EECS1012
MOBILE COMPUTING

STRINGS

(SLIDES ADAPTED FROM PROF.H. ROUMANI)

PROF. Y. LESPÉRANCE
Dept. of Electrical Engineering & Computer Science

1

ABOUT STRINGS

- They are objects like the rest
  But with syntactic sugar to make instantiation easier and with an operator to make concatenation seem natural.
- A String is a sequence of chars
  This sequence is the state held by the object.
- They have an extremely rich API
  Available through a number of classes such as String, StringBuffer, Matcher, Pattern, …
- They Are Immutable
  For performance purposes.

2

SYNTACTIC SUGAR AND MUTABILITY

Declaration

The statement:

```
String name = new String("York");
```

is the same as:

```
String name = "York";
```

Concatenation

To join two strings, use the coercive + operator:

```
String fullName = name + " University";
```

in place of the messy:

```
String fullName = ((new StringBuffer(name)).append(" University")).toString();
```

Mutability

The statement:

```
name.toUpperCase();
```

does *not* change name. To do so, re-assign name.

3

THE STRING API

Surgery
length
charAt
substring

Transforms
toLowerCase
toUpperCase
trim

Patterns
indexOf
replaceAll
split / Tokenizer

Numeric Strings
+ coercion
parseInt, …
char minus '0'

Comparators
Equals *[IgnoreCase]*
compareTo
Empty vs null

Regex
Pattern compile
Matcher matcher
find and group

4

Exercises

In a class with a String attribute implement these features:

- A constructor that takes a string parameter
  Must set the attribute accordingly.
- public String get() and public void set(String s)
  Accessor and mutator.
- public int repeatCount(char c) and (String c)
  Returns the number of times c occurs in the state.
- public String toDayName(int d)
  Returns the state with any 0-6 digit replaced with Sun…Sat.
- public String trimLeadingBlanks()
  Returns the state with any leading spaces removed.

5

Regex

CHARACTER SPECIFICATIONS	
`[a-m]`	*Range. A characters between a and m, inclusive*
`[a-m[A-M]]`	*Union. a through m or A through M*
`[abc]`	*Set. The character a, b, or c*
`[^abc]`	*Negation. Any character except a, b, or c*
`[a-m&&[^ck]]`	*Intersection. a though m but neither c nor k*
CONSTRUCTS	
`^ and $`	*Designate the Beginning and End of a string*
`X | Y`	*Either X or Y*
`(X)`	*Treat X as a capturing group*
`(?:X)`	*Treat X as a non-capturing group*
`\`	*Escape*
`.`	*Any character*
`\d`	*A digit, [0-9]*
`\s`	*A whitespace character, [\t\n\x0B\f\r]*
`\w`	*A word character, [a-zA-Z_0-9]*
`\p{Punct}`	*A punctuation, [!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~]*
QUANTIFIERS	
`x?`	*x, once or not at all*
`x*`	*x, zero or more times*
`x+`	*x, one or more times*
`x{n,m}`	*x, at least n but no more than m times*

6

Search vs Pattern Search

Find the substring m4n7r2 in this string:
Mine Canadian postal codes in this string:

```
4#Q5 G6/Q50b6/ y7H7$U1)q5-t50v0l1$C1)v2(f3(D20G9.U9%R5%c9+a80d2#k1#u3,
M9,x2$N8#o3$O1(f8,m5(f3h1#u9+T1)T1D2*K8#d3&u3'b0A6R4D6%B70Z0+g00J8't
5&c8u0P2w4( x2#u5H9$J5&X3#U1'w0'y1%B3+y3+L8+k7.Y0%f9'd8z3X3%F4%z2,o2l
0*h1#Y0$S0(o4.x4s8&p4R2u2%M20c2%s1(z9$i90I3,t9o0'o8+a5,a8F3'h0(w0( w0)o
0(f1$Q4'D80M7e7%r4J0)N5)i6,l0#x80C6*v40j9'R7'G9$J1$B2#S7&r5&f4*e2p3*D6'
v7.k6N7V0.u5)L2%t1%x30E3'p0+J7%u9&A4$I1&k9Y2*a5v9)E1&G8+S1(D1(u1'm8(e
8a5h7'j6g7%R9&G0$E10h50S7#m9$o3#k0#c5's8I9#n5*U9A2(y6w1#E0)H1%j0&a1
%l4,r6&X30j0$r5&z9*w4*L8.i4.M9+F3.B0c70r4)n6'd0\e7&M3O1( w7j1s5+x3)c0\3(
a70 A5/T0.k7Q5$q2%F4r1(c1$A8*x8)x2Z3o9)k1,E60u0G20T2,U1e5)u5(T0,Q5c9.j6/
T8.d2a8+T3$C9h8&p00d5B1.N7'W8n6$H6,l1a1#c7J5$P7*b3'N7+Y0)q0.F1e3*Y7B5.
X7+j2+n0'g4+e1*c9(E50Q7,m8b00b4/ P60b7-z1)U4,V4#O6$ I2/ m20 b8#p7,F1+u1*
U8t8o3%t2v2U0w6O1,S8b1$f0+y6h8)j5$N1$o6*D70h0+s5#z2%e4n0C0,A8S4(z5*d
9#W2K7+b2*I0$d90D4#u8'I10L8'w8&i3+Y9,t7)v1'r7%d1)o6#i6#C1,M3e9,K50Q4*V
4%o5)y5&p0&n4*x8' W2+b0i1(l0#O1-r4$o8%B8)x0$P3#u2,P3i4d4#c3'm4n7r2.r3/
x1%R50T6r3)f90p5a4%N4'B0(m0X90o5Z2&V8)v4*b6L0&S1$M9*j7#s1$K6*x4&q4
%x3$s3&l8%b2$o60j5#Y5m9(K2(R4m3)Q4%E0q1(W0.G5L4)m8(e1'Y3c4w3Y2'T9%k
```

7

Code Exemplars

Given the string s:

```
String query = "m4n7r2";
System.out.println(s.indexOf(query));
// Can also use indexOf(int, String) in a loop to capture all occurrences
```

Search

```
String regex = "[A-Za-z]\\d[A-Za-z][ ]?\\d[A-Za-z]\\d";
Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(s);
while (matcher.find())
{
    // use matcher.group(); .start(); .end();
}
```

Mine

8

Exercises

Given a String s, write a fragment that:

- Mine all telephone numbers in it.
- Determines if s has a person's height (expressed in feet and inches, as in 5'9") imbedded in it.
- Determines the largest word in s. We define a word as a sequence of non-space characters. *Use split.*
- Validates that s represents a query string. These are used in web applications. They begin with ? and consist of one or more var=value clauses delimited by &.
- *Validates that s is a DNA strand (consists of A,C,G,T, begins with ATG and its length is divisible by 3). If so, output the most frequent amino acid in it (any sequence of 3 letters).*

9