Introduction to Software Design I
computers and programming are tools for empowering people through the art of problem solving
Empowering People
Programming in Scientific Work
computers are becoming increasingly dominant in our jobs and lives
The Power of Computers

Three distinct advantages

1. can remember/process a large amount of information
2. can process information quickly
3. can repeat a well-defined task forever

Take advantage of this through programming
programming is a tool necessary to make computers work for us
programming is a tool necessary to make computers work for us

goal: work smarter, not harder
Giving Instructions
Giving Instructions

Turn on 21st.
Giving Instructions
Giving Instructions

Turn on 21st.
Giving Instructions
Programming as a Language
Programming as a Language

Two main differences:
Programming as a Language

Two main differences:

1. A programming language speaks on a computer’s terms
Programming as a Language

Two main differences:

1. A programming language speaks on a computer’s terms
2. A computer is not smart and adaptable like a human
Two Main Aims of this Course

```java
int i = 0;

while (i < arr.length) {
    if (arr[i] % 2 == 0) {
        arr[i] = i * 2;
    } else {
        arr[i] = i / 2;
    }
    ++i;
}
```

Fundamental Programming Concepts

Computational Thinking & Problem Solving
Two Main Aims of this Course

```java
int i = 0;
while (i < arr.length) {
    if (arr[i] % 2 == 0) {
        arr[i] = i * 2;
    } else {
        arr[i] = i / 2;
    }
    ++i;
}
```

Fundamental Programming Concepts

Computational Thinking & Problem Solving
Fundamental Programming Concepts

Learning the fundamentals of a programming language

in our case, Java

Enables us to communicate instructions to the computer

Serves as a tool for problem solving
Two Main Aims of this Course

1. **Fundamental Programming Concepts**

2. **Computational Thinking & Problem Solving**

```plaintext
int while
{}
```
Computational Thinking & Problem Solving

How we approach problems and form our solutions to them

Computers are not very smart

- limited set of instructions to communicate with
- not able to adapt/interpret what we meant if we do not clearly communicate

Two steps:

1. recognizing how we might solve a problem as humans
2. translating that problem solving approach such that a computer will understand it
How can I solve a given problem with the tools I have?
Computer Science Ecosystem
Computer Science Ecosystem

- computer architecture
- computer hardware

1000101011101
010111010010
Computer Science Ecosystem

- Operating System
  - Windows
  - macOS
  - Linux

- Computer Architecture
  - Binary Code: 10001010110101011010010

- Computer Hardware
Computer Science Ecosystem
Computer Science Ecosystem
Computer Science Ecosystem

- Operating System
  - Windows
  - macOS
  - Linux

- Computer Architecture
- Computer Hardware
Computer Hardware

*hardware*: the physical components of the computer
Computer Hardware

hardware: the physical components of the computer
Computer Hardware

*hardware*: the physical components of the computer

central processing unit (CPU)
processes input
“brains” of the computer
Computer Hardware

*hardware*: the physical components of the computer

- **CPU**: central processing unit (CPU) processes input “brains” of the computer
- **memory**
Computer Hardware

*hardware*: the physical components of the computer

- **CPU**: central processing unit (CPU) processes input “brains” of the computer
- **memory**: saves/stores data, instructions e.g., hard drive, RAM, flash drive
Computer Hardware

*hardware*: the physical components of the computer

memory

CPU
Computer Hardware

*hardware*: the physical components of the computer

- Input devices
- Memory
- CPU
**Computer Hardware**

*hardware*: the physical components of the computer

- **input devices**
- **memory**
- **CPU**
- **output devices**
Computer Science Ecosystem

- Operating System
  - Windows
  - macOS
  - Linux

- Computer Architecture
  - 100010101101
  - 010111010010

- Computer Hardware
Operating System

Allows people to easily interact with a computer
Manages computer resources (e.g., hardware, software)

*applications*: programs written for a computer for a specific task
  - document creation, music/video playback, web interfaces
Computer Science Ecosystem

- Operating System
- Computer Architecture
- Computer Hardware

1000101011010101101
010111010010101101010010
Computer Science Ecosystem

operating system

computer architecture

computer hardware

100010101101
010111010010

10001010101101
01011101010010
Computer Science Ecosystem

- Operating system
- Computer architecture
- Computer hardware
Computer Science Ecosystem

operating system

computer architecture

computer hardware

100010101101
010111010010

010111010010
Computer Science Ecosystem

operating system

computer architecture

allows the hardware to communicate with the architecture in a basic programming language (low level programming language)

computer hardware
Computer Science Ecosystem

operating system

allows the OS to communicate with the architecture in a more English-like programming language (*high level programming language*)

computer architecture

allows the hardware to communicate with the architecture in a basic programming language (*low level programming language*)

computer hardware
Computer Science Ecosystem

- Operating System
  - Windows
  - macOS
  - Linux

- Computer Architecture
  - 100010101101
  - 010111010010

- Computer Hardware
Computer Science Ecosystem

**operating system**

- Windows
- Apple
- Linux

Applications run on the OS
People need to program these too!

