MODULE 1A – COMPUTER PROGRAMMING

ITSE 1003

Introduction to Programming Languages

ITSE 1003 Goals

- Understand the context for programming
- Learn some principles of software development
- Learn some core constructs of all programming languages
- Learn basics of Java programming, including tools

Context for Programming

- Why we program
- What are computer systems

Why do we program?

Generally …

- Solve a problem
- Perform a ***well-defined*** task
- …

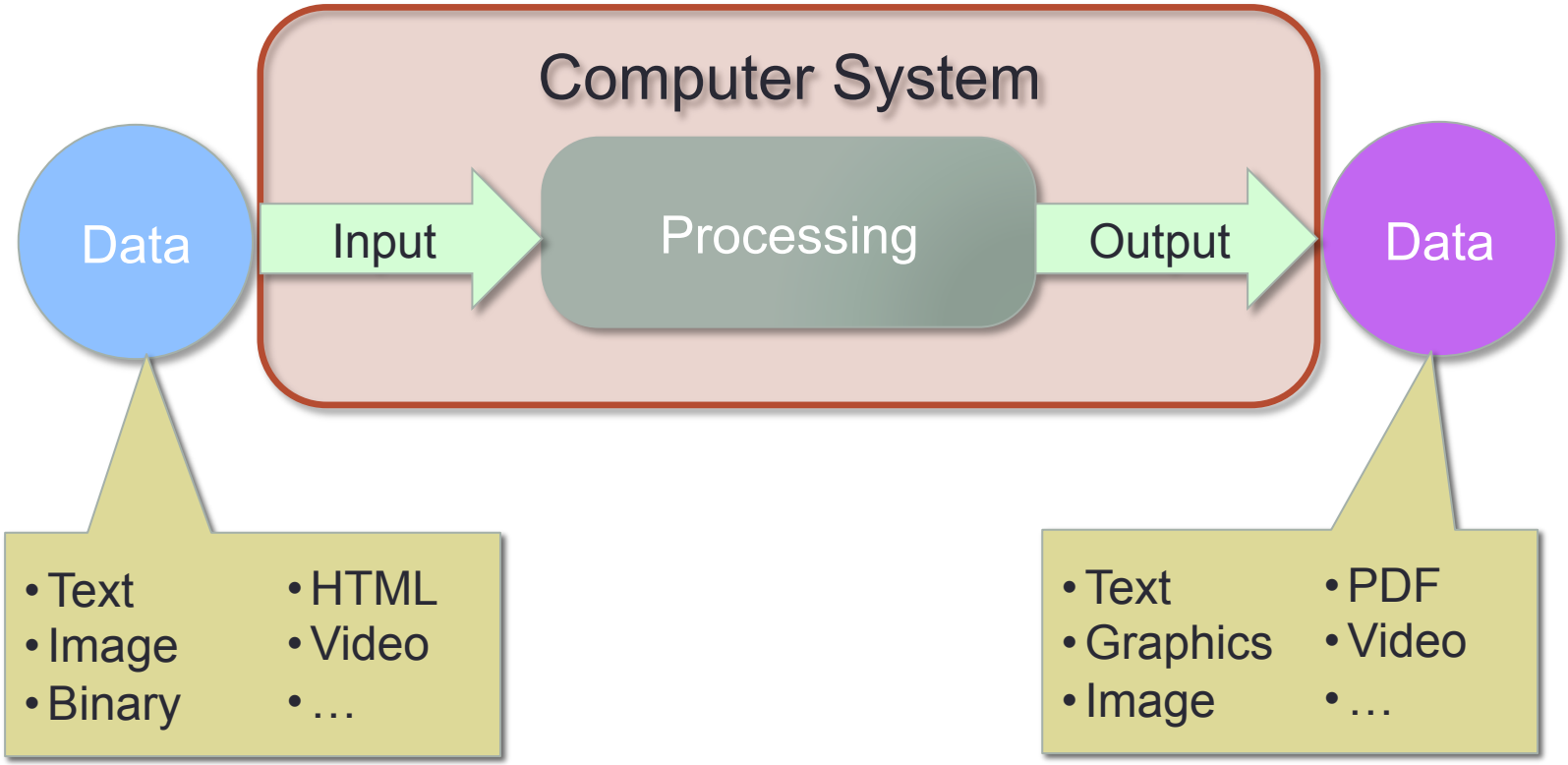
Specifically …

- Sell books, clothes, …
- Do payroll
- Play games
- Perform research
- Support the above tasks!
- …

Programming requires …

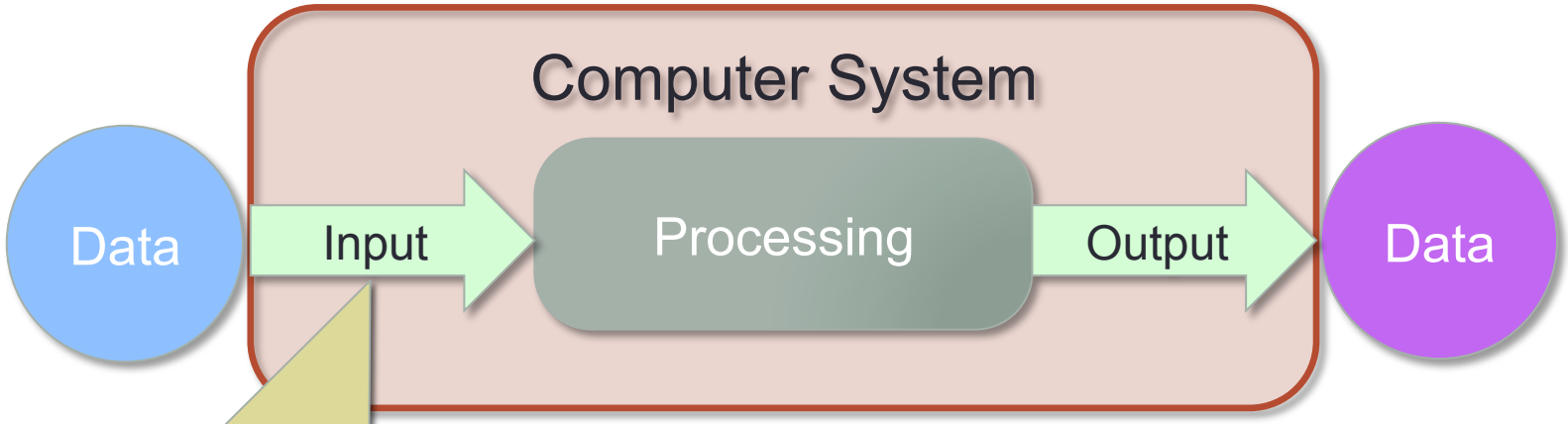
- A computer system
  - Perform the task
- Programmers … that's you!
  - Identify the problem
  - Define the task
  - Write the program
- Tools
  - Support the programmer

Computer system overview



Sometime the *output data* is called ***information***.

