   }

   public int Circumference()                     // Metod
   {
      return side_a + side_b + side_c;
   }
}
```

21. Skapa ett objekt av typ **`Triangle`** (uppg. 17) som initieras till valfria, meningsfulla värden. (1/1/0)

```
Triangle t = new Triangle(4, 6, 5, 3);
```

****

22. Skapa först en array av 10 referenser till objekt av typ **`Triangle`** (uppg. 17) och sedan i en **`for`**-sats 10 objekt av typ **`Triangle`**. Tilldela varje objekt till en av referenserna. (1/1/2)

```
Triangle[] tr = new Triangle[10];
```

```
for (int i = 0; i < 10; i++)
    tr[i] = new Triangle();
```

****

23. Vilka verktyg har man i C# för att *ändra* klassens privata datamedlemmar efter initieringen?

Accessmetoder: Get-metoder och Set-metoder. (0/1/1)

****

24. Vad är *Property* i C# och vad är i dess samband med *accessmetoder*? (0/1/1)

Property är en slags generaliserad datamedlem som är publik och som automatiserar Get-metoder och Set-metoderna för att utifrån klassen kunna läsa värdet av och skriva ett nytt värde till klassens privata datamedlemmar.

****

25. På vilket sätt måste man anropa en metod som har egenskapen **`static`**? (1/1/0)

En statisk metod måste man anropa med klassnamnet, inte med objektnamnet.

****

26. Vad är reglerna för *variablers livslängd* i C#: (1/1/0)

Variablers livslängd börjar med deklarationen och slutar med det block i vilket de är deklarerade.
De är giltiga i det block de är deklarerade och i alla underblock, men inte i överordnade block.
Variabler "söker" efter sin deklaration uppåt i blockstrukturen.

****

27. Skriv ett exempel på kod som består av två block dvs har *blockstruktur*. Skapa i den en variabel som *överskuggar* en annan variabel. Använd **`this`** för att komma åt den överskuggade variabeln (Följ god programmeringsstil och kommentera din kod). (0/2/3)

```
class Scoping
{
   int x;                          // Datamedlem
   static void Main()
   {
      Scoping s = new Scoping();   // Objekt initierar x till 0
      s.x = 100;                   // Referens ändrar x

      s.Inner();                   // Anrop av metod: Inre block
   }

   void Inner()                    // Definition av metod
   {
      int x = 1;                   // Lokal variabel: samma namn som
                                   // datamedlem: överskuggar den
      this.x++;                    // Datamedlem åtkomligt med this
   }
}
```

****

28. Följande är huvudet till en metods definition. Vilken typ av metod är detta? Vad betyder koden **`<T>`** och vilken datatyp har metodens parameter **`t`** ? (0/1/1)

```
public static void G_out <T> (T[] t)
```

Generisk metod. Koden **`<T>`** betyder att är **`T`** en variabel datatyp. Parametern **`t`** är av typ **`T`**.

29. Anta att du har en klass **`Circle`** med en privat datamedlem **`radius`**. a) Skriv huvudet till en subklass **`Cylinder`** som ärver **`Circle`** dvs utvidgar cirkeln genom att lägga till en privat datamedlem **`height`**. b) Skriv huvudet till subklassen **`Cylinder`**:s konstruktor.

a) **`class Cylinder : Circle`**

b) **`public Cylinder(double radius, double height) : base(radius)`**

(0/1/2)

****

30. Förklara med egna ord begreppet *polymorfism*. Nämn minst ett exempel. (1/1/1)

Polymorfism modifierar helt eller delvis funktionaliteten hos metoder med samma namn som förekommer i en arvhierarki. Polymorfa metoder har samma namn och samma parameterlista. Deras huvuden är identiska. Men deras kroppar (innehåll) är olika.
Exempel: Metoden **`WithDraw()`** i klasserna (arvhierarkin) **`Account/MinimalAccount`** (sid 115).

****

Lycka till!

Poängfördelning:
Vid varje uppgift anges maximala antalet poäng som du kan få för din lösning.
Det finns E-, C- och A-poäng. T.ex. betyder (3/2/1) att uppgiften kan ge max. 3 E-, 2 C- och 1 A-poäng.
Uppgifter med A-poäng erbjuder möjligheter att visa kunskaper på A-nivå, s.k. A-kvaliteter. Dessa bedöms *kvalitativt* enligt betygskriterierna för A. Man tittar snarare på *A-kvaliteterna:* generella lösningar, välstrukturerad kod, användning av programmeringstekniska koncept, korrekt språk osv.

Betygsgränser:

Totalt 32/29/16 varav minst: 16 p ger E.

38 p varav 18 C-poäng ger C.

60 p varav 22 C-poäng och dessutom 12 av 16 A-poäng ger A.

Inlämningsuppgifter (se kursens webbsida) kommer att bidra med bonuspoäng till provbetygen.