

# PRG – Pràctica 2

## Resolució d'alguns problemes amb recursió

Marisa Llorens (mllorens@dsic.upv.es)

Bluej: Bluej: Crida a mètode

```
// Determina si a és o no prefixe de b.  
// @param a String.  
// @param b l'altra String.  
// @return boolean, true si a és prefixe de b i, en cas contrari, false.  
boolean isPrefix(String a, String b)
```

PRGString.isPrefix( "rec" ,  
"recursion" )

Cancelar Aceptar

Bluej: Bluej: Crida a mètode

```
// Determina si a és o no subcadena de b.  
// @param a String.  
// @param b l'altra String.  
// @return boolean, true si a és subcadena de b i, en cas contrari, false.  
boolean isSubstring(String a, String b)
```

PRGString.isSubstring( "curs" ,  
"recursion" )

Cancelar Aceptar

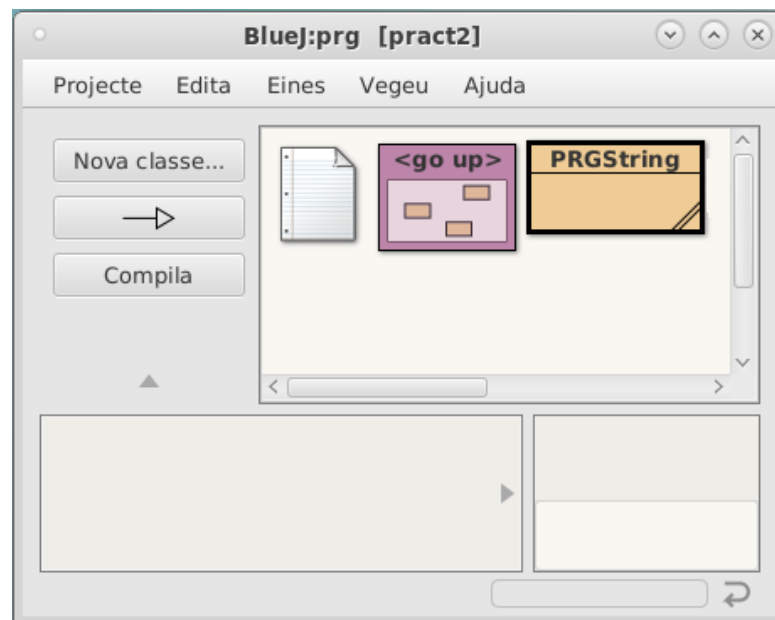
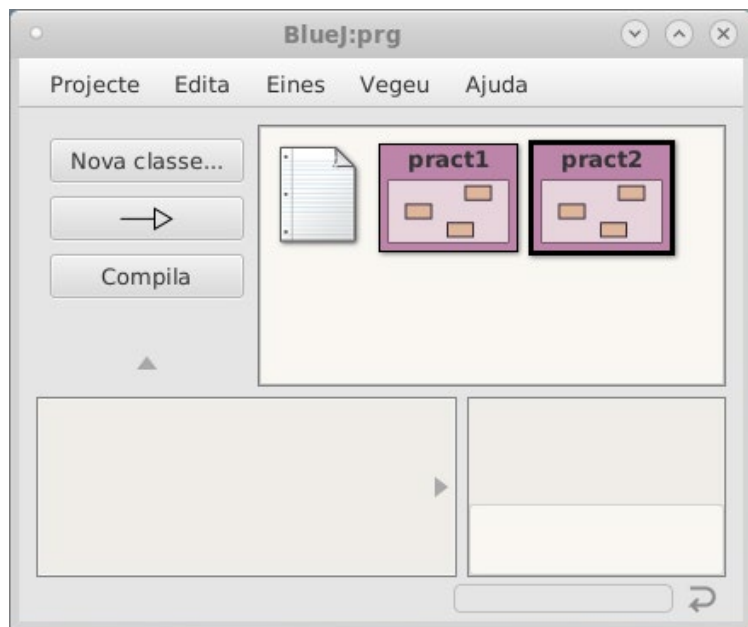
# 1 Context i treball previ

En aquesta pràctica es proposa la resolució de forma recursiva de dos problemes amb `Strings`. Per a això, es dissenyaran els mètodes corresponents i les classes de prova per assegurar que les solucions dels problemes siguin correctes.