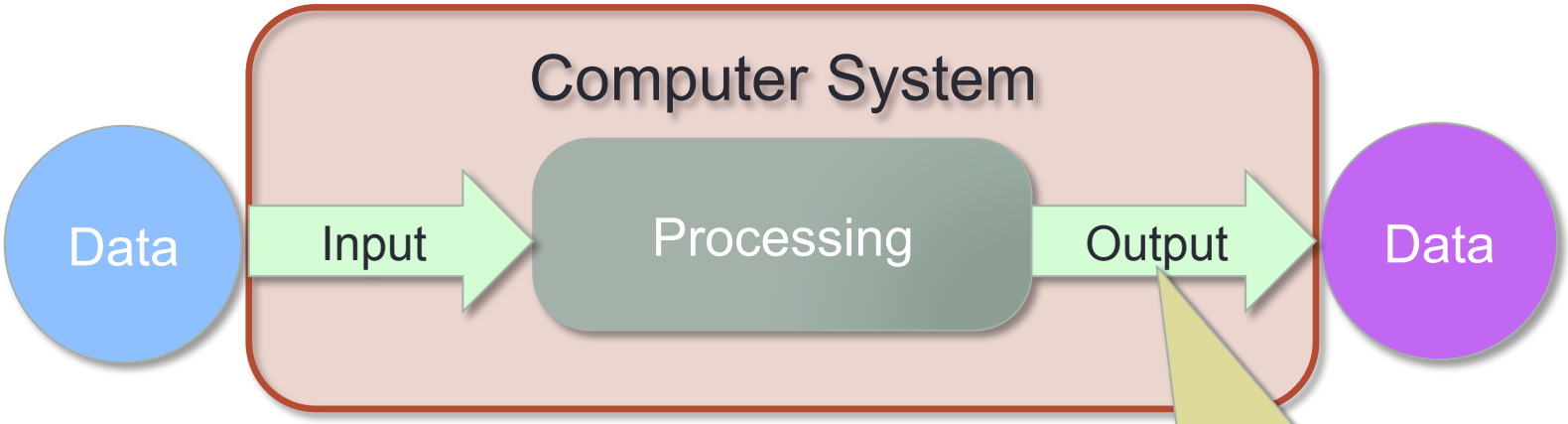
Computer system – input “devices”



Input Device/peripheral/channel

- ***Network*** – most important today
- Persistent storage
- Keyboard, Mouse
- Touch screen
- Scanner
- Microphone
- Camera
- …

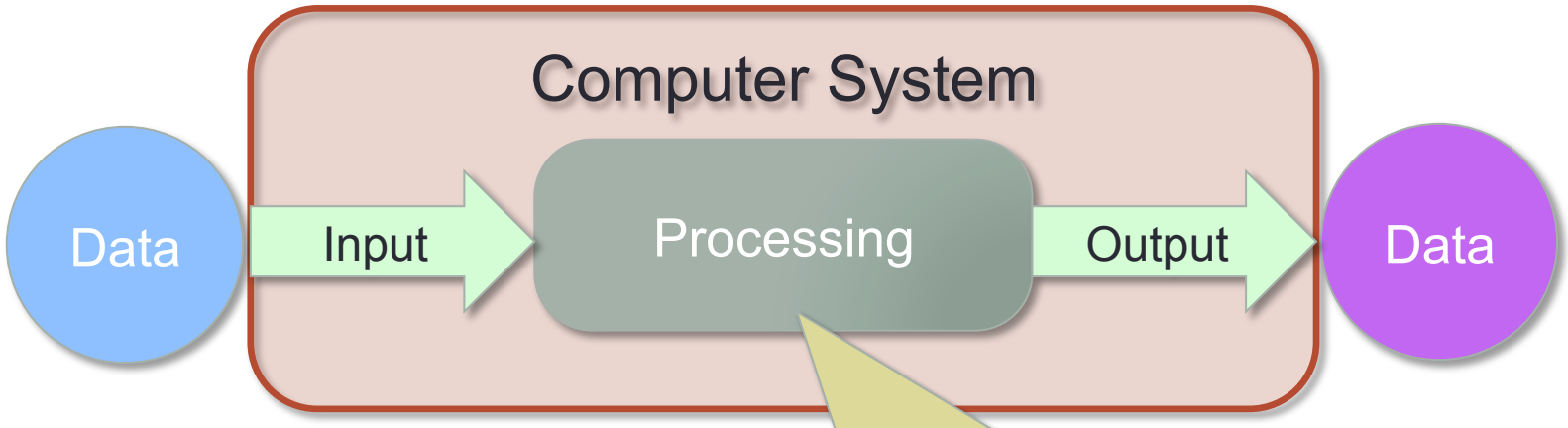
Computer system – output “devices”



Output Device/peripheral/channel

- ***Network*** – most important today
- Persistent storage
- Display
- Speaker
- Printer
- …

