Begrepp i Programmering

**Variabel**- innehåller ett värde, representerar något

**Referensvariabel**- refererar till objekt.

Värdeanrop- argumentens värde kopieras till lokala kopior i metoden. T1.sätt(a,b,c). Vid värde anrop av referenstyper kopieras referensen och inte objektet.

**Initiering**- ge/tilldela en variabel ett värde. Kan vara arrayer, m.m.

**Objekt**- ”förpackning” med olika variabler som hör ihop och definierar ofta något. Objekt kan skapas i klasser.

**Satser**- få programmet att göra något.

**Deklarera**- ange vilken datatyp- String, double, int.

**Logiska** uttryck- && / == jämföra true eller false.

**Val/selection**- använd if satser för at bestämma vad som ska göras.

**Instansvariabel**- variabel deklarerade i ett block i ett objekt. Hör samman med klassen. Deklarerar utanför metoderna. Kan ses i hela filen. Kan ha samma namn som lokala variabler.

**Klass**- introducerar en klasstyp, en referenstyp. Nya klasser= nya typer. Kan deklarera variabler av denna typ och sedan initiera den genom new-operatorn tillsammans med metoden som heter som klassen. En klass beskriver vilka variabler som ingår i förpackningen. Vi måste skapa en klass och därefter skapa objekt utifrån klassen. Vi måste kunna ange typ för alla variabler.

**Punktnotation**- för att komma åt enskilda variabler.

**Sidoeffekt**- icke uppenbar förändring som uppstår när en instruktion i ett program utförs. Ett exempel är att en instruktion ändrar en variabel som också används av andra delar av programmet.

**Initiering**- ge variabeln ett start värde.

**This**- en konstant referens till sig själv, används explicit vid namnkrockar. Annars underförstått.

**Konstruktor**- en metod som anropas vid instansiering av klasser, initierar värden för nya objekt variabler. Klassvariablen tilldelas värdet av konstruktorn.

**Instansmetoder**- metoder deklarerade i en klass, måste finnas i ett objekt. Kan anropas på alla objekt skapade utifrån klassen.

**Icke muterbara** – om ordet final placeras framför ett objekt, variabel etc. så kan inte det initierade värdet ändras utan det är satt.

**Överlagring**- metoder med samma namn men olika parametrar. Overload. Objektets typ bestömmer

**Överskuggning**- metoder med samma namn och samma parametrar. Override: Variabelns typ bestämmer.

**Annotation**- @Override

**Static variabler**- tillhör klassen(klassobjektet)

**Metod**- koden innanför blocket kallas kroppen. Första raden kallas huvudet. Avgränsad del av ett program som utför en viss uppgift.

**Funktionell abstraktion**- anta att man har en metod som gör det man vill åstadkomma.

**Subtyp**-Om man istället för värden ur en aktuell typ, A, kan använda värden ur en annan typ, B, på en godtycklig plats utan att man åstadkommer otillåtna operationer (kompileringsfel) kallas B en subtyp till A. Alla typer är subtyper till sig själva.

**Supertyp**- kan

**Stack**- ett lagringställe för lokala variabler och metoder. Kan bli fullt, StackOverFlow.

**Heap**- här sparas objekt och instansvariabler. Större än stacken i minne.

**Åtkomst**- public, private. Avgör om bara metoden/ klassen kan komma åt eller alla. Använd punknotation för att komma åt enskilda varaibler.

**Kompilator**- ett program som översätter källkod till körbar byte-kod. Kontrollerar att vi skriver rätt annars kompileringsfel.

**Instans**- ett objekt av en klass/typ. Objekt är en instans av en klass. Recept= klass, kaka=objekt.

**Insansmetod**- en metod som hanterar en enskild instans av en klass.

**Instansiera**- skapa ett nytt objekt i samband med deklaration och initiering.

**New-operatorn-** instansierar ett objekt.

**Array**- många variabler av samma typ. Är konsekutiv = inga tomrum. Deklareras av typ och namn + ev. initiering. Referenstyp.

**Indexering**- Komma åt enskilda variabler/elementen i arrayen. Om index utanför arraylängden exekveringsfel

**Taversering**- genomlöpning/ gå igenom alla element. Ofta med en for-loop.

**Tilldelning**- kopiera från höger sida till vänster

**Referenslikhet**- två variabler refererar till samma objekt

**Matris**- array av arrayer/2D-arrayer. Deklareras med dubbla []-parenteser. Har rad och kolumn. Referenstyp.

**String**-

**Generiska** **metoder**- kan bara ta in referenstyper.

**Association**- när två klasser har med varandra att göra. T.ex. om den ena har en instansvariabel som den andra klassen också använder.

**Typomvandling-** implicit sker automatiskt från super till sub. Explicit måste fixa själv, skriva ut. Måste finnas en super/sub relation. Alltid super = sub annars fungerar inte omvandlingen!

