Regular expressions (regex): Finding patterns in text

Prof. Dionne Aleman

MIE250: Fundamentals of object-oriented programming
University of Toronto

What are "regular expressions"?

- regex for short
- A way to describe a set of strings based on common characteristics
- Used to search, edit, and manipulate text and data
- Not Java-specific, but Java has very nice built-in tools to implement regex pattern matching

What does regex do?

Takes in a pattern and searches for that pattern in a string

Who cares?

- Data validation
  - Email addresses
  - Phone numbers
  - Credit card numbers
- Text replacement
- Syntax and grammar highlighting
  - Eclipse
  - Dreamweaver
  - Word
- Search engines

Real-world research example

The Princess Margaret Cancer Centre is one of the top five cancer research and treatment institutes in the world. Retrospective studies are common:

- Determine the performance of certain treatments
- Determine survival rates of certain cancers
  - Dependence on treatment modality
  - Dependence on cancer tissue location

The problem

- Find all patients treated in the last five years with the following criteria:
  - Lung cancer
  - Radiation prescription dose: 70Gy

- Being able to find ALL patients is important!
  - Otherwise, study results may be biased

Basic ideas to find matches

- Identify lung cancer patients
  - Common organs outlined in CT scans
    - Left lung, right lung, heart, liver, spine, skin, tumor (a.k.a. GTV and PTV)[1]
- Identify correct treatment dose
  - Look for "70Gy" in the database information
- Sounds easy enough

[1] GTV: Gross tumor volume
PTV: Planning tumor volume

The reality

- Real-world data is messy
- Different doctors contour different organs and name them without convention:
  - `RLung, Right Lung, r_lung, lungR`
  - `PTV, Target70-54, GTVmargin, PTV-minus-GTV`
  - `Lung-Not-PTV, LungPTV70`
- Radiation prescription dosage must be determined from structure names

The answer: regex

- Go through old patients and rename structures to match current conventions
- For example:
  - If a name contains "`lung`" and "`r`" $\rightarrow$ right lung
  - If a name contains "`PTV`" and a number $\rightarrow$ the number is prescription dose
  - If a name contains "`not`", disregard whatever follows
- Still messy, but we can catch most patients so far

Pattern: foo

f	o	o	1	2	3
0	1	2	3	4	5

String to search	Pattern found at index
foo123	

Pattern: foo

f	o	o	1	2	3
0	1	2	3	4	5

String to search	Pattern found at index
foo123	0-3

Note that the ranges include the starting index but not the ending index: 0-3 means [0,3)

Pattern: foo

f	o	o	f	o	o	f	o	o
0	1	2	3	4	5	6	7	8

String to search	Pattern found at index
foo123	0-3
foofoofoo	

Note that the ranges include the starting index but not the ending index: 0-3 means [0,3)

Pattern: foo

f	o	o	f	o	o	f	o	o
0	1	2	3	4	5	6	7	8

String to search	Pattern found at index
foo123	0-3
foofoofoo	0-3

Note that the ranges include the starting index but not the ending index: 0-3 means [0,3)

Pattern: foo

f	o	o	f	o	o	f	o	o
0	1	2	3	4	5	6	7	8

String to search	Pattern found at index
foo123	0-3
foofoofoo	0-3, 3-6

Note that the ranges include the starting index but not the ending index: 0-3 means [0,3)

Pattern: foo

f	o	o	f	o	o	f	o	o
0	1	2	3	4	5	6	7	8

String to search	Pattern found at index
foo123	0-3
foofoofoo	0-3, 3-6, 6-9

Note that the ranges include the starting index but not the ending index: 0-3 means [0,3)

Pattern: `cat.`

c	a	t	s	?	!
0	1	2	3	4	5

String to search	Pattern found at index
`cats?!`	

Pattern: `cat.`

c	a	t	s	?	!
0	1	2	3	4	5

String to search	Pattern found at index
`cats?!`	0-4

Wait, what? There's no "cat." in "cats?!"

Metacharacters

Metacharacters

Metacharacters are wildcards! And there are a lot of them ...

Metacharacters

Character	Description
`[]`	Match anything inside the square brackets for ONE character position
-	Range separator (e.g., [0-9])
^	"Not" (negation)
.	Any character in this position
?	Preceding character is optional
*	Preceding character can be repeated 0 or more times
+	Preceding character can be repeated 1 or more times
`{n}`	Preceding character (or range) must appear `n` times exactly
`{n,m}`	Preceding character (or range) must appear at least `n` times but not more than `m` times
()	Grouping
| (pipe)	Find left OR right side values

But if you literally meant '.' (a period)?

Use \ to escape metacharacters, e.g., '\.', '\?'.