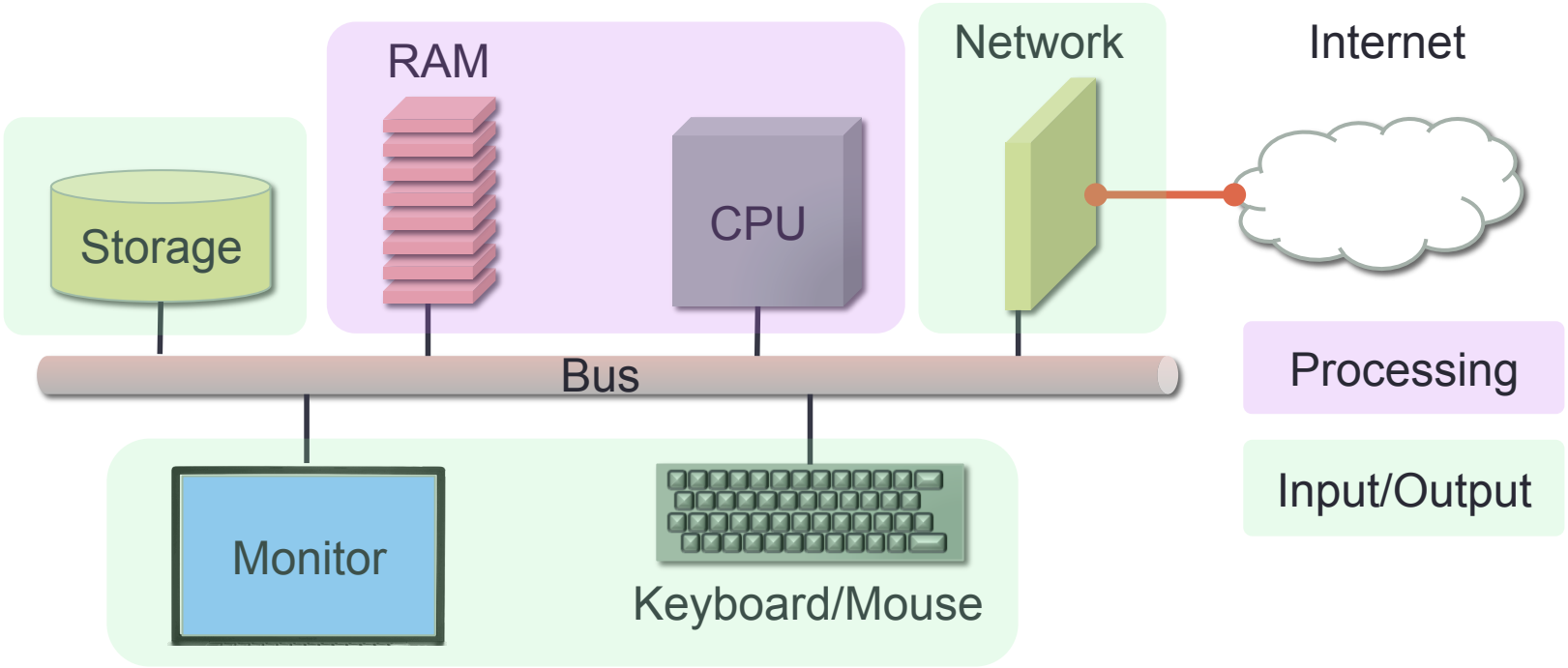
Computer system – processing



Central Processing Unit

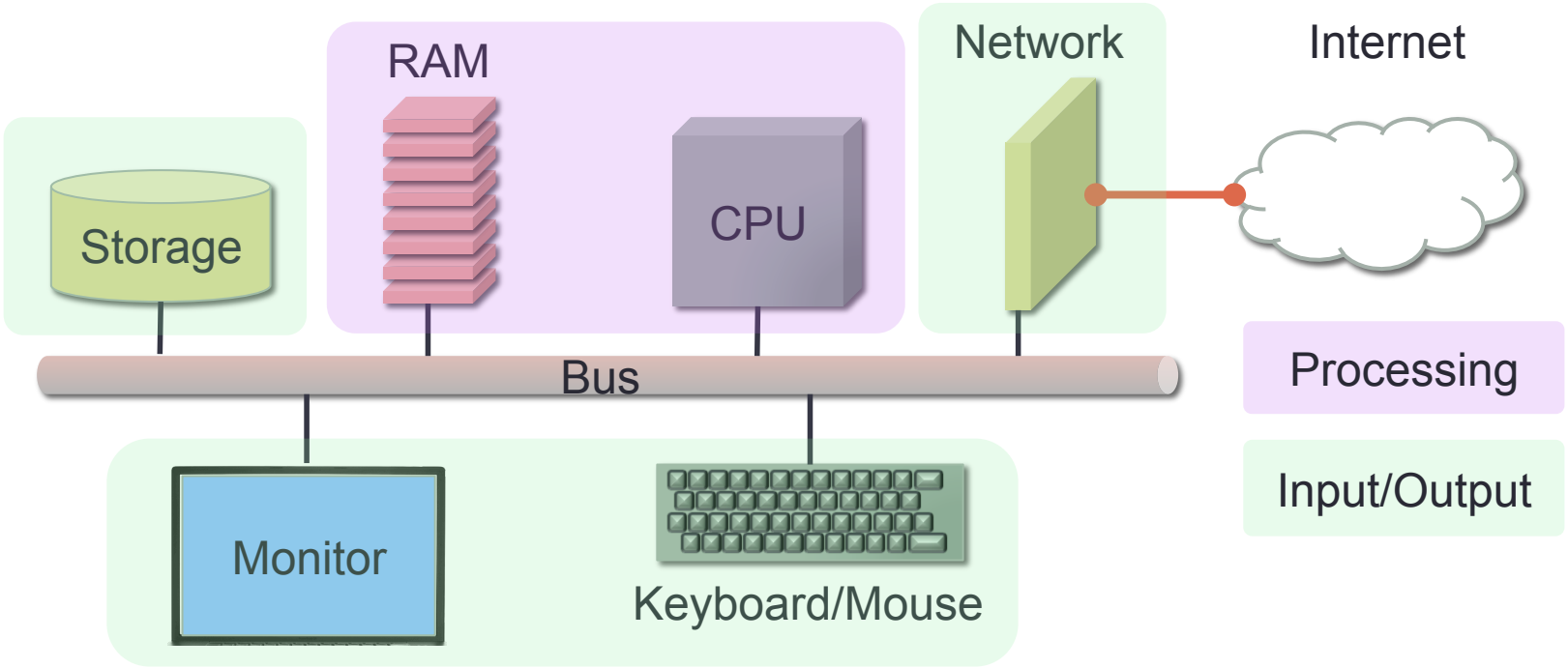
- ***Transforms/translates*** data into a different form
- Acquires input
- Performs operations
  - Mathematical
  - Comparison
  - Control (e.g., branching, looping)
- Produces output

Computer system drill down (simplified)



- CPU controls entire system
- RAM used for CPU instructions and data being processed
- Keyboard/Mouse and Monitor enable user interaction
- Storage and Network allow local and worldwide I/O

Computer system drill down (simplified)