És convenient haver estudiat la secció 10.6 *Recursividad con objetos de tipo String* de la 3<sup>a</sup> edició del llibre de l'assignatura<sup>1</sup> i haver comprès alguns exemples com el problema de comptar el nombre de caràcters 'a' en certa `String s`.

## Activitat 1: Creació del paquet BlueJ pract2

Obre el projecte *BlueJ* de treball de l'assignatura (prg) i crea un nou paquet `pract2`. Agrega al paquet el fitxer `PRGString.java` que hauràs descarregat prèviament de la carpeta `Recursos/Laboratorio/Práctica 2` de la PoliformaT de PRG. La classe `PRGString` és una classe d'utilitats que inclou els mètodes que resolen el problema de comptar el nombre de 'a's en una `String s` (secció 10.6 del llibre) i els mètodes (a completar) que resolen els problemes que se't plantegen a continuació.



## Method Summary

All Methods	Static Methods	Instance Methods	Concrete Methods	Deprecated Methods
Modifier and Type	Method	Description		
char	<b>charAt</b> (int index)	Returns the char value at the specified index.		
int	<b>length</b> ()	Returns the length of this string.		
<b>String</b>	<b>substring</b> (int beginIndex)	Returns a string that is a substring of this string.		
<b>String</b>	<b>substring</b> (int beginIndex, int endIndex)	Returns a string that is a substring of this string.		

### substring

```
public String substring(int beginIndex)
```

Returns a string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

Examples:

```
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)
```

#### Parameters:

**beginIndex** - the beginning index, inclusive.

#### Returns:

the specified substring.

#### Throws:

**IndexOutOfBoundsException** - if **beginIndex** is negative or larger than

### substring

```
public String substring(int beginIndex, int endIndex)
```

Returns a string that is a substring of this string. The substring begins at the specified **beginIndex** and extends to the character at index **endIndex** - 1. Thus the length of the substring is **endIndex**-**beginIndex**.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

#### Parameters:

**beginIndex** - the beginning index, inclusive.

**endIndex** - the ending index, exclusive.

#### Returns:

the specified substring.

#### Throws:

**IndexOutOfBoundsException** - if the **beginIndex** is negative, or **endIndex** is larger than the length of this **String** object, or **beginIndex** is larger than **endIndex**.

```
package pract2;
```

```
/**
 * Classe PRGString: classe d'utilitats amb mètodes
 * per a treballat amb Strings.
 *
 * @author PRG - ETSINF - DSIC - UPV
 * @version Curs 2019/2020
 */
public class PRGString {

    /** No hi ha objectes d'aquesta classe. */
    private PRGString() { }

    /**
     * Torna el número de 'a's en la String donada.
     * @param s String on es volen comptar les 'a's.
     * @return int.
     */
    public static int countA(String s) {
        // Cas base: String buida
        if (s.length() == 0) { return 0; }
        // Cas general: String no buida. Tractar la substring posterior.
        else if (s.charAt(0) == 'a') { return 1 + countA(s.substring(1)); }
        else { return countA(s.substring(1)); }
    }

    /**
     * Torna el número de 'a's en la String donada.
     * @param s String on es volen comptar les 'a's.
     * @return int.
     */
    public static int countA2(String s) {
        // Cas base: String buida
        if (s.length() == 0) { return 0; }
        // Cas general: String no buida. Tractar la substring anterior.
        else if (s.charAt(s.length() - 1) == 'a') {
            return 1 + countA2(s.substring(0, s.length() - 1));
        } else { return countA2(s.substring(0, s.length() - 1)); }
    }
}
```

## 2 Problema A. *Prefixe*

Donades dues `Strings` `a` i `b`, potencialment buides, es diu que `a` és *prefixe* de `b` quan tots els caràcters de `a` estan consecutius, en el mateix ordre original, al començament de `b`.

Conseqüència de la definició anterior és que la *cadena buida* és prefixe de qualsevol altra, fins i tot si aquesta altra també estigués buida. Nota, d'altra banda, que una cadena no pot ser prefixe d'una altra si la primera és de longitud més gran que la segona.

