Jatha FAQ

How do I run Jatha?

```
java -classpath jatha.jar org.jatha.Jatha

   or, if you don't want to use the GUI:

java -classpath jatha.jar org.jatha.Jatha -nodisplay
```

How do I use Jatha from within another Java program?

```
import org.jatha.Jatha;

 ...

Jatha myLisp  = new Jatha(false, false);
myLisp.init();
myLisp.start();
 ...
// Create a new symbol
LispValue  symbol1  = myLisp.parse("FOO", LispParser.PRESERVE); // preserve case
 ...
// Evaluate a form:
try {
  String input  = "(* 5 10)";
  LispValue result = myLisp.eval(input);
  System.out.println(input + " = " + result);
} catch (Exception e) {
 System.err.println("LISP Exception: " + e);
}
```

How do I use LISP expressions with variables from Java?

Any of the following will work in v2.7.1 and higher. All return the value **35**:

```
1) System.out.println(lisp.eval("(let ((x 7)) (* 5 x)))"));  // uses a local variable

2) System.out.println(lisp.eval("(progn (setq x 7) (* 5 x))"));  // executes multiple statements

3) System.out.println(lisp.eval("(setq x 7)"));  // creates a global variable
   System.out.println(lisp.eval("(* 5 x)"));     // uses a global variable
```

The `eval` method accepts an optional second parameter that is a list of global variables. However, the global variable list is difficult to construct. The examples here are easier to use.

How do I use the dynatype package?

The `org.jatha.dynatype` package provides a rich set of dynamically-typed objects that parallel the Common LISP type system.

The root class is an interface called `LispValue` that is implemented by StandardLispValue. All values in the system are instances of `LispValue`. Usually, you should declare variables of type `LispValue` rather than more specific types.

```
import org.jatha.Jatha;
import org.jatha.dynatype.*;

Jatha lisp = new Jatha(false, false);
lisp.init();
lisp.start();

LispValue foo1 = lisp.makeInteger(7);
LispValue foo2 = lisp.makeSymbol("Hello");
LispValue foo3 = lisp.makeReal(3.14159);
LispValue foo4 = lisp.makeList(foo1, foo2, foo3);

LispValue foo5 = foo4.car();
LispValue foo6 = foo4.second();

System.out.println(foo1);   // and so on.
```

Sometimes you will want to access the Java version of a value rather than the LISP version. Usually this happens in boolean expressions. To access these values, use the `basic_` methods. Instead of:

```
  if (foo1.numberp() == lisp.T)

    or

  if (foo4.length().greaterThan(lisp.makeInteger(5)) == lisp.T)
```

```
It is easier to say:
```

```
  if (foo1.basic_numberp())

    and

  if (foo1.basic_length() > 5)
```

```
All Sequence types have iterators, so you can iterate over a LispValue that
is a list by doing standard Java iteration:


  Iterator listIterator = foo4.iterator();
  while (listIterator.hasNext())
  {
    LispValue value = (LispValue)listIterator.next();

    ...
  }


Of course you can also write a while loop that steps through
the list using car() and cdr() if
you like.
```