Pattern Matching

# Agenda

## Introduction

* **Simple Search**
* **Search Using Metacharacters**
* **Search Using Quantifiers**

**Introduction**

There is a need to search through large amounts of textual data, looking for some specific stuff. Regular expressions (regex for short) are a kind of language within a language, designed to help programmers with these searching tasks. Every language that provides regex capabilities uses one or more regex engines. Regex engines search through textual data using instructions that are coded into expressions. A regex expression is like a very short program or script. When you invoke a regex engine, you’ll pass it the chunk of textual data you want it to process ( In java this is usually a string or stream), and you pass it the expression you want it to use to search through the data.

## Simple Search

#1

source : abaaaba pattern: ab output 0 4

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegexSmall {

**public static void** main(String[] args) { Pattern p=Pattern.*compile*("ab"); Matcher m=p.matcher("abaaaba"); **while** (m.find()) {

System.***out***.println(m.start());

}

}

}

#2

source : abababa pattern: aba output 0 4

Try by yourself

# Searches using Metacharacters

We have following metacharacters in regex.

\d - digits

\s - A whitespace charater

\w - A word character(letters, digits, or "\_" underscore)

#1: Search through some source data looking for all occurrences of numeric digits. regex : \d

source : a12c3e456f pattern: "\\d"

output 1 2 4 6 7 8

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** PatternMatchingDemo {

**public static void** main(String[] args) { Pattern p=Pattern.*compile*("\\d"); Matcher m=p.matcher("a12c3e456f"); **while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}

#2: Search through some source data looking for all occurrences of (letters, digits, or \_(Underscore)) regex : \w

source : a 1 56 \_Z pattern: "\\w"

output 0 2 4 5 7 8

**import** java.util.regex.Matcher; **import** java.util.regex.Pattern; **public class** PatternMatchingDemo {

**public static void** main(String[] args) { Pattern p=Pattern.*compile*("\\w"); Matcher m=p.matcher("a 1 56 \_Z"); **while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}

#3: Search through some source data looking for all spaces. regex : \s

source : a 1 56 \_Z pattern: "\\s" output 1 3 6

**import** java.util.regex.Matcher; **import** java.util.regex.Pattern; **public class** PatternMatchingDemo {

**public static void** main(String[] args) { Pattern p=Pattern.*compile*("\\s"); Matcher m=p.matcher("a 1 56 \_Z"); **while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}

You can also specify sets of characters to search for using brackets and ranges of characters to search for using square brackets and dash:

**Searches for Specific characters or Range of characters**

exm:

[abc] - it searches only for a's, b's and c's

[a-f] - it searches only for a,b,c,d,e,f characters #

regex : [abc]

source : abcefgabc123abxaybzc pattern: [abc]

output 0126781213151719

**import** java.util.regex.Matcher; **import** java.util.regex.Pattern; **public class** PatternMatchingDemo {

**public static void** main(String[] args) { Pattern p=Pattern.*compile*("[abc]");

Matcher m=p.matcher("abcefgabc123abxaybzc");

**while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}

you can search across several ranges at once. The following expression is looking for occurrences of the letters a- f or A-F, it is not looking for an fA combination

[a-fA-F] – Searches for the first six letters of the alphabet, both cases. #

source : abcefgabc123abxaybzc pattern: [abc]

output 0126781213151719

**Note : you can also apply the following attributes to sets and ranges within square brackets: *“^”* to negate the characters specified, nested brackets to create union of sets, and *“&&”* to specify the intersection of sets. To understand this more refer *java.util.regex.Patter API***

## Search Using Quantifiers

Let’s say that we want to create a regex pattern to search for hexadecimal literals. As a first step, let’s solve the problem for one-digit hexadecimal numbers.

0[xX][0-9a-fA-F]

The previous regex can be read as : “Find a set of characters in whinch the first character is a “0”, the second character is either an “x”or an “X”, and the third character is either a digit from “0” to “9” , a letter from “a” to “f” or an upper case letter “A” to “F”

#

source : “12 0x 0x12 0Xf 0Xg” pattern: 0[xX][0-9a-fA-F] output 6 11

**import** java.util.regex.Matcher; **import** java.util.regex.Pattern; **public class** PatternMatchingDemo {

**public static void** main(String[] args) {

Pattern p=Pattern.*compile*("0[xX][0-9a-fA-F]"); Matcher m=p.matcher("12 0x 0x12 0Xf 0Xg"); **while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}

what if we just wanted regex to find occurrences of integers? Integers can be one or more digits long, so it ourld be great if we could say “one or more” in an expression. There is a set of regex constructs called ***quantifiers*** that let us specify concepts such as ***“one or more”***. In fact, the quantifier that represents ***“one or more”*** is the ***“+”*** character***.*** There are some other quantifiers.

#

Source : “1 a12 234b” Patter: \\d+

Output: 0 3 6

**import** java.util.regex.Matcher; **import** java.util.regex.Pattern; **public class** PatternMatchingDemo {

**public static void** main(String[] args) { Pattern p=Pattern.*compile*("\\d+"); Matcher m=p.matcher("1 a12 234b"); **while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}

Two other quantifiers we’re going to look at are (\*) Zero or more occurrence

(?) Zero or one occurrence

# Suppose we have a text file that contains a comma separated list of all the file names in a directory that contains several very important projects information. You want to extract all the files whose names start with proj1. You might discover .txt , .java, .pdf file.? What kind of regex expression could we create to find these various proj1 files?

Suppose we have following list of files in one txt file. “proj3.txt,proj1abc.pdf, proj1.txt,proj2.txt,proj1.java”

To solve this problem we will be using the regex expression [^] carat sign.

**import** java.util.regex.Matcher; **import** java.util.regex.Pattern; **public class** PatternMatchingDemo {

**public static void** main(String[] args) {

Pattern p=Pattern.*compile*("proj1([^,])\*");

Matcher m=p.matcher("proj3.txt,proj1abc.pdf,proj1.txt,proj2.txt,proj1.java");

**while**(m.find()){

System.***out***.print(m.start()+" ");

}

}

}