### Activitat 2: mètode `isPrefix(String, String)`

Defineix recursivament un mètode `isPrefix(String, String)` per comprovar si una cadena és prefixe d'una altra. Per a això:

- Estableix els casos base i general de la recursió definint, a més, la solució del problema en cadascun d'aquests casos. La capçalera del mètode (en la que no hi ha paràmetres posicionals) haurà de ser necessàriament la que segueix:

```
public static boolean isPrefix(String a, String b)
```

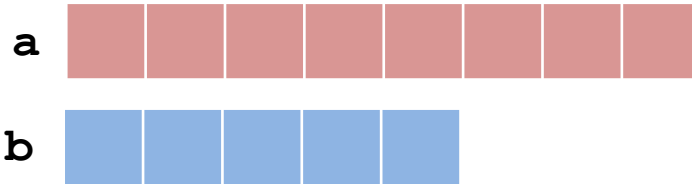
- Documenta adequadament el mètode, explicitant quins són els seus paràmetres, el tipus del seu resultat i, cas d'haver-ne, la seua precondició.
- Comprova que el codi del mètode segueix les normes d'estil usant el *Checkstyle* de *BlueJ* i corregeix-lo si no és el cas.

```
/**
 * Determina si a és o no prefixe de b.
 * -- COMPLETAR --
 */
public static boolean isPrefix(String a, String b) {
    /* COMPLETAR */
    return true;
}
```

# Análisi de casos

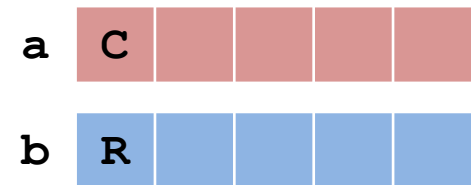
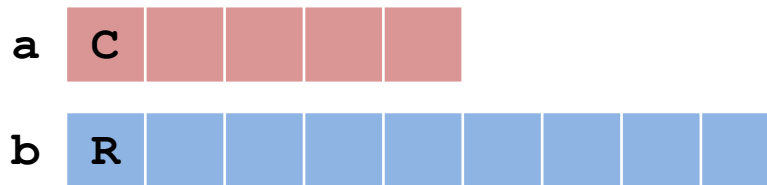
## Cas base:

- La **String** **a** és buida, **a** és **prefixe** de **b**. **a** ""
- La **String** **a** és de longitud major que la **String** **b**, **a** **no** és **prefixe** de **b**.

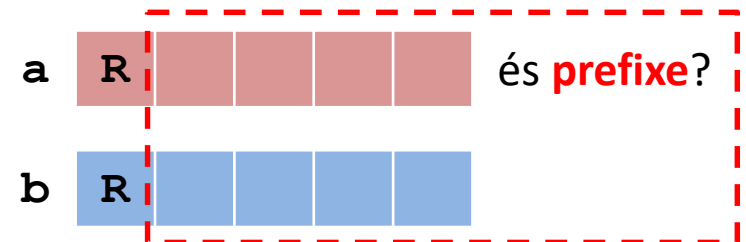
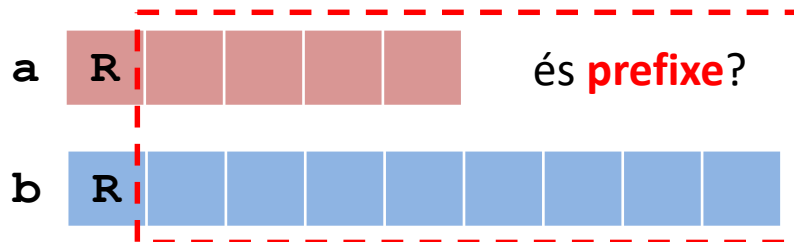


## Cas general o recursiu:

- La **String** **a** no és buida ni de longitud major que la **String** **b**,
  - Si el primer caràcter de **a** **no** coincideix amb el primer caràcter de **b**, **a** **no** és **prefixe** de **b**.



- Si el primer caràcter de **a** **sí** coincideix amb el primer de **b**, aleshores **a** és **prefixe** de **b** si la **substring** de **a** des del caràcter **1** endavant és **prefixe** de la **substring** de **b** des del caràcter **1** endavant. En caso contrari, **a** **no** és **prefixe** de **b**.



