UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI **MATEMATICA E INFORMATICA**

Interfacce Grafiche e Programmazione ad Eventi

Carmine Dodaro

Anno Accademico 2019/2020

Espressioni regolari

Cosa sono

Un'espressione regolare è una stringa di testo che può essere utilizzata per trovare dei pattern all'interno di un'altra stringa

In Java

La classe di riferimento per le espressioni regolari in Java è la classe Pattern che si trova nel package java.util.regex.Pattern.
La classe Pattern si può usare in due modi:

- usando il metodo statico matches, che restituisce true se input soddisfa l'espressione regolare, false altrimenti
- usando il metodo static compile, che permette di compilare un'espressione regolare e poi utilizzarla più volte

Metodo matches

Come si usa

```
public boolean test(String regex, String input) {
  boolean res = Pattern.matches(regex, input);
  if(res)
    System.out.println("Espressione regolare soddisfatta");
  else
    System.out.println("Espressione regolare non soddisfatta");
}
```

Metodo compile

Come si usa

```
public boolean test(String regex, String input) {
  Pattern pattern = Pattern.compile(regex);
  Matcher matcher = pattern.matcher(input);
  boolean res = matcher.matches();

  if(res)
    System.out.println("Espressione regolare soddisfatta");
  else
    System.out.println("Espressione regolare non soddisfatta");
}
```

Esempi

Supponiamo di implementare un metodo public boolean test1(String input) che restituisce true se input è la stringa "ab", false altrimenti.

Esempi

Supponiamo di implementare un metodo public boolean test1(String input) che restituisce true se input è la stringa "ab", false altrimenti.

```
public boolean test1(String input) {
  if(input.equals("ab"))
    return true;
  return false;
}
```

Esempi

Supponiamo di implementare un metodo public boolean test1(String input) che restituisce true se input è la stringa "ab", false altrimenti.

```
public boolean test1(String input) {
  if(input.equals("ab"))
    return true;
  return false;
}
```

... usando la classe Pattern?

Esempi

Supponiamo di implementare un metodo public boolean test1(String input) che restituisce true se input è la stringa "ab", false altrimenti.

```
public boolean test1(String input) {
  if(input.equals("ab"))
    return true;
  return false;
}
```

... usando la classe Pattern?

```
public boolean test1(String input) {
  return Pattern.matches("ab", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una b (es. aaaaaab).

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una b (es. aaaaaab).

```
public boolean test2(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') continue;
    if(input.charAt(i)=='b' && input.length()-1 ==i && i > 0)
      return true;
    return false;
  }
  return false;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una b (es. aaaaaab).

```
public boolean test2(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') continue;
    if(input.charAt(i)=='b' && input.length()-1 ==i && i > 0)
      return true;
    return false;
  }
  return false;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una b (es. aaaaaab).

```
public boolean test2(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') continue;
    if(input.charAt(i)=='b' && input.length()-1 ==i && i > 0)
      return true;
    return false;
  }
  return false;
}
```

... usando la classe Pattern?

```
public boolean test2(String input) {
  return Pattern.matches("a+b", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (anche vuota) di a seguita da una b (es. aaaaaab).

Esempi

Restituiamo true se la stringa in input contiene una sequenza (anche vuota) di a seguita da una b (es. aaaaaab).

```
public boolean test2v(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') continue;
    if(input.charAt(i)=='b' && input.length()-1 ==i)
      return true;
    return false;
  }
  return false;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (anche vuota) di a seguita da una b (es. aaaaaab).

```
public boolean test2v(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') continue;
    if(input.charAt(i)=='b' && input.length()-1 ==i)
      return true;
    return false;
  }
  return false;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (anche vuota) di a seguita da una b (es. aaaaaab).

```
public boolean test2v(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') continue;
    if(input.charAt(i)=='b' && input.length()-1 ==i)
      return true;
    return false;
  }
  return false;
}
```

... usando la classe Pattern?

```
public boolean test2v(String input) {
  return Pattern.matches("a*b", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b (es. aaabbbbb).

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b (es. aaabbbbb).

```
public boolean test3(String input) {
  boolean switch_ = false;
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if(input.charAt(i)=='b' && !switch_ && i!=0) switch_ = true;
    if(input.charAt(i)=='b' && switch_) continue;
    return false;
  }
  return switch_;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b (es. aaabbbbb).

```
public boolean test3(String input) {
  boolean switch_ = false;
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if(input.charAt(i)=='b' && !switch_ && i!=0) switch_ = true;
    if(input.charAt(i)=='b' && switch_) continue;
    return false;
  }
  return switch_;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b (es. aaabbbbb).