**computer architecture**

```
100010101101
010111010010
```

**computer hardware**

- Hard drive
- RAM
- Processor
Computer Science Ecosystem

operating system

Applications run on the OS
People need to program these too!

computer architecture

100010101101
010111010010

computer hardware

10111101010010
010111010010

Applications run on the OS
People need to program these too!
Programming

A tool to give instructions to a computer

Consists of a set of fundamental *programming constructs*

These constructs are then implemented in a programming language

  - each language implements these constructs in slightly different ways
  - but, the concepts are universal across languages

Knowing one language makes it easier to learn others
Programming: Printing a Message

Displaying “Hello world!” to the user:

Java: `System.out.print("Hello world!");`
Python: `print "Hello world!"
Lisp: `(write-line "Hello world!")`
C: `printf("Hello world!");`
Programming: Printing a Message

Displaying “Hello world!” to the user:

Java: `System.out.print(“Hello world!”);`

Python: `print “Hello world!”`

Lisp: `(write-line “Hello world!”);`

C: `printf(“Hello world!”);`

basic syntax: `<printing instruction> <what to print>`
Programming Languages

Hundreds of languages
  each has unique features, uses
Can be classified into a variety of paradigms

*programming language paradigm*: a fundamental style of programming
  multiple paradigms
  each language implements one or more paradigms

In this class, we’re learning a language called Java
The Java Programming Language
The Java Programming Language

Created in 1995 by Sun Microsystems; now developed by Oracle
The Java Programming Language

Created in 1995 by Sun Microsystems; now developed by Oracle

current version is Java 8
The Java Programming Language

Created in 1995 by Sun Microsystems; now developed by Oracle

current version is Java 8

Free for everyone to use
The Java Programming Language

Created in 1995 by Sun Microsystems; now developed by Oracle
current version is Java 8
Free for everyone to use
Primarily an object-oriented programming (OOP) language
The Java Programming Language

Created in 1995 by Sun Microsystems; now developed by Oracle

Current version is Java 8

Free for everyone to use

Primarily an object-oriented programming (OOP) language

Language is OS-agnostic
The Java Programming Language

Created in 1995 by Sun Microsystems; now developed by Oracle

- current version is Java 8
- Free for everyone to use
- Primarily an object-oriented programming (OOP) language
- Language is OS-agnostic
  - identical code can be run on any machine with the same results
Object-Oriented Programming

OOP: based on the premise that programming is typically about representing real-world concepts in a way that a computer can understand/use them

overview today; will discuss this in more detail later in the semester

other OOP languages: Python, C#, Ruby, C++, Objective-C

Describes how we organize our instructions
UWL as Data

- Professors
- Students
- Classes
UWL as Data

professors

students

classes
UWL as Data

- first name
- last name
- department
UWL as Data

- first name
- last name
- department

professors

- first name
- last name
- major
- birthday

students

classes
UWL as Data

- first name
- last name
- department

- first name
- last name
- major
- birthday

- department (e.g., CS)
- number (e.g., 120)
- section (e.g., 1)
UWL as Data

- first name
- last name
- department
- list of classes teaching this semester

professors

- first name
- last name
- major
- birthday

students

- department (e.g., CS)
- number (e.g., 120)
- section (e.g., 1)

classes
UWL as Data

- first name
- last name
- department
- list of **classes**
- teaching this semester

- first name
- last name
- major
- birthday
- list of **classes**
- taking this semester

- department (e.g., CS)
- number (e.g., 120)
- section (e.g., 1)

---

Lorem Ipsum
UWL as Data

- first name
- last name
- department
- list of **classes**
  teaching this semester

- first name
- last name
- major
- birthday
- list of **classes**
  taking this semester

- department (e.g., CS)
- number (e.g., 120)
- section (e.g., 1)
- **professor** of record
- list of **students** enrolled
UWL as Data

- professors
  - first name
  - last name
  - department
  - list of classes teaching this semester

- students
  - first name
  - last name
  - major
  - birthday
  - list of classes taking this semester

- classes
  - department (e.g., CS)
  - number (e.g., 120)
  - section (e.g., 1)
  - professor of record
  - list of students enrolled
UWL as Data

- birthday
UWL as Data

- students
- birthday
UWL as Data

- birthday: Feb. 23, 1994
  - year: 1994
  - month: 2
  - day: 23
Calculating a student's age: Write out instructions to calculate a student’s age, given their birthday (i.e., year, month, day) and a value for today’s date. Avoid using words like “before” or “after”; instead, use words for numerical comparison (e.g., “greater than”, “less than or equal to”). Test your instructions with the following possibilities for today’s date:

- March 26, 2016
- January 26, 2016
- February 22, 2016
- February 24, 2016
- February 23, 2016