Simplifications

- RAM can actually be in multiple layers called caches
- There can be buses for various purposes (e.g., main, storage, USB)
- There are many more forms of I/O devices
- Most I/O devices have “controllers” that offload CPU (e.g., graphics)

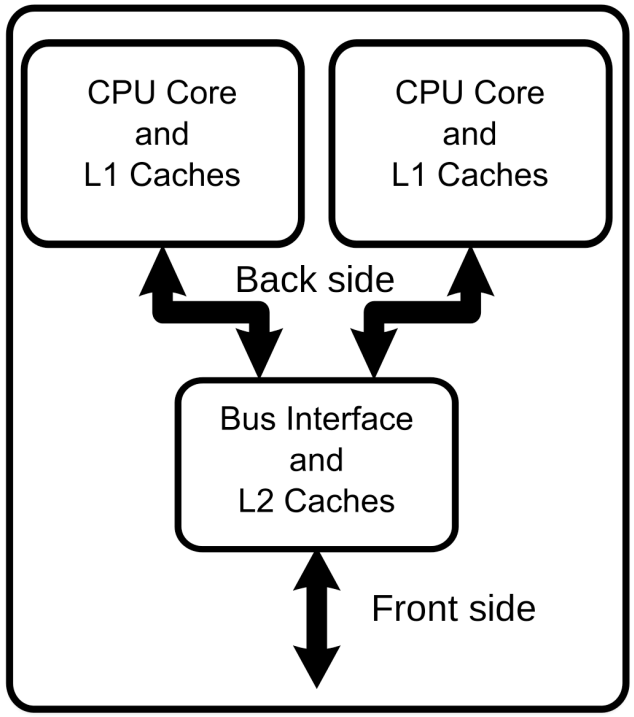
CPU

- Control Unit (generalized control)
  - Retrieve instruction from memory
  - Decodes instruction
  - Executes, or schedules other units to execute, instruction
- Other units
  - ALU – arithmetic logic unit (integer arithmetic, comparison, logical functions)
  - FPU – floating point unit (floating point arithmetic)
- Instruction set (atomic operations)
  - Load memory to register; store register to memory
  - Add registers; subtract registers; increment/decrement register
  - Compare two registers
  - Branch on condition

CPU

- Modern processor modules can contain multiple CPUs (called cores)
  - Share memory
  - Operate independently
  - Increases system performance

CPU Manufacturers
- Intel
- AMD
- IBM
- …



Bits and bytes

- Today's computer systems are digital (versus analog)
- The smallest unit of information in a binary system is a ***bit***
  - Derived from binary digit
  - Can represent 0 or 1
  - Common way of measuring communication speeds
- The smallest unit computers address is a ***byte***
  - Intentionally misspelling of "bite"
  - 8 bits
  - Can represent 0 to 255 (2^8-1) or -128 (-2^7) to 127 (2^7-1)
  - Convenient for encoding characters (ASCII)
  - Common way of measure memory or storage size/capacity

Bits and bytes

- Sizes & speeds measured in multiples of 2^{10} = 1024 bytes

Size in bits (b) or bytes (B)	Prefix
$1024^1 = 2^{10}$	K … kilo
$1024^2 = 2^{20}$	M … mega
$1024^3 = 2^{30}$	G … giga
$1024^4 = 2^{40}$	T … tera
$1024^5 = 2^{50}$	P … peta
$1024^6 = 2^{60}$	E … exa
$1024^7 = 2^{70}$	Z … zetta
$1024^8 = 2^{80}$	Y … yotta

Examples
- 16 GB memory
- 1 TB storage
- 1 Gb network

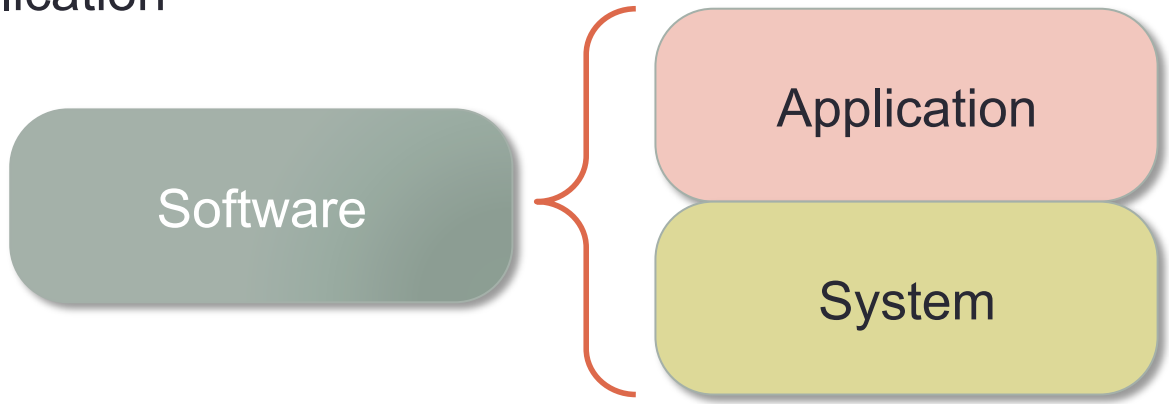
32 bit versus 64 bit architecture

- Modern computer systems are said to support
  - 32 bit architecture
  - 64 bit architecture
- This impacts
  - Instruction size
  - Memory addressing capability
  - Integer size and related arithmetic capability (native)
- Arguments still abound about which is better/faster
  - Some CPUs can do either
  - Some software is written to support either

Computer system – software

- Software is at least as important as the hardware
  - Runs on the hardware
  - Controls the hardware (hardware is “useless” without software)
- ***All*** software created in the instruction set of system CPU

- Software arranged in two layers
  - System
  - Application



System software

- Operating system
  - Manages hardware resources
  - Provides common services for applications
    - Process management, Memory management
    - Interrupts (e.g., I/O)
    - File system, Networking (TCP/IP, UDP)
    - Security (Process/Memory protection)
  - Single or multi-process
  - Single or multi-user
- Graphical User Interface (GUI)
  - Simplifies user interaction with
    - OS
    - Applications
  - Often tightly tied to OS
- Utilities
  - Device drivers
  - Virus protection
  - …

Multi-user OS
- Windows NT, …
- Unix (variants)
- Linux (variants)
  - Mac OS
- …

