CSC 311 (SURVEY OF PROGRAMMING LANGUAGES)
TOPIC: PROGRAMMING PARADIGM

DISCLAIMER
The contents of this document are intended for leaning purposes at the undergraduate level. The materials are from different sources including the internet and the contributors do not in any way claim authorship or ownership of them.

Overview of the main programming paradigms

A programming paradigm is a style, or “way,” of programming.

Some Common Paradigms are:

- **Imperative**: Programming with an explicit sequence of commands that update state.
- **Declarative**: Programming by specifying the result you want, not how to get it.
- **Structured**: Programming with clean, goto-free, nested control structures.
- **Procedural**: Imperative programming with procedure calls.
- **Functional (Applicative)**: Programming with function calls that avoid any global state.
- **Function-Level (Combinator)**: Programming with no variables at all.
- **Object-Oriented**: Programming by defining objects that send messages to each other. Objects have their own internal (encapsulated) state and public interfaces. Object orientation can be:
  - **Class-based**: Objects get state and behavior based on membership in a class.
  - **Prototype-based**: Objects get behavior from a prototype object.
- **Event-Driven**: Programming with emitters and listeners of asynchronous actions.
- **Flow-Driven**: Programming processes communicating with each other over predefined channels.
- **Logic (Rule-based)**: Programming by specifying a set of facts and rules. An engine infers the answers to questions.
Paradigms are not meant to be mutually exclusive; a single program can feature multiple paradigms!

**Overview of the functional paradigm**
Functional programming is in many respects a simpler and cleaner programming paradigm than the imperative one. The reason is that the paradigm originates from a purely mathematical discipline: the theory of functions. The imperative paradigm is rooted in the key technological ideas of the digital computer, which are more complicated, and less 'clean' than mathematical function theory.

Below we characterize the most important, overall properties of the functional programming paradigm.

- **Characteristics:**
  - Discipline and idea
  - Mathematics and the theory of functions
  - The values produced are *non-mutable*
    - Impossible to change any constituent of a composite value
    - As a remedy, it is possible to make a revised copy of composite value
  - Atemporal
    - Time only plays a minor role compared to the imperative paradigm
  - Applicative
    - All computations are done by applying (calling) functions
  - The natural abstraction is the function
    - Abstracts a single expression to a function which can be evaluated as an expression
  - Functions are first class values
    - Functions are full-fledged data just like numbers, lists, ...
  - Fits well with computations driven by needs
    - Opens a new world of possibilities

**Overview of the logic paradigm**

The logic paradigm is dramatically different from the other three main programming paradigms. The logic paradigm fits extremely well when applied in problem domains that deal with the extraction of knowledge from basic facts and relations. The logical paradigm seems less natural in the more general areas of computation.

Below we briefly characterize the main properties of the logic programming paradigm.
Overview of the object-oriented paradigm

The object-oriented paradigm has gained great popularity in the recent decade. The primary and most direct reason is undoubtedly the strong support of encapsulation and the logical grouping of program aspects. These properties are very important when programs become larger and larger.

The underlying, and somewhat deeper reason to the success of the object-oriented paradigm is probably the conceptual anchoring of the paradigm. An object-oriented program is constructed with the outset in concepts, which are important in the problem domain of interest. In that way, all the necessary technicalities of programming come in second row.

As for the other main programming paradigms, we will now describe the most important properties of object-oriented programming, seen as a school of thought in the area of computer programming.

Characteristics:
- Discipline and idea
  - The theory of concepts, and models of human interaction with real world phenomena
  - Data as well as operations are encapsulated in objects
  - Information hiding is used to protect internal properties of an object
  - Objects interact by means of message passing
    - A metaphor for applying an operation on an object
  - In most object-oriented languages objects are grouped in classes
    - Objects in classes are similar enough to allow programming of the classes, as opposed to programming of the individual objects
    - Classes represent concepts whereas objects represent phenomena
  - Classes are organized in inheritance hierarchies
    - Provides for class extension or specialization

This ends the overview of the three main programming paradigms.
Languages and Paradigms

One of the characteristics of a language is its support for particular programming paradigms. For example, Smalltalk has direct support for programming in the object-oriented way, so it might be called an object-oriented language. OCaml, Lisp, Scheme, and JavaScript programs tend to make heavy use of passing functions around so they are called “functional languages” despite having variables and many imperative constructs.

There are two very important observations here:

- Very few languages implement a paradigm 100%. When they do, they are pure. It is incredibly rare to have a “pure OOP” language or a “pure functional” language. A lot of languages have a few escapes; for example in OCaml, you will program with functions 90% or more of the time, but if you need state, you can get it.
- A lot of languages will facilitate programming in one or more paradigms. In Scala you can do imperative, object-oriented, and functional programming quite easily. If a language is purposely designed to allow programming in many paradigms is called a multi-paradigm language.

Sources

- [http://cs.lmu.edu/~ray/notes/paradigms/](http://cs.lmu.edu/~ray/notes/paradigms/)