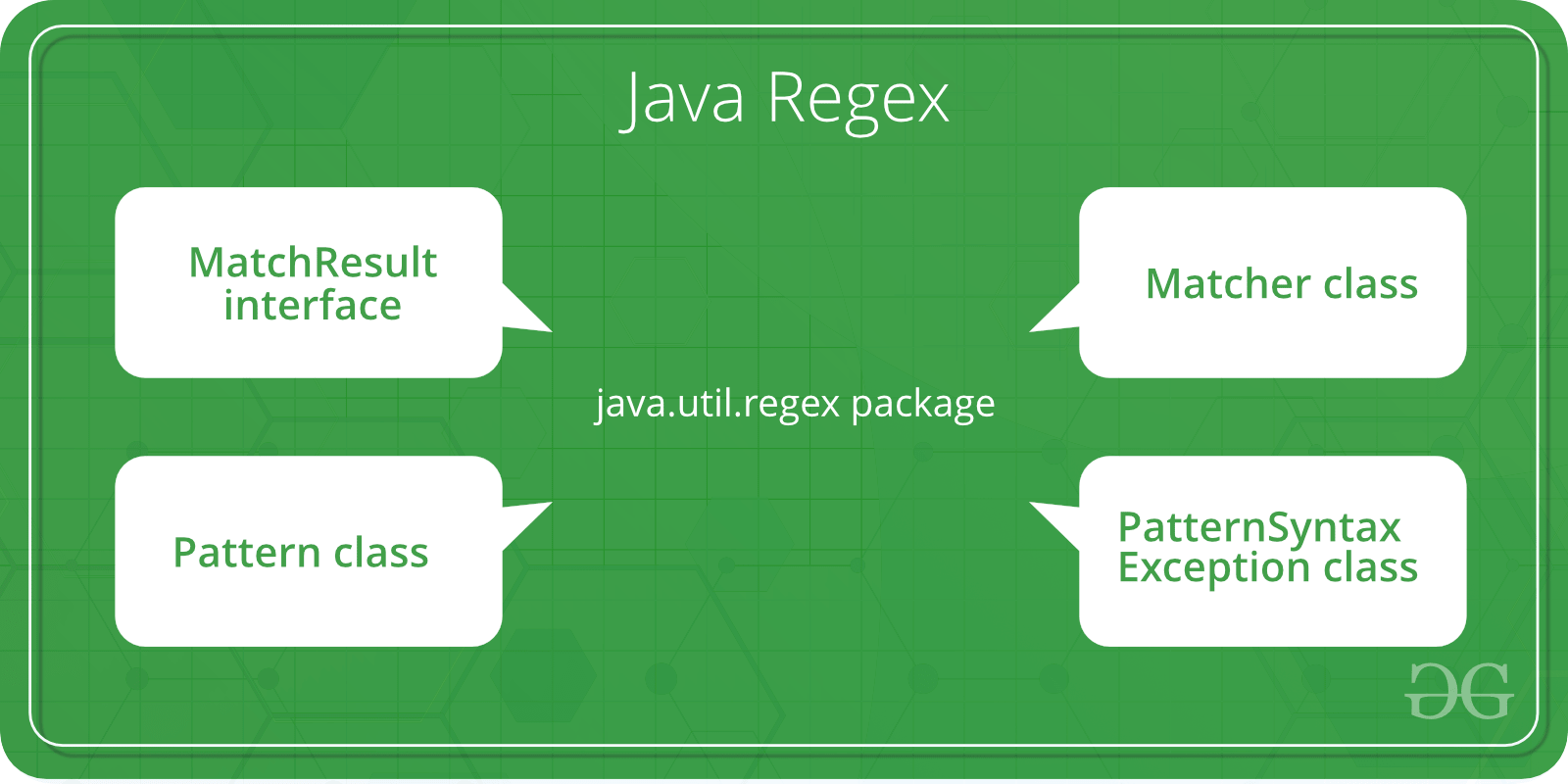
Regular Expressions in Java

<https://www.geeksforgeeks.org/regular-expressions-in-java/?ref=lbp>

Regular Expressions or Regex (in short) is an API for defining String patterns that can be used for searching, manipulating and editing a string in Java. Email validation and passwords are few areas of strings where Regex are widely used to define the constraints. Regular Expressions are provided under java.util.regex package. This consists of 3 classes and 1 interface.