### Activitat 3: validació del mètode isPrefix(String, String)

Escriu una classe programa `TestIsPrefix` que permeti executar el mètode amb diferents dades per tal de comprovar que no hi ha errors d'execució i que el resultat que torna en cada cas és el correcte.

Les dades a provar han de reflectir les diferents situacions que es poden donar en l'execució del mètode, com ara, per exemple: que ambdues cadenes estiguen buides, que ho estiga només una, que la primera cadena siga més llarga que la segona, que la primera cadena siga prefixe o no de la segona, etc. A la taula següent es detallen els diferents casos, amb instàncies concretes i el resultat esperat per a cada cas.

Cas	a	b	Resultat
a i b buides	""	""	true
Només a buida	""	"recursion"	true
Només b buida	"recursion"	""	false
a de major longitud que b	"recursion"	"rec"	false
a i b de la mateixa longitud i a és prefixe de b	"recursion"	"recursion"	true
a i b de la mateixa longitud i a no és prefixe de b	"123456789"	"recursion"	false
a de menor longitud que b i a és prefixe de b	"rec"	"recursion"	true
a de menor longitud que b i a no és prefixe de b:			
- pel primer caràcter	"pecur"	"recursion"	false
- pel darrer caràcter	"recurso"	"recursion"	false
- per un caràcter intermedi	"remursi"	"recursion"	false

La classe `TestIsPrefix` ha d'incloure un mètode amb el següent perfil:

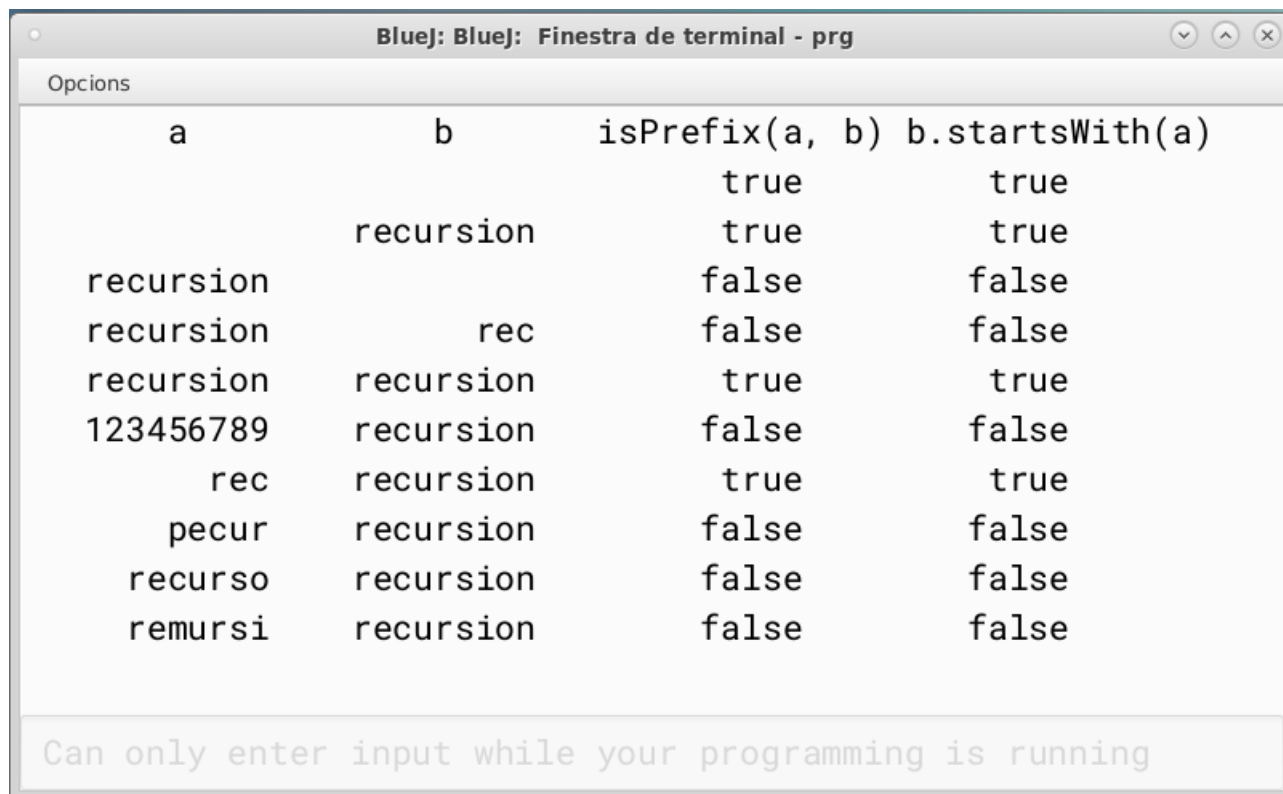
```
private static void testIsPrefix(String a, String b)
```

