```
    /** Return the value of sym */
    public Object lookup(String sym) throws UnboundSymbolException
    {
        Object o = cache.get(sym);
        if(o != null)
            return o;
        if(ancestor == null)
            throw new UnboundSymbolException(sym);
        return ancestor.lookup(sym);
    }

    /** Bind the symbol to the value */
    public void bind(String symbol, Object value)
    {
        cache.put(symbol, value);
    }
}
```

The hash table `cache` maps symbols (`String`s) to values (`Object`s). Each environment contains a pointer to the previous environment, called the *ancestor*. The `lookup` method looks up the symbol in the cache. If it is not found, it attempts to look it up in the ancestor. If it's not found in any environment, then an exception is thrown, indicating that the symbol is not bound.

The evaluator compiles each form into bytecodes that have the same semantics as the form. Each form evaluates to a value. The bytecodes leave the value of that form on the stack.

Numbers compile into instances of the class `Integer`. To compile the form

```
5
```

the compiler generates this code:

```
new java/lang/Integer                          ; Create an integer
dup
iconst_5                                       ; Push the value
invokespecial java/lang/Integer/<init>(I)V     ; Construct the
                                               ; object
```

To compile a symbol, the program looks up the symbol in the current binding environment using the `lookup` method. The compiler keeps the current binding environment in local variable 1. To compile the form

```
x
```