**The java.util.regex** package primarily consists of the following three classes:

| **CLASS** | **DESCRIPTION** |
| --- | --- |
| util.regex.Pattern | Used for defining patterns |
| util.regex.Matcher | Used for performing match operations on text using patterns |
| PatternSyntaxException | Used for indicating syntax error in a regular expression pattern |

**MatchResult interface:** This interface is used to determine the result of a match operation for a regular expression. It must be noted that although the match boundaries, groups and group boundaries can be seen, the modification is not allowed through a MatchResult.

1. **java.util.regex.Pattern class:**This class is a compilation of regular expressions that can be used to define various types of patters, providing no public constructors. This can be created by invoking the compile() method which accepts a regular expression as the first argument, thus returns a pattern after execution.

// A Simple Java program to demonstrate working of

// Pattern.matches() in Java

import java.util.regex.Pattern;

class Demo

{

    public static void main(String args[])

    {

        // Following line prints "true" because the whole

        // text "geeksforgeeks" matches pattern "geeksforge\*ks"

        System.out.println (Pattern.matches("geeksforge\*ks",

                                            "geeksforgeeks"));

        // Following line prints "false" because the whole

        // text "geeksfor" doesn't match pattern "g\*geeks\*"

        System.out.println (Pattern.matches("g\*geeks\*",

                                            "geeksfor"));

    }

}

1. **java.util.regex.Matcher class:**This object is used to perform match operations for an input string in java, thus interpreting the previously explained patterns. This too defines no public constructors. This can be implemented by invoking a matcher() on any pattern object.

Note that Pattern.matches() checks if the whole text matches with a pattern or not. Other methods (demonstrated below) are mainly used to find multiple occurrences of pattern in text.

**Java Programs to demonstrate workings of compile(), find(), start(), end() and split() :**

**Java Program to demonstrate simple pattern searching**

// A Simple Java program to demonstrate working of

// String matching in Java

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class Demo

{

    public static void main(String args[])

    {

        // Create a pattern to be searched

        Pattern pattern = Pattern.compile("geeks");

// Create a pattern to be searched

        Pattern pattern = Pattern.compile("ge\*");

        // Search above pattern in "geeksforgeeks.org"

        Matcher m = pattern.matcher("geeksforgeeks.org");

        // Print starting and ending indexes of the pattern

        // in text

        while (m.find())

            System.out.println("Pattern found from " + m.start() +

                               " to " + (m.end()-1));

    }

}

Output:

Pattern found from 0 to 4

Pattern found from 8 to 12

1. **PatternSyntaxException class:**This object of Regex is used to indicate a syntax error in a regular expression pattern and is an unchecked exception.

**Important Observations/Facts:**

1. We create a pattern object by calling Pattern.compile(), there is no constructor. compile() is a static method in Pattern class.
2. Like above, we create a Matcher object using matcher() on objects of Pattern class.
3. Pattern.matches() is also a static method that is used to check if given text as a whole matches pattern or not.
4. find() is used to find multiple occurrences of pattern in text.
5. We can split a text based on a delimiter pattern  
   using split()

# Check if URL is valid or not in Java

<https://www.geeksforgeeks.org/check-if-url-is-valid-or-not-in-java/>

Given a URL as string, we need to find if the given URL is valid or not.

Input : str = "https://www.geeksforgeeks.org/"

Output : Yes

Input : str = "https:// www.geeksforgeeks.org/"

Output : No

Note that there is a space after https://

**Using java.net.url**  
We can use [java.net.url](https://www.geeksforgeeks.org/url-class-java-examples/) class to validate a URL. The idea is to create a URL object from the specified String representation. If we do not get exception while creating the object, we return true. Else we return false.