que mostre per pantalla les `Strings` de prova, el resultat del teu mètode `isPrefix(String, String)` i el resultat esperat. Per a això últim, pots utilitzar el mètode `startsWith(String)` de la classe `String`.

boolean      `startsWith(String prefix)`

Tests if this string starts with the specified prefix.

El `main` ha d'invocar al mètode `testIsPrefix(String, String)` per a cada cas de prova. Pots definir un array de `String` per emmagatzemar les diferents instàncies dels casos a provar.



a	b	isPrefix(a, b)	b.startsWith(a)
		true	true
	recursion	true	true
recursion		false	false
recursion	rec	false	false
recursion	recursion	true	true
123456789	recursion	false	false
rec	recursion	true	true
pecur	recursion	false	false
recurso	recursion	false	false
remursi	recursion	false	false

Can only enter input while your programming is running



```

public class TestIsPrefix {
    /** No hi ha objectes d'aquesta classe. */
    private TestIsPrefix() { }

    public static void main(String[] args) {
        String[] s = {"", "rec", "pecur", "recurso", "remursi",
                     "123456789", "recursion"};

        System.out.printf("%8s %12s %20s %12s\n",
                          "a", "b", "isPrefix(a, b)", "b.startsWith(a)");

        // a i b buides
        testIsPrefix(s[0], s[0]);
        // nomes a buida

        // nomes b buida

        // a de major longitud que b

        // a i b de la mateixa longitud i a és prefixe de b

        // a i b de la mateixa longitud i a no és prefixe de b

        // a de menor longitud que b i a és prefixe de b

        // a de menor longitud que b i a no és prefixe de b:
        // pel primer caràcter

        // a de menor longitud que b i a no és prefixe de b:
        // pel darrer caràcter

        // a de menor longitud que b i a no és prefixe de b:
        // per un caràcter intermedi
    }
}

```

```

private static void testIsPrefix(String a, String b) {
}

```

### 3. Problema B. *Subcadena*

Donades dues `Strings` `a` i `b`, potencialment buides, es diu que `a` és *subcadena* de `b` quan tots els caràcters de `a` estan consecutius, en el mateix ordre original, en algun lloc de `b`. O, el que és el mateix, quan `a` és prefixe de `b` o d'alguna de les possibles subcadenaes de `b`.

Naturalment, com passava en el cas de `isPrefix(String, String)`, es pot veure que la *cadena buida* és subcadena de qualsevol altra, encara que aquella altra també estigués buida. A més, una cadena no pot ser subcadena d'una altra si la primera és de longitud més gran que la segona.

#### Activitat 4: mètode `isSubstring(String, String)`

Defineix recursivament, en termes de `isPrefix(String, String)`, el mètode `isSubstring(String)`, per tal de poder comprovar si una cadena és subcadena d'una altra. Per a això:

- Enuncia els casos base i general de la recursió, definint la solució del problema en cada cas. La capçalera del mètode ha de ser necessàriament:

```
public static boolean isSubstring(String a, String b)
```

Nota que, igual que per a l'operació `isPrefix(String, String)`, no hi ha paràmetres posicionals en la capçalera anterior.

