**Föreläsning F2 – 04/09/25**

**Introduktion till systemutveckling och programmering**

**Vad är problemet?**

För att förstå vad det är för domän man är på

**Hur löser vi problemet?**

Vad är syftet

Vad ska göras

Hur är det uppbyggt

Vad är där för begränsningar

**Informationssystem**

Behövs inte vara digitalt (läs anteckningar)

Går att koppla till administrativt arbete

Där vi håller reda på löner, studentinformation

Hittas överallt i vardagliga livet

Har utvecklats till att hjälpa vid tågspår eller inte andra världskriget

Dök upp mer frekvent för ungefär 100 år sen

Fortfarande relativt ”nytt”

**IS är system**

A diagram of a system

AI-generated content may be incorrect.

Handlar om att ta data och göra till begriplig information

**System**

Ett system är en uppsättning av interagerande komponenter

Arbetar även tillsammans för samma mål

Ett system sett utifrån:

* Mål
* Systemgräns
* Omgivning
* Input
* Output

**Kaffeautomat**

Mål:

* Att göra kaffe

Systemgräns:

* Apparaten

Omgivning:

* Den som köper
* Var den står

Input:

* Knappar
* Val av dryck
* Mängd
* Kaffe
* El

Output:

* Kaffe
* Återkoppling till användare
* Kvitto

**Datorbaserade system och verkligheten**

Systemet är inte verklig, det är vår representation

Kan ha fysiska delar, fortfarande vår representation

Baserat på kontext och perspektiv

**System och subsystem**

Systemet har en omgivning

Kan samarbeta med andra system (subsystem)

Dyker upp i bl.a kod

Subsystem kan vara fristående från helheten

Kan vara digitala, inte nödvändigt

En organisation kan ses som ett enda system med flera subsystem

Används för att stötta IS och underlätta

**Problemlösning och projektmodeller**

Generisk modell av problemlösning:

Samma information om problemet

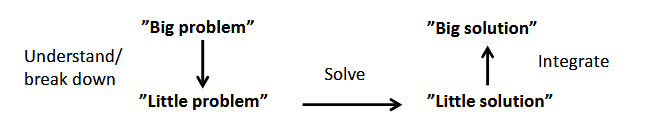
Analysera och potentiellt omformulera det

Undersök efter möjliga lösningar

Hitta en lösning

Implementera den valda lösningen

Ta ett stort problem och dela upp det till mindre problem



**Varför Java?**

Väldigt grundläggande för många back-end-system

Ett av de mest använda språken inom programmering

Lätt att känna till i andra språk

Robust och fungerar bra för nybörjare, inte lika effektivt men mer förlåtande

Python är inte fullt objektorienterat

LTS (Long term support)

Java versioner är väldigt länge underhållna (Java 8 fram till 2030)

Fortfarande flera företag som använder sig av det

Oracle försökte licensiera Java 17 under 2020 och 2021

Java och JavaScript delar rötter i syntax

Java är ett språk som måste kompileras, tillskillnad från JavaScript

JavaScript interpreteras vilket har en betydelse vilken ordning koden står i

Java kompilerar och då sköts interpreteringen samtidigt

**Vad är en dator?**

Babbage – Datorns fader

Skapade mekaniska beräkningsmaskiner

Anlytical Engine, kunde programmeras med hålkort

Avancerad att det inte gick att realisera pågrund av materialbegränsningar

Alan Turing – definierade vad en modern dator innebär

Allt vi kan göra i en dator idag kan emuleras på en Turing-maskin

Såhär fungerar fortfarande datorer idag, mer digitaliserat

**Vad är ett programmeringsspråk?**

**Ada Lovelace** – den första programmeraren

Arbetade med Anlytical Engine med Babbage

Krediteras med att ha skapat det första algoritmerna

Algoritm: En lösning på ett återkommande problem

1800 talet

Vid andra världskriget när elektronik blev introducerat, mer digitalisering

Programmerades för hand genom att koppla om sladdar

Programmerarna på Blechley Park hölls hemliga i årtionden

Ett av de hemligaste projekten

Maskinkod består av siffror, representeras med ettor och nollor

Inte lästbart för en människa i längden

**Assembly**

Lite lättare att förstå

Kallas som låg-nivåspråk

Används för att konvertera koden till maskinkod

**Grace Hopper**

Introducerade programmeringsspråk på 50 talet som använde engelska ord

Det vi kallar för Hög-nivå-språk idag

Är det vi jobbar med i de vanligaste språken

A diagram of a computer network

AI-generated content may be incorrect.

Ju högre upp desto äldre språk är det

Cobol används fortfarande i vissa banksystem

Smalltalk ett av de första renordade objektorienterade språken

C++, C#, Java väldigt lika i hur de skrivs (familj)

Python är egentligen skapat för datorhandling, används numera för html

Källkod/source code är det vi kan läsa

Datorn kan ej läsa, måste översättas till maskinkod

* C-stil

1. Källkod -> Kompilera -> Object Code, förstås av datorn

* Java

1. Källkod -> Kompilera > Bytecode som vi lagrar programmet i
2. Nu vill vi använda vårt Java-program:

JVM läser byte-koden och interpreterar denna till Object Code

JVMen vet om vad du har för kompententer i din dator och använder det vid att göra Object Code

Java översätter i 2 steg, gör det mer portabelt

Negativa är att det är lite långsammare

Programmet (kommando) javac är kompilatorn och programmet java startar JVM som tolkar och exekverar byte-koden

**Syntax**

Grammatiken för hur vi skriver kod

* Tillåtna tecken och vart

Bryter vi mot syntaxet då kommer kompilatorn ge felmeddelande

Kompilatorn visar VART det har dykt upp fel, inte vad som är fel

**Semantik**

Semantik är vad orden som är tillåtna i syntaxen har för innebörd

När andra ska lösa din kod, gör det lättare om du har bra semantik

Semantiken för datorn är irrelevant

Syntax begränsar vad vi kan skriva

Javas nyckelord: main, class, if, else, while

Reserverade ord I språket, kan inte använda för saker du namnger själv

Java är case-sensitive

**Logik**

Hur vi organiserar delarna i vårt program

Kan ha korrekt syntax och vettig semantik, men ändå dålig logik

Svårt att hitta vid logikfel

**Kod**

Källkod består av satser (statements) som är instruktioner till datorn

Varje sats utgör en beräkning som utförs

I Java avslutas en sats med ;

Satser kan delas in i block av kod {}

Indentering är indrag av rader för att göra det lättare att följa block av kod

I python använder man bara av indenteringar

Kommentarer används som förklaringar, ignoreras av datorn

Kommentar på en rad //