# Classes of Languages

***Programming Languages***

*William Killian*

Millersville University

## Lecture Outline

- Ways to Compare Languages
  - Declarative vs. Imperative
  - Structured vs. Non-structured
  - Compiled vs. Interpreted
- Modern Classes of Languages
  - Procedural
  - Functional
  - Object-Oriented
  - Data-Driven and Query
  - Multi-Paradigm*

## Declarative vs. Imperative

# Imperative Languages

- uses statements that change a program's state

```
const container = document.getElementById('container');
const btn = document.createElement('button');
btn.className = 'btn red';
btn.onclick = function(event) {
  if (this.classList.contains('red')) {
    this.classList.remove('red');
    this.classList.add('blue');
  } else {
    this.classList.remove('blue');
    this.classList.add('red');
  }
};
container.appendChild(btn);
```

# Declarative Languages

- express the logic of a computation without describing its control flow

```
class Button extends React.Component {
  this.state = { color: 'red' }
  handleChange = () => {
    const color = this.state.color === 'red' ? 'blue' : 'red’;
    this.setState({ color });
  }

  render() {
    return (<div>
      <button
        className=`btn ${this.state.color}`
        onClick={this.handleChange}>
      </button>
    </div>);
  }
}
```

## Structured vs. Unstructured

## Related Terms

- **Structured in three ways**
    1. Selection statements (if/else or switch)
    2. Sequence statements (successive statements)
    3. Iteration statements (loops – for/while/do-while)
- A language doesn’t need to have all three

## Structured Languages

Selection Statements:

```
if <cond> <then>
if <cond> <then> <else>
```

Sequence Statements:

```
<stmt1>
<stmt2>
…
```

Iteration Statements:

```
while <cond> <body>
```

# Non-Structured Languages

- What would we have without if/while/for?

```
string_len:
        cbnz  x0, .L8
        mov   w0, 0
        ret
.L8:
        stp   x29, x30, [sp, -16]!
        mov   x29, sp
        add   x0, x0, 1
        bl  string_len
        add   w0, w0, 1
        ldp   x29, x30, [sp], 16
        ret
```

labels

jumps/branches

## Questions

- Why would you want to program in a non-structured language?
- What languages are structured?
- What languages are unstructured?

## Compiled vs. Interpreted

## Compiled Languages

- Source language is translated AHEAD OF TIME to the target architecture language
- Done **once**
- Necessary for performance-critical applications

Examples?

# Interpreted Languages

- Source language is translated ON DEMAND to the target architecture language
- Can be done many times for the same code
- Necessary for (dynamic) scripting languages

Examples?

## Modern Classes of Languages

## Procedural Languages

- based on the concept of the procedure call
- Procedures contain a series of computational steps to be carried out
- Any procedure might be called during a program's execution, including by other procedures or itself.
- One “global” state (which can be subdivided)

- Features:
  - Modularity
  - Scoping

# Functional Languages

- Programs are constructed via functions/procedures
- **Declarative --** doesn’t capture any state
- *Mathematical model*

- Features:
  - Functions can take functions as parameters / return
  - Functions are **pure** – have no side-effects
  - Functions are often **recursive** – no looping constructs
  - Use **strong types** to reject invalid programs early

# Object-Oriented Languages

Objects contain two main parts of information

- **State (or data)**
    - the underlying data model used to represent an object.
- **Behavior (or code)**
    - the available set of actions which can be used to update an object’s state or interact with other entities in the program

- Features
    - Object's own procedures can access and often modify the data fields of itself (via **this** or self)
    - Objects are usually *instances* of classes, which also determine their type.

# Data-Driven and Query Languages

- We often need languages to ONLY operate on data
- Data-Driven
  - Operate on data being “matched”
  - Process different matches of data accordingly
  - Command-Line Tools: awk, sed
- Query Languages
  - Operate on a **Data Model**
  - Three main classes of operations:
    - Adding, Deleting, Modifying

  -
    ```
    INSERT INTO employees (first_name, last_name,
    fname) VALUES ('Bob', 'Smith', 'bsmith1');
    ```

## Multi-Paradigm Languages

- Most languages are **multi-paradigm** languages
- C++, Java, Python, Rust, Javascript
    - Can be object oriented
    - Can be purely functional
    - Can be purely procedural
- Often, we just use what features we **need** to solve the type of problem we are facing