Character classes[2]

Construct	Translation
`[abc]`	a, b, or c (simple class)
`[^abc]`	Any character except a, b, or c (negation)
`[a-zA-Z]`	a through z, or A through Z, inclusive (range)
`[a-d[m-p]]`	a through d, or m through p: `[a-dm-p]` (union)
`[a-z&&[def]]`	d, e, or f (intersection)
`[a-z&&[^bc]]`	a through z, except for b and c: `[ad-z]` (subtraction)
`[a-z&&[^m-p]]`	a through z, and not m through p: `[a-lq-z]` (subtraction)

[2] Adapted from Oracle's Java Tutorials

Pattern: `in[du]`

Search string	Match
Windows	

Pattern: `in[du]`

Search string	Match
Windows	✓

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓
Mac OS X	

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓
Mac OS X	X

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓
Mac OS X	X
Dolphins	

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓
Mac OS X	X
Dolphins	X

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓
Mac OS X	X
Dolphins	X
Find them and destroy them!	

Pattern: `in[du]`

Search string	Match
Windows	✓
Linux	✓
Mac OS X	X
Dolphins	X
Find them and destroy them!	✓

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X
I don't like xRays	

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X
I don't like xRays	✓

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X
I don't like xRays	✓
I don't like x-rays	

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X
I don't like xRays	✓
I don't like x-rays	X

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X
I don't like xRays	✓
I don't like x-rays	X
X11 or x11?	

Pattern: `x[0-9A-Z]`

Search string	Match
He's excitable	X
He's EXCITABLE	X
I don't like xRays	✓
I don't like x-rays	X
X11 or x11?	✓

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X
Linux2.2.16-22	

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X
Linux2.2.16-22	✓

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X
Linux2.2.16-22	✓
IE6	

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X
Linux2.2.16-22	✓
IE6	X

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X
Linux2.2.16-22	✓
IE6	X
Mozilla/4.75-2.1	

Pattern: `((4\.[0-3])|(2\.[1-4]))`

Translation: “4.x” or “2.y”, where x is a number in [0-3] and y is a number in [1-4]

Search string	Match
Mozilla/4.0	✓
Opera 4.5	X
Linux2.2.16-22	✓
IE6	X
Mozilla/4.75-2.1	✓

Pattern: `ba{2,4}b`

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X
abba	

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X
abba	X

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X
abba	X
baab	

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X
abba	X
baab	✓

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X
abba	X
baab	✓
hubaaaab	

Pattern: ba{2,4}b

Translation: “bab”, where ‘a’ is repeated 2-4 times

Search string	Match
bab	X
abba	X
baab	✓
hubaaaab	✓

Pattern: `ba{2,4}b`

Translation: "bab", where 'a' is repeated 2-4 times

Search string	Match
bab	X
abba	X
baab	✓
hubaaaab	✓
baaaaab	

Pattern: `ba{2,4}b`

Translation: "bab", where 'a' is repeated 2-4 times

Search string	Match
bab	X
abba	X
baab	✓
hubaaaab	✓
baaaaab	X

More examples

- Pattern: `colou?r`
  - Translation: “color” or “colour”
- Pattern: `colou*r`
  - Translation: “colour” with any number of u’s (including zero)
- Pattern: `[0-9&&[^345]]`
  - Translation: A number 0-9, except 3, 4, and 5

One more time

Character	Description
`[]`	Match anything inside the square brackets for ONE character position
-	Range separator (e.g., [0-9])
^	"Not" (negation)
.	Any character in this position
?	Preceding character is optional
*	Preceding character can be repeated 0 or more times
+	Preceding character can be repeated 1 or more times
`{n}`	Preceding character (or range) must appear `n` times exactly
`{n,m}`	Preceding character (or range) must appear at least `n` times but not more than `m` times
()	Grouping
| (pipe)	Find left OR right side values

Shorthand in Java

Shorthand	Longhand	Description
.	.	Any character
\d	[0-9]	A digit
\D	[^0-9]	A non-digit
\s	[\t\n\x0B\f\r]	A whitespace character
\S	[^\s]	A non-whitespace character
\w	[a-zA-Z_0-9]	A word character
\W	[^\w]	A non-word character

Shorthand examples

- Pattern: `[\d]{3}-[\d]{4}`
  - Translation: "xxx-xxxx", where x is a digit

- Pattern: `[\w]+@[\w]+(\.[a-z]{2,4})`
  - Translation: Simple email validation
  - `[\w]+`: Any number of word characters (at least one)
  - `@`: Followed by the `@` sign
  - `[\w]+`: Followed by at least one word character
  - `(\.[a-z]{2,4})`: Followed by '.', then 2-4 lower case letters

Say we want a number in the range 1-255[3]

Will `[0-255]` work?