```
public boolean test3(String input) {
  boolean switch_ = false;
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if(input.charAt(i)=='b' && !switch_ && i!=0) switch_ = true;
    if(input.charAt(i)=='b' && switch_) continue;
    return false;
  }
  return switch_;
}
```

... usando la classe Pattern?

```
public boolean test3(String input) {
  return Pattern.matches("a+b+", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di ab (es. abababababab).

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di ab (es. abababababab).

```
public boolean test4(String input) {
  if(input.length()%2!=0)
    return false;
  for(int i = 0; i < input.length()-1; i+=2) {
    if(input.charAt(i) == 'a' && input.charAt(i+1)=='b')
      continue;
    return false;
  }
  return true;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di ab (es. abababababab).

```
public boolean test4(String input) {
  if(input.length()%2!=0)
    return false;
  for(int i = 0; i < input.length()-1; i+=2) {
    if(input.charAt(i) == 'a' && input.charAt(i+1)=='b')
      continue;
    return false;
  }
  return true;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di ab (es. abababababab).

```
public boolean test4(String input) {
  if(input.length()%2!=0)
    return false;
  for(int i = 0; i < input.length()-1; i+=2) {
    if(input.charAt(i) == 'a' && input.charAt(i+1)=='b')
      continue;
    return false;
  }
  return true;
}
```

... usando la classe Pattern?

```
public boolean test4(String input) {
  return Pattern.matches("(ab)+", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b e c (es. aaabbbcccbcbcbcb)

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b e c (es. aaabbbcccbcbcbcb)

```
public boolean test5(String input) {
  boolean switch_ = false;
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && !switch_
        && i!=0) switch_ = true;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && switch_)
        continue;
    return false;
  }
  return switch_;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b e c (es. aaabbbcccbcbcbcb)

```
public boolean test5(String input) {
  boolean switch_ = false;
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && !switch_
        && i!=0) switch_ = true;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && switch_)
        continue;
    return false;
  }
  return switch_;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b e c (es. aaabbbcccbcbcbcb)

```
public boolean test5(String input) {
  boolean switch_ = false;
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && !switch_
        && i!=0) switch_ = true;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && switch_)
        continue;
    return false;
  }
  return switch_;
}
```

... usando la classe Pattern?

```
public boolean test5(String input) {
  return Pattern.matches("a+[bc]+", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b oppure di c (es. aaabbbbbb oppure aaaaccccc)

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b oppure di c (es. aaabbbbbb oppure aaaaccccc)

```
public boolean test6(String input) {
  boolean switch_ = false;
  char carattere = '';
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && !switch_
        && i!=0) {
      switch_=true;
      carattere=input.charAt(i);
    }
    if((input.charAt(i)==carattere) && switch_) continue;
    return false;
  }
  return switch_;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b oppure di c (es. aaabbbbbb oppure aaaaccccc)

```
public boolean test6(String input) {
  boolean switch_ = false;
  char carattere = '';
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && !switch_
        && i!=0) {
      switch_=true;
      carattere=input.charAt(i);
    }
    if((input.charAt(i)==carattere) && switch_) continue;
    return false;
  }
  return switch_;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di a seguita da una sequenza (non vuota) di b oppure di c (es. aaabbbbbb oppure aaaaccccc)

```
public boolean test6(String input) {
  boolean switch_ = false;
  char carattere = '';
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a' && !switch_) continue;
    if((input.charAt(i)=='b'||input.charAt(i)=='c') && !switch_
        && i!=0) {
      switch_=true;
      carattere=input.charAt(i);
    }
    if((input.charAt(i)==carattere) && switch_) continue;
    return false;
  }
  return switch_;
}
```

... usando la classe Pattern?

```
public boolean test6(String input) {
  return Pattern.matches("a+(b+|c+)", input);
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di qualunque carattere tranne a (es. bbbbcsd)

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di qualunque carattere tranne a (es. bbbbcsd)

```
public boolean test7(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') return false;
  }
  return input.length()>0;
}
```

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di qualunque carattere tranne a (es. bbbbcsd)

```
public boolean test7(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') return false;
  }
  return input.length()>0;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input contiene una sequenza (non vuota) di qualunque carattere tranne a (es. bbbbcsd)

```
public boolean test7(String input) {
  for(int i = 0; i < input.length(); i++) {
    if(input.charAt(i)=='a') return false;
  }
  return input.length()>0;
}
```

... usando la classe Pattern?

```
public boolean test7(String input) {
  return Pattern.matches("[^a]+", input);
}
```