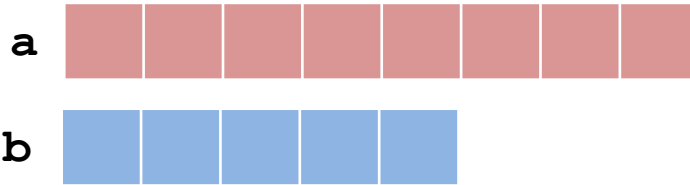
- Documenta adequadament el mètode, explicitant quins són els seus paràmetres, el tipus del seu resultat i, cas d'haver-ne, la seua precondició.
- Comprova que el codi del mètode segueix les normes d'estil usant el *Checkstyle* de *BlueJ* i corregeix-lo si no és el cas.

```
/**
 * Determina si a és o no subcadena de b.
 * -- COMPLETAR --
 */
public static boolean isSubstring(String a, String b) {
    /* COMPLETAR */
    return true;
}
```

# Análisi de casos

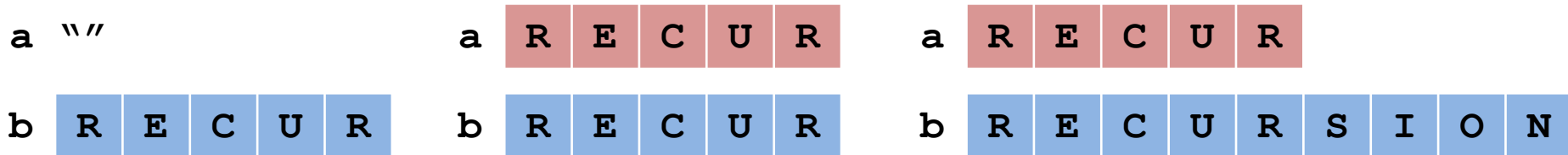
## Cas base:

- La **String** **a** és buida, **a** és **subcadena** de **b**.      **a** ""
- La **String** **a** és de longitud major que la **String** **b**, **a** **no** és **subcadena** de **b**.



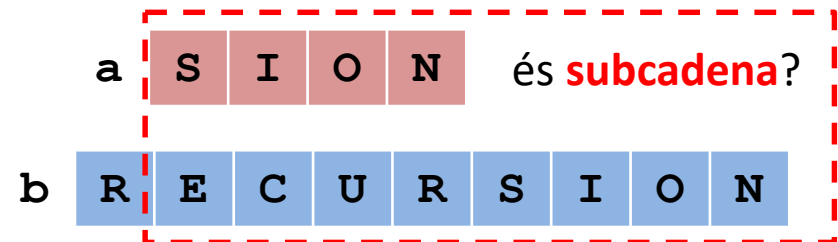
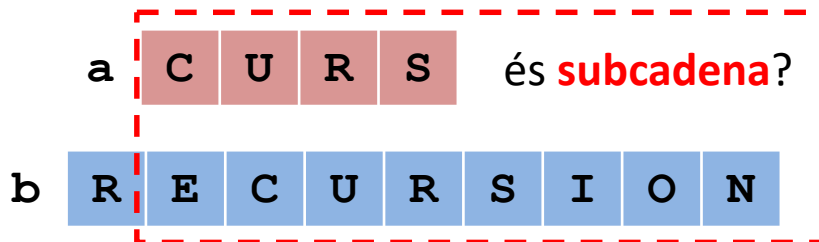
## Cas general o recursiu:

- La **String** **a** és de longitud menor o igual que la **String** **b**, **a** és **subcadena** de **b**
  - si **a** és **prefixe** de **b**,



O

- si **a** és **subcadena** de la **substring** de **b** des del caràcter 1 endavant.