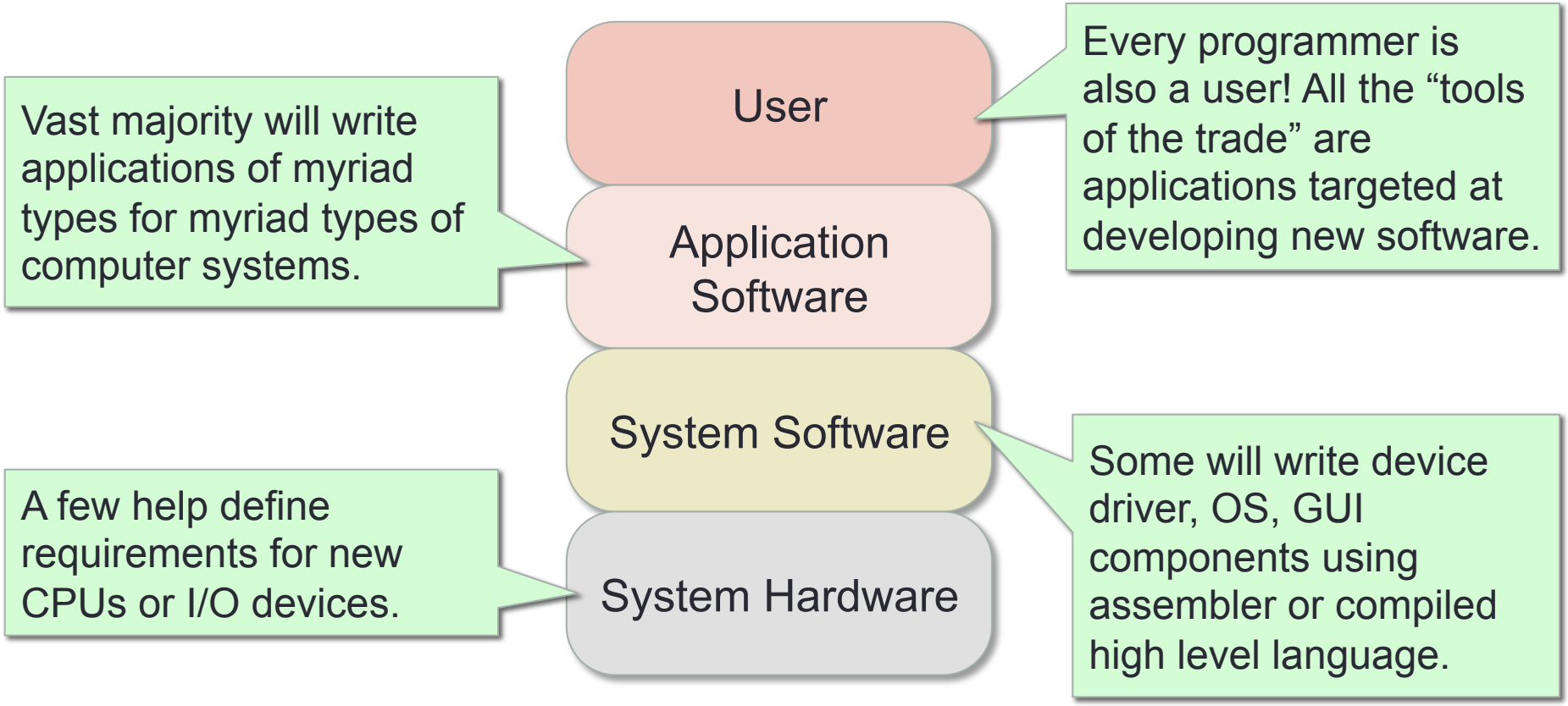
Single-user OS
- Android
- iOS
- Windows RT
- …

Cross Platform GUIs
- X (Unix-like OS)
- …

Application software

- Really why computer systems exist
- Process input and produce output
- Thousands of application programs exist
  - Word processing
  - Spreadsheets
  - Image/graphics editing/creation
  - Games
  - Scientific (weather, health, …)
  - Programming tools!
  - …

The role of the programmer



Interesting observation

- In theory, any computer can solve any problem that's possible to compute, given enough memory and time

- In practice, solving a problem involves ***constraints***
  - Time
    - Weather forecast, next frame of animation, ...
  - Cost
    - Cell phone, automotive engine controller, ...
  - Power
    - Cell phone, handheld video game, ...

Types of computer systems

- There a many different types of computing systems
- Distinguished by
  - Purpose
  - Input/output devices
  - Processing speed
  - Capacity (of input/output and processing)
  - User group size
  - Portability
  - Power & environmental requirements

Types of computer systems

- Calculator
  - Arithmetic processing
  - Some are programmable by user!

Still exist!

- Personal Digital Assistant
  - Calculator function
  - Personal productivity applications
    - Note taking or contact management

Extinct!

Types of computer systems

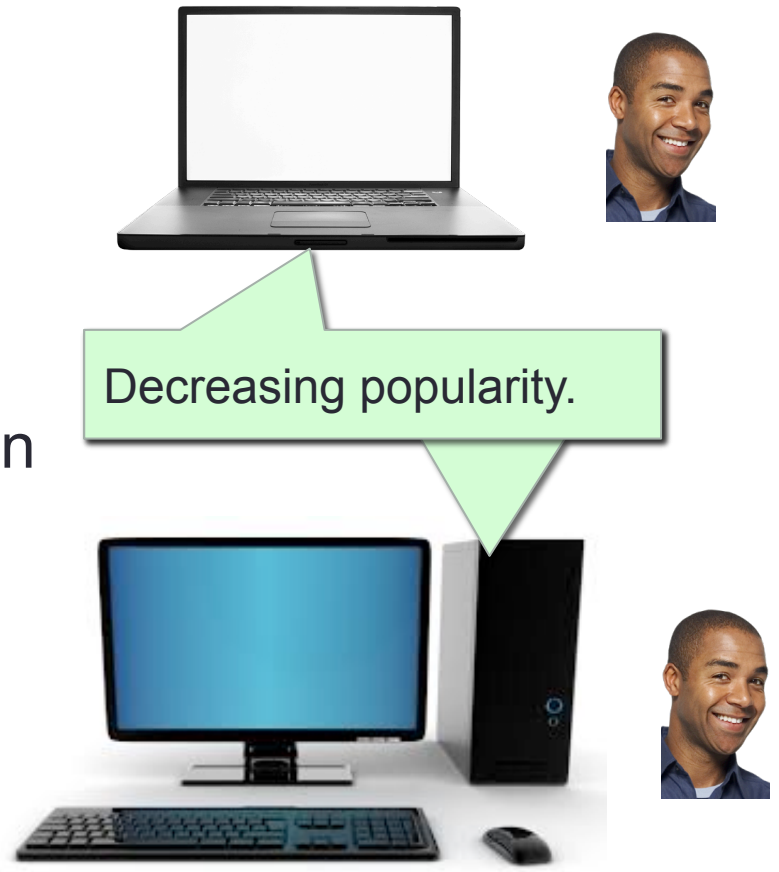
- Smart phone
  - Hybrid of cellular phone and PDA
  - Web access, email access, social media
  - Apps, apps, and more apps

"Shiny bits" currently.