[3] Adapted from `http://www.regular-expressions.info/numericranges.html`

Say we want a number in the range 1-255[3]

Will `[0-255]` work?

No! That pattern is actually just "0-2 or 5 or 5"!

[3] Adapted from `http://www.regular-expressions.info/numericranges.html`

Can we find a number in a certain range with regex?

Can we find a number in a certain range with regex?

Yes, but it won't be pretty.

First things first

- The intent behind regex is to find patterns in *text*.
- All matching is done on a per-character basis.
- So, “1” is one character and “255” is three characters.

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	`[1-9][0-9][0-9]`

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	`[1-9][0-9][0-9]`
4	1000-9999	

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	`[1-9][0-9][0-9]`
4	1000-9999	`[1-9][0-9][0-9][0-9]`

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	`[1-9][0-9][0-9]`
4	1000-9999	`[1-9][0-9][0-9][0-9]`
2-4	10-9999	

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	`[1-9][0-9][0-9]`
4	1000-9999	`[1-9][0-9][0-9][0-9]`
2-4	10-9999	`[1-9][0-9]{1,3}`

Brute force pattern creation for range 0-9999

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-999	`[1-9][0-9][0-9]`
4	1000-9999	`[1-9][0-9][0-9][0-9]`
2-4	10-9999	`[1-9][0-9]{1,3}`

Put it all together: `[0-9]|[1-9][0-9]{1,3}`

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	`1[0-9][0-9]`

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	`1[0-9][0-9]`
3	200-249	

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	`1[0-9][0-9]`
3	200-249	`2[0-4][0-9]`

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	`1[0-9][0-9]`
3	200-249	`2[0-4][0-9]`
3	250-255	

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	`1[0-9][0-9]`
3	200-249	`2[0-4][0-9]`
3	250-255	`25[0-5]`

0-999 is easy ... but what about 0-255?

We need to exclude numbers 256-999.

Number of digits	Number range	Pattern
1	0-9	`[0-9]`
2	10-99	`[1-9][0-9]`
3	100-199	`1[0-9][0-9]`
3	200-249	`2[0-4][0-9]`
3	250-255	`25[0-5]`

Put it all together:

```
[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]
```

We did it!

- ... but it was ugly, hard to put together, and hard to read.
- It was basically everything I am trying to teach you not to do.

Regex can still be very useful

1. Use regex to extract numbers in text: `-?[0-9]+`

2. And then ... ?

Regex can still be very useful

1. Use regex to extract numbers in text: `-?[0-9]+`

2. And then ... ?

```
1 if (x >= 0 && x <= 255) {
2     // do something
3 }
```

Now that we have the idea ...

How to do regex in Java?

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

Regex library

`Pattern` class

- Holds the pattern

`Matcher` class

- Does the matching

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

A bit unusual ...

src/regex.java

The `Pattern` class

- No public constructor, so must call `Pattern.compile(str)` to initialize object

- Is the `compile()` method static or not?

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

The Matcher class

src/regex.java

The Matcher class

The `matcher()` method in the `Pattern` class returns a `Matcher` object

`Matcher` method	Description
`find()`	Finds the next matching subsequence; returns boolean
`find(int n)`	Finds pattern starting from n
`matches()`	Boolean to check if the entire sequence matches the pattern
`start()`	Returns the start index of the previous match
`end()`	Returns the end index of the previous match
`group()`	Returns the previous match as a `String`

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" startin
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end(
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

While there is a matching subsequence, print out the details of the matched pattern

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.*;

public class regex {
    public static void main(String[] args) throws IOException{
        String strPattern, strInput;
        BufferedReader cin = new BufferedReader(new InputStreamReader(System.in));

        while (true) {
          System.out.print("\nEnter your pattern: ");
          strPattern = cin.readLine();
          Pattern pattern = Pattern.compile(strPattern);

          do {
            System.out.print("\nEnter your input string (0 to enter new pattern): ");
            strInput = cin.readLine();

            if (!strInput.equals("0")) {

                Matcher matcher = pattern.matcher(strInput);

                boolean found = false;
                while (matcher.find()) {
                    System.out.format("I found the text \"%s\" starting at " +
                       "index %d and ending at index %d.\n",
                        matcher.group(), matcher.start(), matcher.end());
                    found = true;
                }
                if (!found) System.out.println("No match found.");
            } // end check if user wants to quit

          } while (!strInput.equals("0"));
        } // infinite loop
    } // end main
} // end class
```

src/regex.java

Ending this program

Does this program ever end? Either way, let's play with the program.

Want more regex fun?

`http://www.regexcrossword.com/`