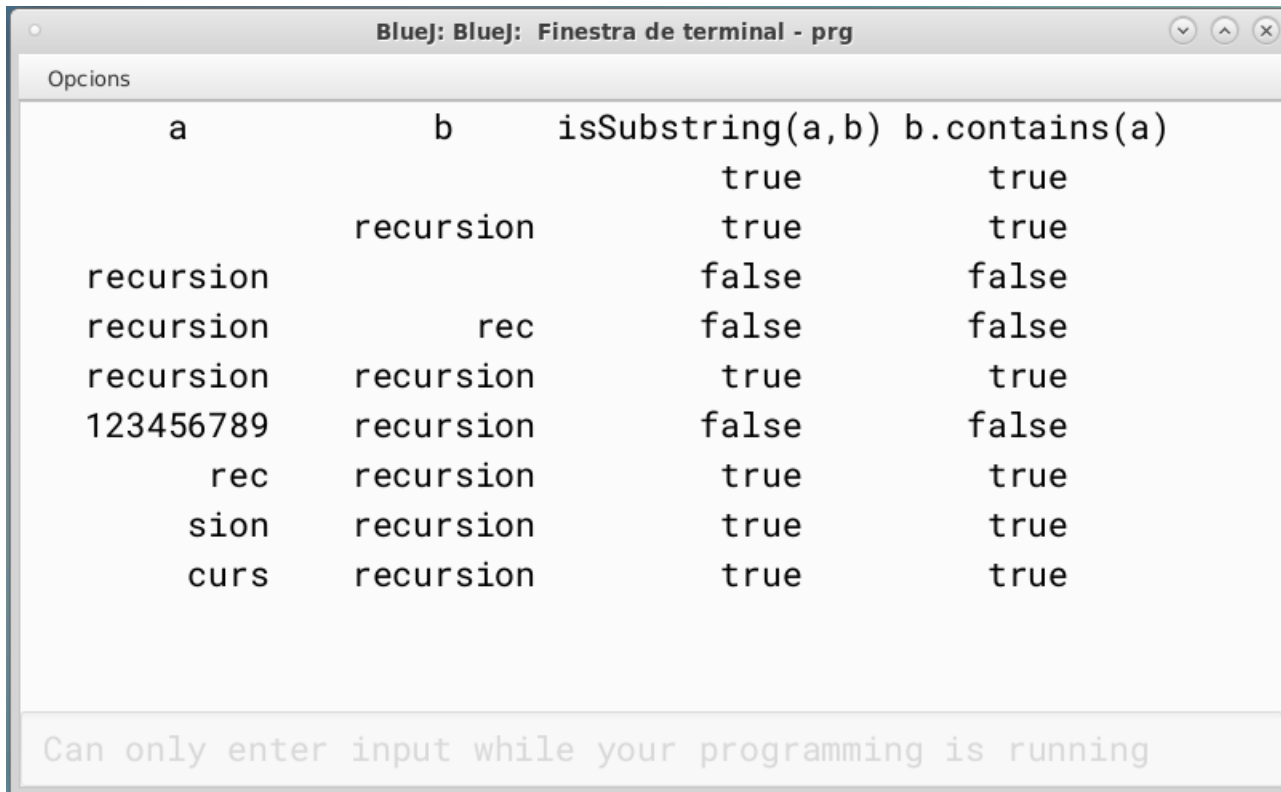
## Activitat 5: validació del mètode `isSubstring(String, String)`

Escriu una classe programa `TestIsSubstring` que permeti executar el mètode amb diferents dades per tal de comprovar que no hi ha errors d'execució i que el resultat que torna en cada cas és el correcte. Com abans, has d'identificar les diferents situacions que es poden donar, provant que, en totes elles, el mètode funciona adequadament. En aquest cas, pots comparar el resultat amb el del mètode `contains(String)` de la classe `String`.

boolean

`contains(CharSequence s)`

Returns true if and only if this string contains the specified sequence of char values.



```
BlueJ: BlueJ: Finestra de terminal - prg
Opcions
a          b          isSubstring(a,b)  b.contains(a)
                                true          true
                                recursion      true          true
recursion                                false      false
recursion          rec          false      false
recursion          recursion      true          true
123456789          recursion      false      false
          rec          recursion      true          true
          sion          recursion      true          true
          curs          recursion      true          true

Can only enter input while your programming is running
```

```

public class TestIsSubstring {
    /** No hi ha objectes d'aquesta classe. */
    private TestIsSubstring() { }

    public static void main(String[] args) {
        String[] s = {"", "rec", "pecur", "recurso", "remursi",
                     "123456789", "recursion", "sion", "curs"};

        System.out.printf("%8s %12s %20s %10s\n",
                          "a", "b", "isSubstring(a,b)", "b.contains(a)");

        // a i b buides
        testIsSubstring(s[0], s[0]);
        // només a buida
        // només b buida
        // a de major longitud que b
        // a i b de la mateixa longitud i a és subcadena de b
        // a i b de la mateixa longitud i a no és subcadena de b
        // a de menor longitud que b i a és sucadena de b
        // perquè a és prefixe de b
        // a de menor longitud que b i a és sucadena de b
        // perquè a és sufixe de b
        // a de menor longitud que b i a és sucadena de b
        // perquè a està en b a partir d'una posició intermèdia
    }
}

```