Discussion Section 6 (10/02/2008)

1. Regular-expression constructs (incomplete):

Characters

\\	The backslash character
\t	The tab character ('\u0009')
\n	The newline (line feed) character
\r	The carriage-return character

Character classes

[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)

Predefined character classes

.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\w	A word character: [a-zA-Z_0-9]

Boundary matchers

^	The beginning of a line
$	The end of a line

Greedy quantifiers

X?	X, once or not at all
X*	X, zero or more times
X+	X, one or more times
X{n}	X, exactly n times
X{n,}	X, at least n times
X{n,m}	X, at least n but not more than m times

Reluctant quantifiers

X??	X, once or not at all
X*?	X, zero or more times
X+?	X, one or more times

Logical operators

XY	X followed by Y
X|Y	Either X or Y
(X)	X, as a capturing group

Back references

\n	Whatever the nth capturing group matched

Special constructs (non-capturing)

(?:X)	X, as a non-capturing group

Example: match "`fall/2005`" with

```
"([a-m/]).*"            "([a-m/]*).*"            ".*([a-m/]*).*"
".*([a-m/]+).*"         ".*?([a-m/]*).*"
```

2. Scanner:

```
public final class Scanner {
    public Scanner useDelimiter(Pattern pattern) { ... }
    public boolean hasNext(Pattern pattern) { ... }
    public String next(Pattern pattern) { ... }
    public Scanner skip(Pattern pattern) { ... }
}
```

3. Command:

```
abstract class Command {
    Command (List<Command> operands) { ... }
    abstract Value execute (Interpreter machine);
}
```

4. Handling variables:

```
package java.util;
public interface Map<K,V> {
    public V get(Object key) { ... }
    public V put(K key, V value) { ... }
}
```