- Tablet
  - All functions of smart phones
    - Except the phone (modulo things like Skype)
  - Plus more
    - Specialized versions of office applications
  - Apps, apps, and more apps

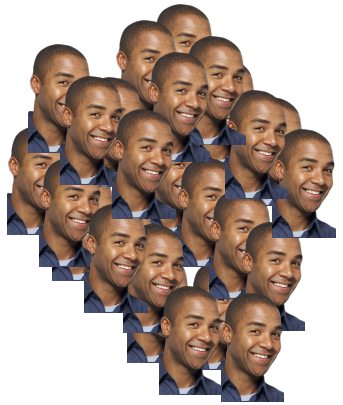
Types of computer systems

- Notebook/Laptop
  - Portable
  - Generalized computing
    - Full versions of office applications
    - Data processing
- Desktop/Deskside/Workstation
  - Non-portable
  - Generalized computing
  - Often some specialization
    - Graphics, e.g., for games
    - Media processing
    - Scientific processing



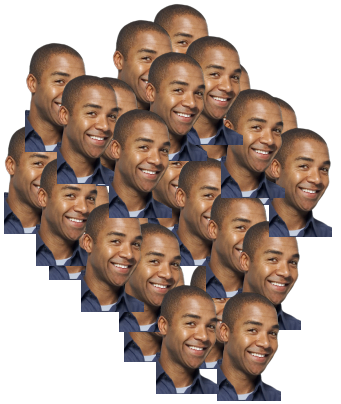
Types of computer systems

- Server [Farm]
  - Multiuser (10^2s to 10^6s)
  - Accessed via network
  - Focused on “backend” tasks
    - Web serving
    - Financial transactions
    - “Big data” processing
- Mainframes
  - Used in large organizations
  - Multiuser (10^3s to 10^4s)
  - Highly secure
  - Enterprise applications
    - Database
    - Organization specific



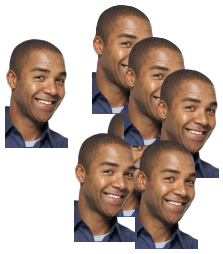
Increasing popularity.

Decreasing popularity.



Types of computer systems

- Supercomputers
  - The most powerful computers made
  - Process trillions of operations per second
  - Found in research organizations, often government sponsored
  - Handle large and complex calculations
    - Nuclear reactions
    - Weather
    - Biology



There are only a few supercomputers in the world. Countries and organizations compete for bragging rights on which is the fastest.

Some interesting additions

- A $35 computer (Raspberry Pi)
  - Credit card size
  - Requires I/O devices
  - Linux, X Windows

- A modern automobile
  - 50+ computers/controllers
  - Performing
    - Climate control, alarm system
    - Automatic transmission, anti-lock brakes
    - Engine management (fuel injection, ignition, …)
    - Satellite navigation
    - Air conditioning, Airbags, …

Principles of Software Development

- Building high quality software is very difficult
- The course examines
  - Methods and concepts of programming
  - Strategies for building real software that addresses real problems

- Software engineering
  - “The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software….” [IEEE]

Principles of software engineering

- Separation of Concerns (& Modularity)
  - Modularity – create components based on function & responsibility
  - Abstraction – separate behavior from implementation
  - Limiting the “need to know”
  - “Black box” methodology
  - Improves development speed, increases agility
- Anticipation of Change
  - Generality – no unnatural restrictions
  - Requirements will change
  - Time-to-market key (no “big bang”)
- Incremental Development
  - Easier to develop
  - Faster to market
  - Can delay unknown requirements
- Consistency
  - Coding style, Methodology, Interfaces (library APIs and user)
  - Makes things easier to understand and use

Programming – the big picture

- Start with the problem or task – often ambiguous
- Flesh out task, i.e., define requirements
- Define solution architecture
  - How the program is structured
  - Can be multiple levels, especially for complex applications
- Define algorithms and components
  - Algorithms define how work gets done
  - Components do work
- Write code for components (and test!)
- Integrate components (and test, test, test!)

