Individuell uppgift 1 – Artikel OOP

Klasser

OOP betyder objektorienterad programmering. Allt handlar om objekt. För att definiera och skapa dessa objekt använder mig sig av klasser. En klass bestämmer vilka egenskaper och attribut ett objekt ska ha. Egenskaper är något som kan göras, som ’skriv ut’, ’spara’. Attribut beskriver hur något är, som ’namn’, ’pris’, ’är skickad’ eller liknande. Båda dessa typer kallas gemensamt för instansvariabler. Klassen fungerar som en mall vid skapandet av objekt. I klassen finns en speciell metod kallad konstruktor, och det är den som skapar, eller instansierar, objekt.

Objekt

Det är objekten som bygger ett program. Varje objekt gör sin lilla del i det stora hela. När man skapar ett objekt bestämmer man vilken information det ska ha, utifrån de attribut som finns genom klassen. Har man attributen ’namn’ och ’pris’ t.ex. så kan man sätta dessa till ”Hammare liten”, ”250”. Sedan skapar man nästa objekt, från samma klass, ”Hammare stor”, ”275” osv.

Komposition

Säg att vi har ett objekt kallat ”Bil”. Detta objekt har andra objekt från andra klasser som attribut, som ”Motor”, ”Växellåda”. Tar vi bort ”Bil”, då försvinner även ”Motor” och ”Växellåda”. De kan inte existera utan ”Bil”.

Arv

Arv är ett ännu starkare ägande än vad komposition är. En klass som ärver från en annan klass får alla dess attribut och metoder. Den ärvda klassen kan man sedan göra sina små modifikationer i. Exempel:

Klass Produkt har attributen ”namn”, ”artikelnr”, ”beskrivning”, ”pris”, ”vikt”. Men nu har vi börjat sälja digitala produkter! Då kan man skapa en ny klass, DigitalProdukt som ärver från Produkt. Alla attribut hänger med, men vi väljer att inte använda oss av ”vikt” utan skapar ett eget attribut ”storlek” eller vad man nu kan behöva.

Inkapsling

Man kan kapsla in, gömma, instansvariabler genom att göra de privata, sätta dem till ”private”. Då kan dessa endast ändras inom klassen. Det fina här är att man kan göra metoder i klassen som är publika, ”public”, som i sin tur kan ändra på instansvariablerna som andra klasser ibland behöver kunna göra.

Att arbeta på detta sätt gör att stora kodmängder inte blir synliga när objekt från klassen används, utan endast det som är nödvändigt. Som användare skickar man kanske in ett värde, och får tillbaka ett annat. Hur eventuell beräkning görs syns inte. Användaren behöver bara veta vad för typ av värden som kan skickas in.

Polymorfism

Polymorfism betyder mångformighet. En klass som ärver från minst en annan är polymorfisk.

Exempel på method overloading är

addera(1, 1); Här returneras 2, summan.

addera(1, 1, 1); Här returneras 3, summan.

addera(1.5, 1.5); Här returneras 3, summan talen med en annan datatyp.

I klassen finns dessa tre metoder beskrivna, alla med samma namn, men med olika antal argument och datatyper.

En annan typ är method overriding, där en ärvande klass kan ha samma namn på en metod som superklassen, men där den ärvande klassens metod kommer att ”köra över” superklassens metod.

Polymorfism är väldigt bra att använda sig då man har många objekt som ska göra ungefär samma saker. Det underlättar kodandet, minskar kodmängden.