**Typsystem-**

**Referens-**

**Likhet-** variabler med samma värde

**Parameter-**

**Klassvariabel-** tillhör inget objekt utan delas av alla objekt av samma klass anges med static vid deklaration. Bör vara final för att undvika större problem.

**Lokal variabel**- variabel i en metod

**Indentering**- dra in texten lite

**Inkrementering** – öka värdet på variabeln ++.

**Derkementering** – minska värdet på variabeln --.

**Uppreping**/interation- while sats.

**Maska** ut siffror ur heltal- % & /.

**Boxing**- göra en primitivtyp till ett objekt, Integer obj= new Integer (”123”);

**Unboxing**- göra ett objekt till primitivtyp, Int i = obj.intValue();

* Public- synlig föra alla klasser i programmet
* Statiskmetod- klassmetod, inte speciellt för ett visst objekt utan tillhör klassen.
* Void- ger inget returvärde.
* Parameter- argument som kan skickas med in i en metod eller i en konstruktor.
* Satser- det som beskriver vad en metod ska göra.
* Reserverade ord- ord som har speciell betydelse i Java, kan inte användas för annat än vad de har för betydelse. OBS! Små bokstäver
* Import- för att importera paket i programmet en \* används för att importera allt i paketet. Java.lang importeras alltid automatiskt
* Modifierare- ange speciella egenskaper ex. final, private, protected osv.
* Variabler- "behållare", lagrar olika typer data.
  + Enkla variabel- lagar värden/data. Bara en lagring av en typ.
  + Referensvariabel- variabel som refererar till ett objekt.
  + Instansvariabel- variabler deklarerade i en klass, tillhör klassen
  + Lokal variabel- variabel som finns deklarerad i en metod, finns inte utanför metoder, dom är tillfälliga
  + Klass/statiska variabler- gemensamma för alla objekt som tillhör en klass.
  + Konstanta variabler- variabler deklarerade med modifieraren final, värdet på dessa går inte att ändra.
* Deklarera- ange en typ och ge ett namn
* Tilldelning- variabelnamn till vänster om = och till höger ett värde som ska kopieras till variabeln
* Initiera- tilldela ett startvärde för en (variabel?) om detta inte görs antas ett defaultvärde utifrån typen
* Initeringsvärde- det värde man ger som startvärde för en variabel.
* Metod- avgränsad del av ett program som utför en viss uppgift.
  + Instansmetod- en metod som tillhör en klass.
* Indentering- indragningar i programmet för lättare läsligt.
* Jämförelse- ==
* Objekt- avgränsad enhet med egna "uppgifter"
* Klass- beskrivning av objekt med samma egenskaper ser ut. Kortfattat en beskrivning.
* Instans- ett objekt som tillhör en viss klass utgör en instans.
* Defaultvärde- värde som automatiskt tilldelas om man inte initierar. Ex. 0 eller null.
* Modulo- för att få ut rest av division
* Inkrementering- öka värdet med ett ++
* Dekrementerieng- minska värdet med ett --
* Cast- explicit typomvandling
* Argument- vid metodanrop skickas argument med s.72
* Call by value- argumentens värden kopieras till lokala variabler i en metod, vid referenstyper kopieras referensen.
* Detta innebär att om vi skickar eg. 5 som argument till en metod, så är det just värdet 5 som metoden tar emot. Ingenting metoden gör med detta värde kan påverka någonting utanför metoden självt͘
* Call by reference- Detta innebär att det är själva referensen som skickas till metoden, och samtliga förändringar som görs på objektet som referensen pekar på kommer att synas även utanför metoden, om anroparen har kvar ett alias till objektet i fråga
* Uttryck- metod som inte har void som typ, returnerar ett värde
* Sats- metod som inte lämnar något resultat
* Konstruktor- en initieringsmetod när man skapar objekt
* Överlagrade metoder- metoder med samma namn men olika parametrar(Overload). Vilken metod som väljs avgörs statiskt vid kompilering.
* Wrapper classes- ex Interger, Double osv. innehåller diverse användbara klassmetoder. Bra då referensvariabler bara kan referera till objekt och inte enkla variabler. Wrapper classes räknas som ett objekt man kan referera till.
* Boxing/unboxing- kapsla in variabler i deras wrapper klasser.
* Array- samling med olika värden av samma typ. Varje "plats" i en array är numrerad från 0-->
* Listor- behöver inte ha en bestämd storlek. Innehåller bara referenser därför kan man inte deklarera en lista men ex. int utan måste använda sig av Wrapper klasserna istället
* Enum/uppräkningsklasser- innehåller olika värden med "möjliga scenarion"
* Stringbulider- Används om man inte har parallella trådar som kan påverka texten man redigerar. Ett objekt av klassen säger man är en behållare med placerade char tecken i.
* Matris- flerdimensionella arrayer. Måste ha dubbla for satser för traversering.