Look for reuse!
- Architectures
- Design patterns
- Algorithms
- Libraries

High level application architectures

- Monolithic
  - Application contained on a single machine

- Distributed
  - Application components on network connected computers

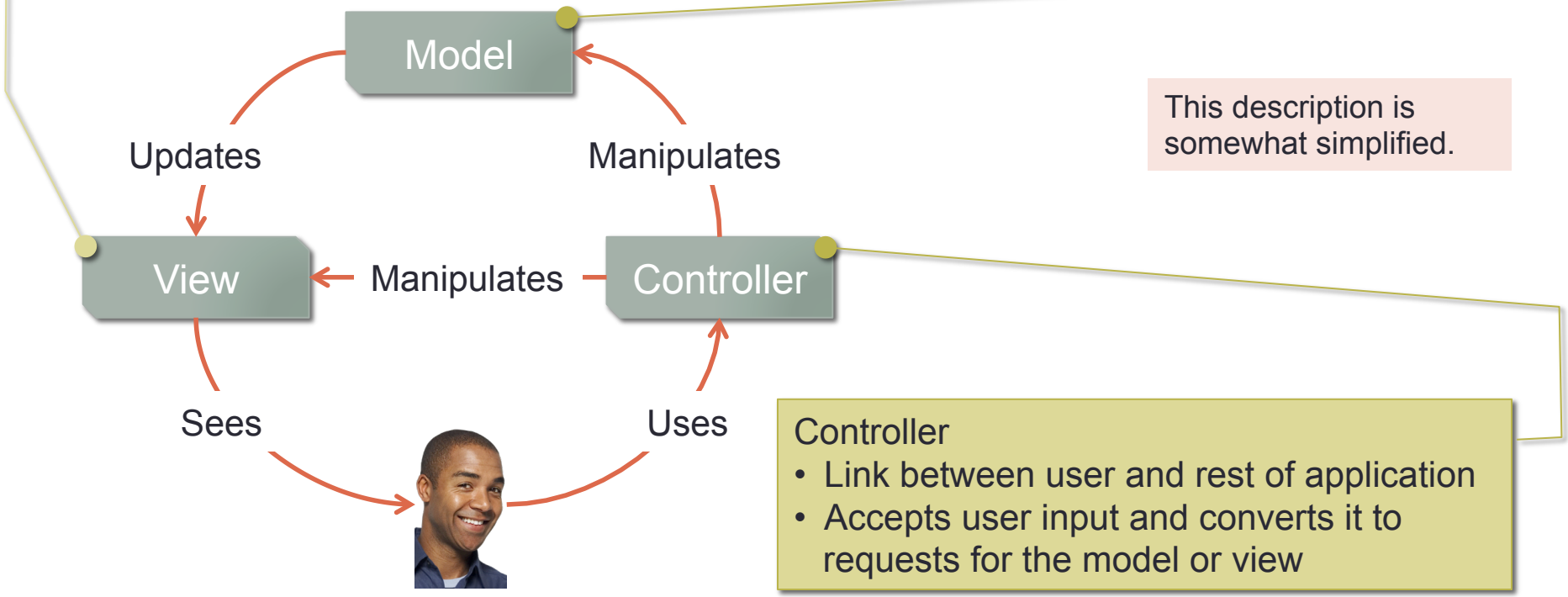
Client – Server

Multi – Tier

Mid-level application architecture -
Model – View – Controller

View
- Representation of model data
- Multiple views of the same data possible
- Responds to requests to modify view

Model
- Application data and logic
- Responds to requests about its state
- Responds to requests to change state

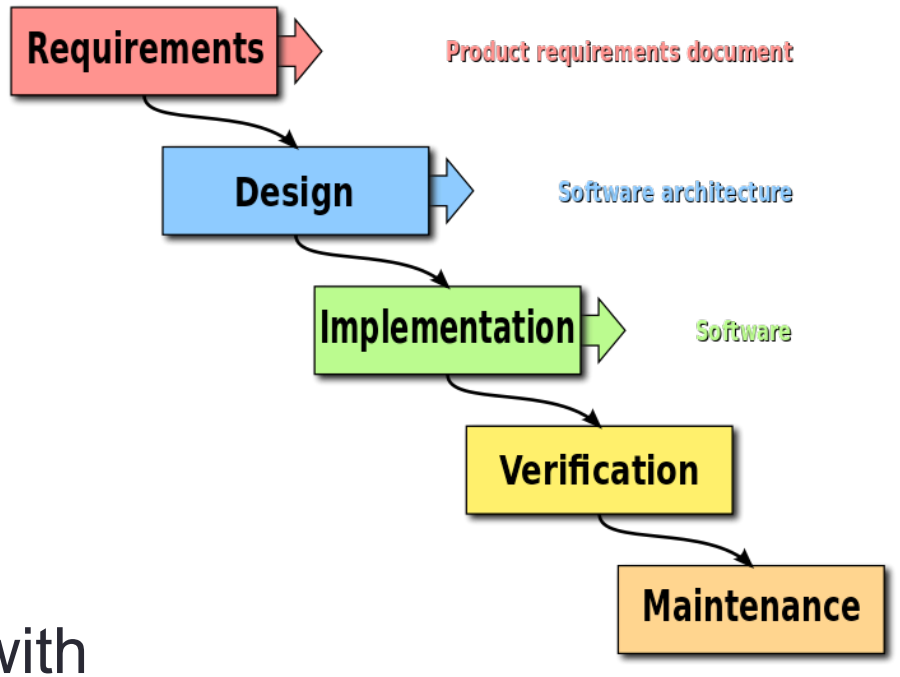


This description is somewhat simplified.

Controller
- Link between user and rest of application
- Accepts user input and converts it to requests for the model or view

Software development process models

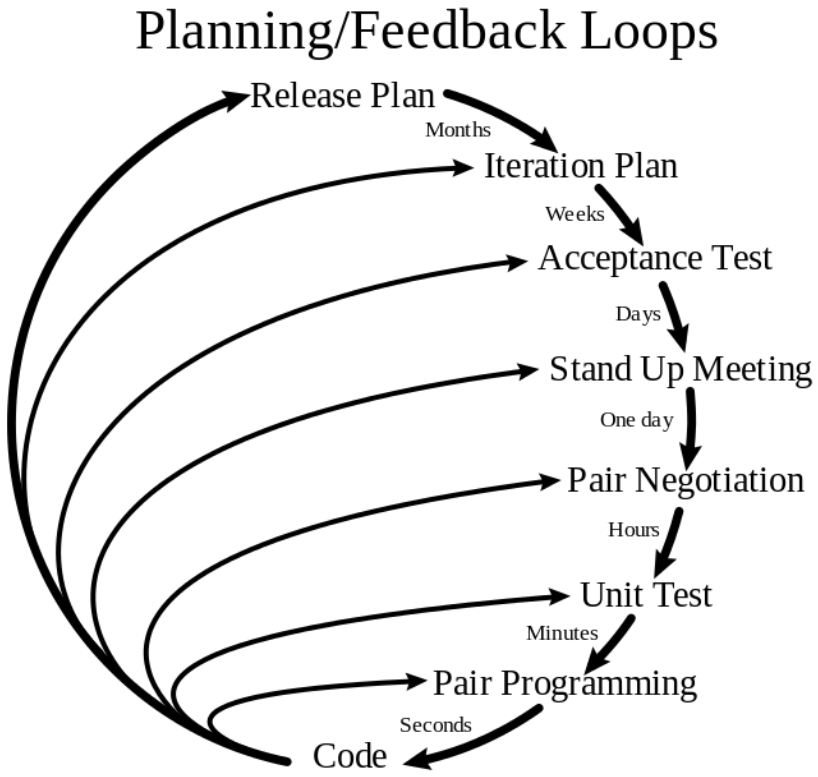
- Waterfall
  - Requirements analysis
  - Software design
  - Unit testing
  - Component testing
  - System testing
  - Maintenance