Esempi

Restituiamo true se la stringa in input è 5 a

Esempi

Restituiamo true se la stringa in input è 5 a

```
public boolean test8(String input) {
  if(input.length() != 5)
    return false;
  for(int i = 0; i < 5; i++)
    if(input.charAt(i) != 'a')
      return false;
  return true;
}
```

Esempi

Restituiamo true se la stringa in input è 5 a

```
public boolean test8(String input) {
  if(input.length() != 5)
    return false;
  for(int i = 0; i < 5; i++)
    if(input.charAt(i) != 'a')
      return false;
  return true;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input è 5 a

```
public boolean test8(String input) {
  if(input.length() != 5)
    return false;
  for(int i = 0; i < 5; i++)
    if(input.charAt(i) != 'a')
      return false;
  return true;
}
```

... usando la classe Pattern?

```
public boolean test8(String input) {
  return Pattern.matches("a{5}", input);
}
```

oppure

```
public boolean test8(String input) {
  int n = 5;
  return Pattern.matches("a{"+n+"}", input);
}
```

Esempi

Restituiamo true se la stringa in input è composta da 5 caratteri diversi da a

Esempi

Restituiamo true se la stringa in input è composta da 5 caratteri diversi da a

```
public boolean test9(String input) {
  if(input.length() != 5)
    return false;
  for(int i = 0; i < 5; i++)
    if(input.charAt(i) == 'a')
      return false;
  return true;
}
```

Esempi

Restituiamo true se la stringa in input è composta da 5 caratteri diversi da a

```
public boolean test9(String input) {
  if(input.length() != 5)
    return false;
  for(int i = 0; i < 5; i++)
    if(input.charAt(i) == 'a')
      return false;
  return true;
}
```

... usando la classe Pattern?

Esempi

Restituiamo true se la stringa in input è composta da 5 caratteri diversi da a

```
public boolean test9(String input) {
  if(input.length() != 5)
    return false;
  for(int i = 0; i < 5; i++)
    if(input.charAt(i) == 'a')
      return false;
  return true;
}
```

... usando la classe Pattern?

```
public boolean test9(String input) {
  return Pattern.matches("[^a]{5}", input);
}
```

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole

```
public boolean test10(String input) {
  return Pattern.matches("[a-z]+", input);
}
```

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole

```
public boolean test10(String input) {
  return Pattern.matches("[a-z]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere maiuscole

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole

```
public boolean test10(String input) {
  return Pattern.matches("[a–z]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere maiuscole

```
public boolean test11(String input) {
  return Pattern.matches("[A–Z]+", input);
}
```

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole

```
public boolean test10(String input) {
  return Pattern.matches("[a–z]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere maiuscole

```
public boolean test11(String input) {
  return Pattern.matches("[A–Z]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole e maiuscole

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole

```
public boolean test10(String input) {
  return Pattern.matches("[a–z]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere maiuscole

```
public boolean test11(String input) {
  return Pattern.matches("[A–Z]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole e maiuscole

```
public boolean test12(String input) {
  return Pattern.matches("[a–zA–Z]+", input);
}
```

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole comprese tra la lettera a e la lettera d, oppure tra la lettera f e la lettera h

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole comprese tra la lettera a e la lettera d, oppure tra la lettera f e la lettera h

```
public boolean test13(String input) {
  return Pattern.matches("[a-d[f-h]]+", input);
}
```

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole comprese tra la lettera a e la lettera d, oppure tra la lettera f e la lettera h

```
public boolean test13(String input) {
  return Pattern.matches("[a-d[f-h]]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole tra a e z tranne p e q

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole comprese tra la lettera a e la lettera d, oppure tra la lettera f e la lettera h

```
public boolean test13(String input) {
  return Pattern.matches("[a-d[f-h]]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole tra a e z tranne p e q

```
public boolean test14(String input) {
  return Pattern.matches("[a-z&&[^pq]]+", input);
}
```

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole comprese tra la lettera a e la lettera d, oppure tra la lettera f e la lettera h

```
public boolean test13(String input) {
  return Pattern.matches("[a-d[f-h]]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole tra a e z tranne p e q

```
public boolean test14(String input) {
  return Pattern.matches("[a-z&&[^pq]]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole tra a e z tranne quelle tra p e t

Esempi

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole comprese tra la lettera a e la lettera d, oppure tra la lettera f e la lettera h

```
public boolean test13(String input) {
  return Pattern.matches("[a-d[f-h]]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole tra a e z tranne p e q

```
public boolean test14(String input) {
  return Pattern.matches("[a-z&&[^pq]]+", input);
}
```

