The History of the Common Lisp Object System (CLOS) started in the late 1970's when some of the ideas from Smalltalk reached Massachusetts Institute of Technology (MIT) and a team of scientists under direction of Howard Cannon developed an object-oriented programming system with multiple inheritance, named Flavors. In the mean time, under the influence of Smalltalk and Knowledge Representation Language (KRL), scientists at the Xerox company worked on a similar project which led to the development of Lisp Object Oriented Programming System (LOOPS) later changed to Common LOOPS. Both systems were extensions of various dialects of the Lisp language, which originally was created by John McCarthy in late 1950's.

In 1986 when the first efforts to standardize the language were made the working group X3J13 was formed. In order to standardize the Lisp a uniform object oriented system was designed. As a starting point, X3J13 used two previously mentioned existing systems, Flavors and Common LOOPS, and in 1990 created Common Lisp Object System (CLOS). When, in 1993 the ANSI (American National Standard Institute) and, in 1994 the ISO (International Standards Organization) standards were developed, CLOS as a part of Common Lisp, became the first object oriented standardized language. The major authors were: Daniel G. Bobrow, Linda G. DeMichiel, Richard P. Gabriel, Sonya E. Keene, Gregor Kiczales, and David A. Moon.

CLOS is a large system used for various purposes including development of complex software systems, modeling and simulating business problems and engineering. CLOS due to its complexity, flexibility and richness of the features is very popular in the Artificial Intelligence community. CLOS is far more advanced than most of the modern and more popular OO languages. It provides support for multimethods, dynamic type creation, multiple inheritance and what is probably one of the most important features class redefinition in running programs. As a result, there is no need to for developers to take the system down, alter the program, recompile it and restart. Another aspect that attributes to CLOS's advantage is its speed. Unlike Java, which is designed to be compiled for the virtual machine, CLOS is designed to compile for a specific type of computer. It is particularly important in case of large programs as CLOS applications very often are.

A part of CLOS is the Meta Object Protocol (MOP), which was designed specifically for CLOS. The main idea behind MOP is that it contains a set of rules determining how CLOS object system works, which are implemented automatically. However, these rules can be modified enabling a programmer to customize the object system to better serve an application. MOP enables modification of any object system even from another language. Other interesting aspects of CLOS include its ability to be run using utilities from other languages, and its ability to create stand-alone applications, which means the need for these parts of library, which are needed only by the application. One more very important element of CLOS is its ability to manipulate programs written in other languages.

Comparing Java or C++ with CLOS someone may point first of all at one very important aspect: If CLOS is so good, why are Java and C++ so popular when CLOS is not? First of all we have to keep in mind that CLOS due to its complexity is difficult to learn and master. It is often used for large experimental or research problems, which very often don't deliver what is expected, as it is in AI. As a result CLOS does not have a good reputation because part of the blame falls on the language. In addition, industry often needs programs, which serve specific relatively simple purposes. Therefore there is no need for such a complex object oriented system like CLOS. For example, Java originally was designed to operate the microprocessors of household appliances. In addition CLOS was created long before the Internet became popular, the main source of JAVA's success. On other hand, JAVA appeared just in time and was simple enough to gain programmers' acceptance. As we see, different languages serve different purposes and the ability to task large problems does not guarantee the language's popularity. In other words, for many problems using CLOS would be over kill.

SAMPLE OF CODE.

1)
```
 (write-line "Hello, world.") ;Print the famous greeting.
```

2)
```
;;; A function to produce the factorial of an integer.
(defun factorial (n)
        (if (= n 1)
        1
        (* (factorial (- n 1)) n)))
```
3)
```
;y=m * x + b
(SETQ y(+(*mx)b))
```

4)creates instance of Submarine "Sub-1" and assigns new values to "Depth" and "Speed"

```
(defclass Submarine ()
  ((Depth :accessor Depth :initform 100)
   (Speed :accessor Speed :initform 5)))

(setq Sub-1 (make-instance 'Submarine))
(Depth Sub-1)
(Speed Sub-1)
(setf (Depth Sub-1) 200)
(setf (Speed Sub-1) 4)
(Depth Sub-1)
(Speed Sub-1)
```

Common Lisp Object System - CLOS
Created 1990
Authors:
Daniel G. Bobrow, Linda G. DeMichiel, Richard P. Gabriel, Sonya E. Keene, Gregor Kiczales, and David A. Moon.

1) Abstraction	10
2) Clarity of Code	4
3) Encapsulation	8
4) Inheritance	10
5) Libraries	7
6) Polymorphism	10
7) Portability	8
8) Reuse	9
9) Verbose	7
10) Size	10
11) Syntax	5
12) Powerful	10
13) Binding	Dynamic
14) Exception Handling	8

Bibliography:
Andreas Paepcke "Object-Oriented Programming: The CLOS Perspective", MIT Press, 1993
http://elwoodcorp.com/alu/table/lisp.htm
http://www.franz.com/
http://cgbin.erols.com/ziring/cgi-bin/cep/cep.pl?_key=Common+Lisp
http://www.apl.jhu.edu/~hall/Lisp-Notes/Defclass.html