- Very early model
- Not necessarily compliant with software engineering principles

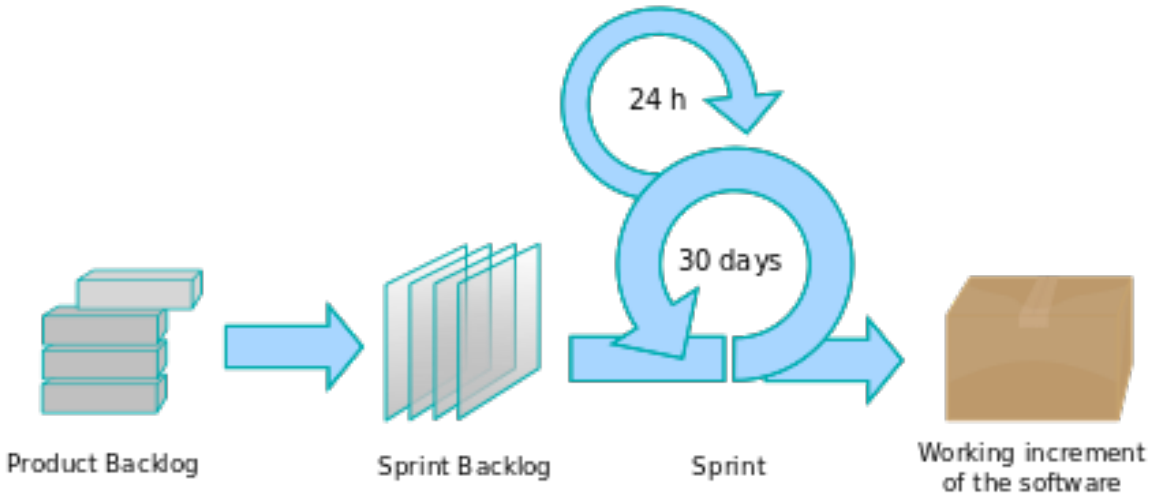
Software development process models

- Agile - Extreme programming
  - frequent "releases" in short development cycles
  - programming in pairs
  - unit testing of all code
  - Delay features until needed
  - flat management structure
  - simplicity and clarity in code



Software development process models

- Agile - Scrum
  - iterative and incremental
  - Physical co-location or close online collaboration of team
  - daily face to face communication
  - deliver quickly and respond quickly to emerging requirements.



Software development process models

- DevOps
  - Aims to help an organization rapidly produce quality software
  - Stresses communication, collaboration and integration between **dev**elopment and **op**eration**s**
  - A response to the interdependence of development and operations
- Test driven development
  - Aims to improve quality and schedule
  - Write test cases first (or simultaneously)
  - Write program
  - Test till program passes test case (typically automated)
  - Add function to test case and repeat

Core Concepts of all Programming Languages

- There are many programming languages available
- All of these languages share core concepts.
- By focusing on these concepts, you are better able to learn any programming language.
- Hence, by learning Java
  - You are poised to learn other languages.
  - You are also much more marketable as you are able to learn new languages quicker.

Programming languages

- All software written in some programming language

- Three basic forms of programming languages
  - Machine language
    - Commands defined by CPU instruction set
    - Manufacturer specific
    - CPU model specific within manufacturer
  - Assembly language
    - Commands human readable but aligned with CPU instruction set
    - Manufacturer specific
    - CPU model specific within manufacturer
  - High level language
    - Manufacturer and CPU agnostic … for programmer

Programming languages – machine

- Commands written in binary or hexadecimal generally, e.g.,
  - 1000 1101 0010 1001 or 0x8D29
- Each bit or bit field in an instruction may have specific meaning
  - Register ID
  - Operation
  - Address
- Very tedious, precise
- Very hard, time consuming
- Optimal performance (with sufficient knowledge/experience)
- Support
  - Few (or no) tools

Programming languages – assembly

- Commands defined by human readable names for
  - Operations: correspond to possible CPU actions
  - Operands: correspond to CPU registers or memory locations
- For example
  - load r1, 0x0FFCABD0
  - add r1, r2
  - branch label7
- Tedious, precise
- Hard, time consuming
- Very good performance
- Support
  - **Assembler** translates assembly language to machine language
  - Editors

Programming languages – high level

- Commands close to "natural" language
  - Meaningful words where necessary (for, do, …)
  - Well-known mathematical notations (+ - * /)
- Abstracts away need for
  - Knowing computer system specifics, e.G., CPU instruction set, I/O devices
  - Implementing difficult programming aspects, e.G. Memory management
- Supports
  - Specific data types, arrays, complex data structures
  - Important programming constructs (loops, decisions, …)
- Example:
  - Name = "fred smith"
  - Grosspay = 0
  - Grosspay = basepay + overtimepay
- Much easier to use
- High productivity, maintainability
- Support
  - **Interpreter** – executes high level language programs without compilation.
  - **Compiler** – translates into machine language
  - **IDE** – continuous edit, [compile,] run cycles

FORTRAN
ALGOL
COBOL
SNOBOL

ADA
Basic
C
APL

C++
C#
Lisp
Eiffel

Java
Smalltalk
Pascal
Perl

Python
Prolog
JavaScript
NodeJS

Visual Basic
Ruby
Groovy
PHP

Basics of Java Programming

- Java is
  - Object oriented, consistent with industry trends
  - A popular programming language
  - Widely used in industry.
- We will learn some specifics of how to program in Java
- We will cover the fundamentals:
  - Data types
  - Variables
  - Arithmetic operations
  - Conditional operations
  - Methods
  - Control (If / Else, For Loops, While Loops)
  - Arrays

Why Java?

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture-neutral
- Portable
- Performance
- Multithreaded
- Dynamic

History of Java

1991: James Gosling led a team from Sun Microsystems to create a new ***processor independent*** language

1993: Sun recognized potential to bring ***dynamic content*** to web pages

1995: Netscape agreed to integrate Java into its browser, and Java radically increases in popularity

2000: Java in mobile phones, servers, and everything in between

2006: Java open-sourced

2010: Oracle bought Sun (and thus Java)

Today: Java used in browsers, in web servers, in consumer devices (including Android devices)

- For more detail see http://oracle.com.edgesuite.net/timeline/java/

Some Java users

- LinkedIn
- Amazon
- Facebook
- Ebay
- Google
- YouTube
- Eclipse