Restituiamo true se la stringa in input è una sequenza (non vuota) di lettere minuscole tra a e z tranne quelle tra p e t

```
public boolean test15(String input) {
  return Pattern.matches("[a-z&&[^p-t]]+", input);
}
```

Esempi

Restituiamo true se la stringa è un indirizzo email di mat.unical (per semplicità assumiamo che gli indirizzi email possano contenere solo lettere)

Esempi

Restituiamo true se la stringa è un indirizzo email di mat.unical (per semplicità assumiamo che gli indirizzi email possano contenere solo lettere)

```
public boolean testEmail(String input) {
 return Pattern.matches("[a–zA–Z]+@mat\\.unical\\.it", input);
}
```

Esempi

Restituiamo true se la stringa è un indirizzo email di mat.unical (per semplicità assumiamo che gli indirizzi email possano contenere solo lettere)

```
public boolean testEmail(String input) {
 return Pattern.matches("[a-zA-Z]+@mat\\.unical\\.it", input);
}
```

Restituiamo true se la stringa è una dichiarazione di una variabile intera in Java

Esempi

Restituiamo true se la stringa è un indirizzo email di mat.unical (per semplicità assumiamo che gli indirizzi email possano contenere solo lettere)

```
public boolean testEmail(String input) {
 return Pattern.matches("[a-zA-Z]+@mat\\.unical\\.it", input);
}
```

Restituiamo true se la stringa è una dichiarazione di una variabile intera in Java

```
public boolean testDichiarazione(String input) {
 return Pattern.matches("int [a-zA-Z_][a-zA-Z_0-9]*;", input);
}
```

Altri elementi

Costrutto	Elemento
.	Ogni carattere tranne il terminatore di linea
\d	Un numero tra 0 e 9
\D	Un carattere che non sia un numero

In un'espressione regolare questi elementi vanno preceduti da un altro \
Ad esempio `\\d` indica un qualsiasi numero tra 0 e 9

Altri elementi

Costrutto	Elemento
.	Ogni carattere tranne il terminatore di linea
\d	Un numero tra 0 e 9
\D	Un carattere che non sia un numero

In un'espressione regolare questi elementi vanno preceduti da un altro \
Ad esempio `\\d` indica un qualsiasi numero tra 0 e 9

Esempio

Restituiamo true se la stringa in input è un indirizzo IP, dove un indirizzo IP è formato da 4 numeri di al massimo tre cifre separati da punto. Esempio: 192.168.1.10

Altri elementi

Costrutto	Elemento
.	Ogni carattere tranne il terminatore di linea
\d	Un numero tra 0 e 9
\D	Un carattere che non sia un numero

In un'espressione regolare questi elementi vanno preceduti da un altro \
Ad esempio `\\d` indica un qualsiasi numero tra 0 e 9

Esempio

Restituiamo true se la stringa in input è un indirizzo IP, dove un indirizzo IP è formato da 4 numeri di al massimo tre cifre separati da punto. Esempio: 192.168.1.10

```
public boolean testIndirizzoIp(String input) {
  return Pattern.matches("(\\d{1,3}\\.){3}\\d{1,3}", input);
}
```

Espressioni regolari e stringhe

```
String input = "192.168.1.2";
String res = input.replaceAll(".*1\\.", "");
System.out.println(res);
```

Espressioni regolari e stringhe

```
String input = "192.168.1.2";
String res = input.replaceAll(".*1\\.", "");
System.out.println(res);

Risultato della stampa: 2
```

Espressioni regolari e stringhe

```
String input = "192.168.1.2";
String res = input.replaceAll(".*1\\.", "");
System.out.println(res);

Risultato della stampa: 2

String input = "192.168.1.2";
String[] elements = input.split("\\.");
for(String s : elements)
   System.out.println(s);
```

Espressioni regolari e stringhe

```
String input = "192.168.1.2";
String res = input.replaceAll(".*1\\.", "");
System.out.println(res);

Risultato della stampa: 2

String input = "192.168.1.2";
String[] elements = input.split("\\.");
for(String s : elements)
   System.out.println(s);

Risultato della stampa:
192
168
1
2
```

Espressioni regolari e stringhe

```
String input = "192.168.1.2";
String res = input.replaceAll(".*1\\.", "");
System.out.println(res);

Risultato della stampa: 2

String input = "192.168.1.2";
String[] elements = input.split("\\.");
for(String s : elements)
   System.out.println(s);

Risultato della stampa:
192
168
1
2
```

Attenzione: non tutti i metodi interpretano espressioni regolari! Ad esempio il metodo replace riceve come parametro una stringa.