**44-642 - App Design: Patterns and Frameworks**

**Java Regex Key**

Note: Code shown is just an example, your own test code will vary.

1. Write a regular expression that validates a phone number in these formats:

|  |  |
| --- | --- |
| Phone Number Format | Regular Expression |
| (543)1234567 | \\(\\d{3}\\)\\d{3}\\d{4} |
| 5431234567 | \\d{10} |
| 543-123-4567 | (?:\d{3}-){2}\d{4} |
| (543)123-4567 | \\(\\d{3}\\)\\d{3}-\\d{4} |
| (543)1234567 or (543)123-4567 | \\(\\d{3}\\)\\d{3}-?\\d{4} |

public class JavaRegex {

public static void main(String[] args) {

String[] phoneNumbers = {

"(543)1234567", "5431234567", "543-123-4567", "(543)123-4567",

"(5431234567)", "12-3456-7890", "123-4567", "abcdefghij"};

String pattern = "\\d{10}|(?:\\d{3}-){2}\\d{4}|\\(\\d{3}\\)\\d{3}-?\\d{4}";

for(String phoneNum : phoneNumbers) {

if (phoneNum.matches(pattern)) {

System.out.println(phoneNum+ " : Valid");

} else {

System.out.println(phoneNum+ " : Invalid");

}

}

}

}

**Output:**

(543)1234567 : Valid

5431234567 : Valid

543-123-4567 : Valid

(543)123-4567 : Valid

(5431234567) : Invalid

12-3456-7890 : Invalid

123-4567 : Invalid

abcdefghij : Invalid

1. Write a regular expression to validate a SSN.

|  |  |
| --- | --- |
| SSN Format | Regular Expression |
| 123-45-6789 / 192837465 | ^(\\d{3}-?\\d{2}-?\\d{4})$ |

public class JavaRegex {

public static void main(String[] args) {

String[] ssnNumbers = {

"123-45-6789", "9876-5-4321", "987-65-4321 ", "192-83-7465"};

String pattern = "^(\\d{3}-?\\d{2}-?\\d{4})$";

for(String ssnNum : ssnNumbers) {

if (ssnNum.matches(pattern)) {

System.out.println(ssnNum+ " : Valid");

} else {

System.out.println(ssnNum+ " : Invalid");

}

}

}

}

**Output:**

123-45-6789 : Valid

9876-5-4321 : Invalid

987-65-4321 : Invalid

192-83-7465 : Valid

1. Write a regular expression to validate passwords. (Use your own criteria for the password to validate)

**Password Criteria:**

|  |  |
| --- | --- |
| Criteria | Regular Expression |
| Minimum length – 8 characters | .{8,} |
| Contains at least one digit | (?=.\*[0-9]) |
| Contains at least one upper case and one lower case | (?=.\*[a-z])(?=.\*[A-Z]) |
| Contains at least one special character | (?=.\*[@#$%^&+=]) |
| Does not contains any spaces or tabs | (?=\\S+$).{ |

public class JavaRegex {

public static void main(String[] args) {

String[] passwords = {

"PassW0rd@", "1234567", "password", "Password$","G00dP@ssWord"};

String pattern = "^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%^&+=])(?=\\S+$).{8,}$";

for(String password : passwords) {

if (password.matches(pattern)) {

System.out.println(password+ " : Valid");

} else {

System.out.println(password+ " : Invalid");

}

}

}

}

**Output:**

PassW0rd@ : Valid

1234567 : Invalid

password : Invalid

Password$ : Invalid

G00dP@ssWord : Valid

1. Write a regular expression to verify if it contains any of the following words.

* Bat
* Cat
* Rat

public class JavaRegex {

public static void main(String[] args) {

String[] words = {

"bat", "hat", "cat", "rat","mat"};

String pattern = "[bcr]at";

for(String word : words) {

if (word.matches(pattern)) {

System.out.println(word+ " : Found");

} else {

System.out.println(word+ " : Not Found");

}

}

}

}

**Output:**

bat : Found

hat : Not Found

cat : Found

rat : Found

mat : Not Found

1. Write a regular expression that validates a person’s name in the following formats.

* FirstName LastName
* LastName, FirstName
* LastName,FirstName
* LastName FirstName

public class JavaRegex {

public static void main(String[] args) {

String[] names = { "Bandi, Ajay", "Bondalapati,Harish", "John-Smith","Aziz.Fellah","Purna Medarametla"};

String pattern = "(?i)(^[a-z])((?![ ,]$)[a-z ,]){0,24}$";

for(String name : names) {

if (name.matches(pattern)) {

System.out.println(name+ " : Valid");

} else {

System.out.println(name+ " : Invalid");

}

}

}

}

**Output:**

Bandi, Ajay : Valid

Bondalapati,Harish : Valid

John-Smith : Invalid

Aziz.Fellah : Invalid

Purna Medarametla : Valid