# Lösningsförslag: Instuderingsfrågor del F

**Uppgift 1.**

Skillnaden mellan en klass och ett objekt är den att klasser fungerar som en ritning/mall utifrån vilken objekt skapas när ett program exekveras. Alla objekt som skapas från en viss klass får samma egenskaper och beteenden. En klass är således en beskrivning av en mängd objekt med gemensamma egenskaper och beteenden. Klassen specificerar den information/data som kan lagras i ett objekt samt de metoder som ett objekt kan utföra.

Ett objekt är en unik och konkret realisering av en klass. Alla objekt har ett tillstånd, ett beteende och en identitet.

**Uppgift 2.**

Det är ingen skillnad, utan två olika sätt att uttrycka samma sak.

**Uppgift 3.**

En konstruktor är den metod som anropas när man skapar ett objekt av en klass. Anropet sker med användning av operatorn **new** enligt:

```
KlassNamn nyttObjekt = new KlassNamn(parameterlista);
```

En konstruktor deklareras inuti en klass och den har samma namn som klassen själv. Det är möjligt att överlagra konstruktorer, dvs en klass kan ha flera konstruktorer, var och en med sin unika parameterprofil.

**Uppgift 4.**

Nej! Deklareras ingen konstruktor i en klass har man ändock tillgång till den parameterlösa konstruktorn. Men om man deklarerar en eller flera egna konstruktorer måste man också själv deklarera den parameterlösa konstruktorn.

**Uppgift 5.**

Attribut är en gemensam beteckning för de variabler (instansvariabler och klassvariabler) och konstanter som finns i en klass.

**Uppgift 6.**

Inkapsling är en grundprincip i objektorienterad programmering som innebär att objektets inre uppbyggnad är dold för den som använder klassen. Allt programmeraren behöver veta är vad objektet kan göra och hur man ber objektet att göra det. Inkapsling innebär att man skiljer mellan objektets implementation och dess gränssnitt (specifikation). Ett objekt kan endast påverkas via de metoder som finns specificerade för objektet. Inkapsling innebär också att objektet kan isoleras mot fel i andra objekt.

**Uppgift 7.**

a) En enkel variabel lagrar data som tillhör någon av de enkla datatyperna, t.ex **int**, **double**, **boolean** och **char**.

b) En referensvariabel är en variabel som refererar till ett objekt. En referensvariabel av en viss klass kan endast kan referera till objekt av denna klass.

c) En lokal variabel är en variabel som deklareras inuti ett block eller inuti en metod . En lokal variabel har begränsad livslängd, den skapas när man börjar exekvera blocket och den är inte längre åtkomlig när man lämnat blocket.

d) En instansvariabel är en variabel som håller information om ett objekts tillstånd. En instansvariabel är specifik för varje instans av en klass.

e) En klassvariabel är en variabel som är gemensam för alla objekt av en viss klass. En klassvariabel är en resurs som delas av alla objekt av en klass.

**Uppgift 8.**

Orsaken är att variablerna value1 och value2 är deklarerad som en instansvariabler men vi har inget objekt av klassen Strange! Det finns tre lösningar på detta problem. Den första, och den som absolut är att föredra, är att göra variablerna lokala i main():

```
public class Strange {
    public static void main(String[] arg) {
        int value1 = 2, value2 = 4;
        System.out.println((2 * value1 + 5 * value2 - 4));
    }//main
}//Strange
```

Ett annat sätt som är möjlig, men olämpligt, är att skapa ett objekt av klassen Strange och referera instansvariablerna via detta objekt:

```
public class Strange {
    public int value1 = 2, value2 = 4;
    public static void main(String[] arg) {
        Strange objekt = new Strange();
        System.out.println((2 * objekt.value1 + 5 * objekt.value2 - 4));
    }//main
}//Strange
```

Ett tredje sätt som är möjlig, men oerhört ”fult”, är att göra variablerna till klassvariabler:

```
public class Strange {
    static int value1 = 2, value2 = 4;
    public static void main(String[] arg) {
        System.out.println((2 * value1 + 5 * value2 - 4));
    }//main
}//Strange
```

**Uppgift 9.**

En statisk metod eller klassmetod är en metod som kan anropas utan att använda en referens till en instans av dess klass. Eftersom ett sådant metodanrop inte är kopplat till något objekt så kan klassmetoder inte referera till klassens instansvariabler.

**Uppgift 10.**

När en metod anropas sker en bindning mellan de aktuella och formella parametrarna. I Java sker bindningen via ett så kallat värdeanrop, vilket innebär att den formella parametern tilldelas värdet av motsvarande aktuell parameter. Således gäller vid anrop till en metod att de de aktuella parametrarna måste överensstämma med de formella parametrarna i antal, typ och ordning. Efter anropet har den aktuella parametern samma värde som innan anropet.

**Uppgift 11.**

En synlighetsmodifierare kan anges för en klass samt för attributen och metoderna som finns i en klass. Synlighetsmodifieraren specificerar vem som ser (= har access) till entiteten i fråga. Följande tre synlighetsmodifierare finns

**public** - anger att entiteten är synlig för alla klasser

**protected** - anger att entiteten är synlig i för klasser inom samma paket och för subklasser

**private** - anger att entiteten är synlig endast inom klassen själv

utelämnad – anger att entiteten är synlig i för klasser inom samma paket

**Uppgift 12.**

Överlagring innebär att det inom en och samma klass finns flera metoder (eller konstruktorer) med samma namn, men med olika parameterprofiler.

**Uppgift 13.**

Det finns två konstruktorer med samma parameterprofiler:

```
public ClassA(int x) { . . .}
public ClassA(int y) { . . .}
```

**Uppgift 14.**

Utskriften blir:

```
0
3
```

**Uppgift 15.**

Utskriften blir:

```
The name is: First
The name is: Second
Second Second
```

**Uppgift 16.**

Följande satser är felaktiga:

| | |
|---|---|
| obj.b = 10; | instansvariabeln b är deklarerad **private** |
| obj.m1(7.8); | metoden m1 skall ha en parameter av typen **int** |
| **double** x = obj.m2(); | metoden m2 är deklarerad som **void** |
| System.out.println(obj.m2(12.8)); | metoden m3 är deklarerad som **private** |

**Uppgift 17.**

Utskriften blir:

```
x = 20
y = 30
z = 20
```

**Uppgift 18.**

Orsaken är att metoden compute är deklarerad som en *instansmetod* men vi har inget objekt av klassen Uppgift1A. Det finns två lösningar på detta problem. Den första, och den som är att föredra, är att göra metoden compute till en *klassmetod* genom att ändra metodhuvudet till

```
public static int compute(int a, int b)
```

Den andra metoden som är möjlig, men onaturlig, är att skapa ett objekt av klassen Uppgift1A och anropa metoden compute för detta objekt. Metoden main skulle då få följande utseende:

```
public static void main(String[] arg) {
    Uppgift1A obj = new Uppgift1A();
    int tal1 = 2, tal2 = 4;
    int res = compute(tal1, tal2);
}
```