Olika programmeringsparadigm

**Programmeringsparadigm**

Ett programmeringsparadigm är ett sätt att strukturera och sätta en sorts idealprocess för kodning. Programmeringsparadigm skiljer sig från det vetenskapligt myntade begreppet paradigm, detta då man inom vetenskapsvärlden beskriver ett paradigmskifte likt en händelse som påverkar sättet man genomför något på stor skala, det kan därför inte ske flera paradigmskiften samtidigt. Inom programmeringsparadigm-termen existerar dock flera paradigm i samtid.

**Objektorienterad programmering**

Objektorienterad programmering hanterar objekt, alltså en instans av en klass, i fokus. Man kan inom objektorienterad programmering skapa ett program som hanterar flera objekt som kan integrera med varandra. Objektorienterad programmering byggs upp av vissa grundkoncept; arv, polymorfism, klass och inkapsling. Arv betyder att klasserna och begreppen är beroende av varandra i stora fall, och kan samspela som en kedja sammanhängande, oftast generella, begrepp. Polymorfism innebär att ett program kan bestå av flera subklasser till en superklass där subklasserna kan hanteras och användas som en instans av superklassen. Detta betyder att subklasserna kan använda sig av ett gemensamt gränssnitt som definieras i superklassen, medan subklasserna skiljer sig när det kommer till implementering av en metod men ändå kan anropas på exakt samma sätt. Klass betyder att programmet själv består av klasser som byggs utefter de begrepp som behöver hanteras. Alla klasser hanteras som en abstraktion av det begreppet. I stora drag består ett program av objekt som samspelar mellan varandra och tillhör klasser. Inkapsling är ett koncept inom objektorientering som binder samman data och funktioner som manipulerar data och håller båda säkra från yttre störningar och missbruk genom att kunna dölja data genom datainkapsling.

De flesta språk som hanterar objektorienterad programmering är klassbaserade, och många av dessa kan vara multi-paradigm, det vill säga att de i stort sett kan användas inom flera paradigm där flertalet mer eller mindre hanterar objektorientering. Exempel av språk som hanterar objektorienterad programmering är Java, Python, Ruby, Perl och PHP. Exempel av multi-paradigmsspråk är C++, Java och Python.

**Logikprogrammering**

Logikprogrammering (logisk programmering) är ett paradigm som i stora lag baseras på ren formell, matematisk logik. Program som programmeras enligt logikprogrammering består av en uppsättning meningar i en logisk form som uttrycks enligt fakta och regler kring dess problemområden.

Logikprogrammering, likt objektorientering och alla programmeringsparadigm, består av grundkoncept. Ett av dess koncept är logik och kontroll, detta innebär att logisk programmering kan ses som kontrollerat avdrag. Ett viktigt begrepp inom logisk programmering är separering av program i sin logikkomponent och deras kontrollkomponent. Med rena logiska programmeringsspråk bestämmer den logiska komponenten själv de lösningar som produceras och skapas. Regler inom logikprogrammering skrivs som logiska klausuler med head och body. till exempel "H är sant om B1 och B2 är sanna." Fakta uttrycks på ett liknande sätt som regler dock utan en body; till exempel "H är sant". Logikprogrammering är en underkategori av deklarativ programmering som då är ett samlingsnamn för både logikprogrammering och funktionell programmering. Exempel på logiska programmeringsspråk är Prolog och Alice.

**Funktionell programmering**

Funktionell programmering beskrivs likt logikprogrammering som deklarativ programmering, detta då det är gjort genom uttryck och deklarationer istället för statements. Funktionell programmering är en sorts utformning av att bygga upp strukturen och elementen i ett program som behandlar beräkning som en utvärdering av matematiska funktioner och då undviker changing state och mutable data. Funktionell programmering är idempotent på så sätt att funktionen i frågas returvärde endast beror på dess argument. Detta betyder att om man kallar på en funktion med samma värde för ett argument så kommer det alltid bli samma resultat. Ett exempel på ett strikt funktionellt programmeringsspråk är Haskell.

**Procedurell programmering**

Procedurell programmering härleder från strukturerad programmering där man strävar efter att koden ska separeras i kodblock. Procedures, alltså rutiner, subrutiner eller funktioner, innehåller en rad beräkningssteg som ska utföras. Vilket procedure som helst kan kallas när som helst under ett programs genomförande, inklusive genom andra procedures eller sig själv. Exempel på ett procedurellt språk är BASIC, Pascal och C.

**Aspektorienterad programmering**

I databehandling är aspektorienterad programmering ett programmeringsparadigm vars syfte är att öka modulariteten genom att tillåta separation av tvärgående problem. Detta sker genom att lägga till ytterligare beteende i befintlig kod utan att ändra koden i sig, utan att i stället specificera vilken kod som är modifierad via en "pointcut"-specifikation, till exempel "logga alla funktionskallningar när funktionens namn börjar med ’test’". Detta möjliggör beteenden som inte är centrala för affärslogiken (t.ex. loggning) som ska läggas till ett program utan att ”klyva” koden, alltså kärnan till funktionaliteten. Aspektorienterad programmering utgör en grund för aspektorienterad mjukvaruutveckling. Ett exempel på en extension som använder sig av aspektorienterad programmering är AspectJ, en extension till Java som inkapslar uttryck i en klass kallad Aspect.