// Java program to check if a URL is valid

// using java.net.url

import java.net.URL;

class Test {

    /\* Returns true if url is valid \*/

    public static boolean isValid(String url)

    {

        /\* Try creating a valid URL \*/

        try {

            new URL(url).toURI();

            return true;

        }

        // If there was an Exception

        // while creating URL object

        catch (Exception e) {

            return false;

        }

    }

    /\*driver function\*/

    public static void main(String[] args)

    {

        String url1 = "<http://www.geeksforgeeks.org/>";

        if (isValid(url1))

            System.out.println("Yes");

        else

            System.out.println("No");

        String url2 = "http:// www.geeksforgeeks.org/";

        if (isValid(url2))

            System.out.println("Yes");

        else

            System.out.println("No");

    }

}

Output:

Yes

No

# Check if email address valid or not in Java

<https://www.geeksforgeeks.org/check-email-address-valid-not-java/?ref=rp>

Given a string, find if the given string is a valid email or not.

Input : email = "contribute@geeksforgeeks.org"

Output : Yes

Input : email = "contribute@geeksforgeeks..org"

Output : No

There is an extra dot before org.

Prerequisite : [Regular Expressions in Java](https://www.geeksforgeeks.org/regular-expressions-in-java/)  
We use below regular expression provided in [OWASP Validation Regex repository](https://www.owasp.org/index.php/OWASP_Validation_Regex_Repository).

**^[a-zA-Z0-9\_+&\*-] + (?:\\.[a-zA-Z0-9\_+&\*-]**

**+ )\*@(?:[a-zA-Z0-9-]+\\.) + [a-zA-Z]{2, 7}**

// Java program to check if an email address

// is valid using Regex.

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class Test

{

    public static boolean isValid(String email)

    {

        String emailRegex = "^[a-zA-Z0-9\_+&\*-]+(?:\\."+

                            "[a-zA-Z0-9\_+&\*-]+)\*@" +

                            "(?:[a-zA-Z0-9-]+\\.)+[a-z" +

                            "A-Z]{2,7}$";

        Pattern pat = Pattern.compile(emailRegex);

        if (email == null)

            return false;

        return pat.matcher(email).matches();

    }

    /\* driver function to check \*/

    public static void main(String[] args)

    {

        String email = "contribute@geeksforgeeks.org";

        if (isValid(email))

            System.out.print("Yes");

        else

            System.out.print("No");

    }

}

# Program to check valid mobile number

<https://www.geeksforgeeks.org/java-program-check-valid-mobile-number/>

**Mobile Number validation criteria:**

* The first digit should contain number between 7 to 9.
* The rest 9 digit can contain any number between 0 to 9.
* The mobile number can have 11 digits also by including 0 at the starting.
* The mobile number can be of 12 digits also by including 91 at the starting

The number which satisfies the above criteria, is a valid mobile Number.

Examples:

Input : Enter Mobile Number:

7873923408

Output :Valid Mobile Number

Input : Enter Mobile Number:

5678729234

Output :Invalid Mobile Number

Prerequisites : [Java Regular Expressions](https://www.geeksforgeeks.org/regular-expressions-in-java/)

// Java program to check if given mobile number

// is valid.

import java.util.regex.\*;

import java.util.Scanner;

class MobileNumberValidation {

    public static boolean isValid(String s)

    {

        // The given argument to compile() method

        // is regular expression. With the help of

        // regular expression we can validate mobile

        // number.

        // 1) Begins with 0 or 91

        // 2) Then contains 7 or 8 or 9.

        // 3) Then contains 9 digits

        Pattern p = Pattern.compile("(0/91)?[7-9][0-9]{9}");

        // Pattern class contains matcher() method

        // to find matching between given number

        // and regular expression

        Matcher m = p.matcher(s);

        return (m.find() && m.group().equals(s));

    }

    public static void main(String[] args)

    {

        String s = "347873923408"

        if (isValid(s))

            System.out.println("Valid Number");

        else

            System.out.println("Invalid Number");

    }

}

Output:

Invalid Number

# Get the first letter of each word in a string using regex in Java

<https://www.geeksforgeeks.org/get-first-letter-word-string-using-regex-java/>