String

Entin Martiana

String

- Menampilkan teks pada aplikasi.
- Text adalah salah satu cara yang paling mudah dan paling umum untuk menyampaikan pesan dari dan untuk user.
- Java menyediakan (API) untuk berinteraksi dengan string.

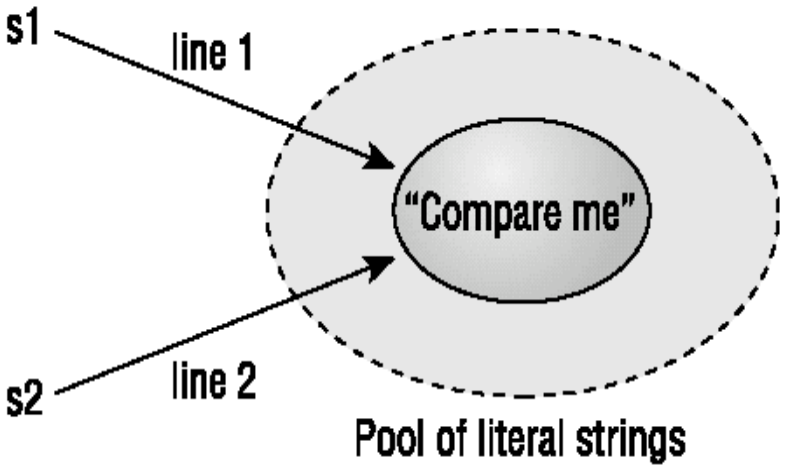
The *String* Class

- Class String berisi string yang tetap (immutable string).
- Sekali intance String dibuat maka isinya tidak bisa diubah.
- Memiliki beberapa konstruktor.
- Common string constructors:

```
String s1 = new String("immutable");
String s1 = "immutable";
```

- Java mempunyai media penyimpanan literal string yang yang disebut "**pool**".
- Jika suatu literal string sudah ada di pool, Java "tidak akan membuat copy lagi".

Identical literals



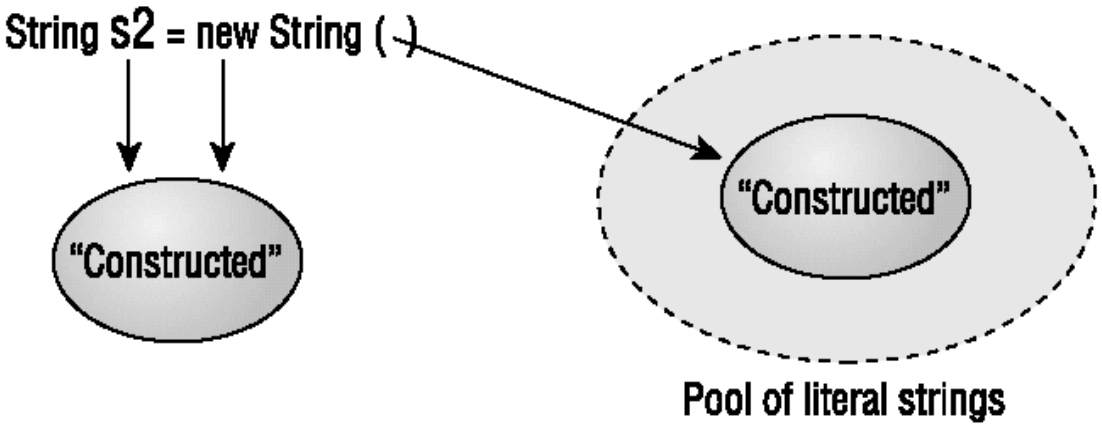

```
1. String s1 = "Compare me";
2. String s2 = "Compare me";
3. if (s1.equals(s2)) {
4.     // whatever
5. }
```

```
1. String s1 = "Compare me";
2. String s2 = "Compare me";
3. if (s1 == s2)) {
4.     // whatever
5. }
```

- Kedua potongan program diatas OK
- Contoh pertama membandingkan contentnya.
- Contoh kedua membandingkan referencesnya.

Explicitly calling the String constructor

```
String s2 = new String("Constructed");
```



- Pada saat kompile → "Constructed" disimpan di pool tersendiri.
- Pada saat runtime → statement new String() dieksekusi, "Constructed" akan dibuatkan lagi di program space.

Penyimpanan dan Kekekalan String

- obyek String bukanlah string itu sendiri, ia hanya merujuk ke lokasi memori yang berisi karakter string.

String a = “hello”;
String b = “hello”;
String c = “hello”;

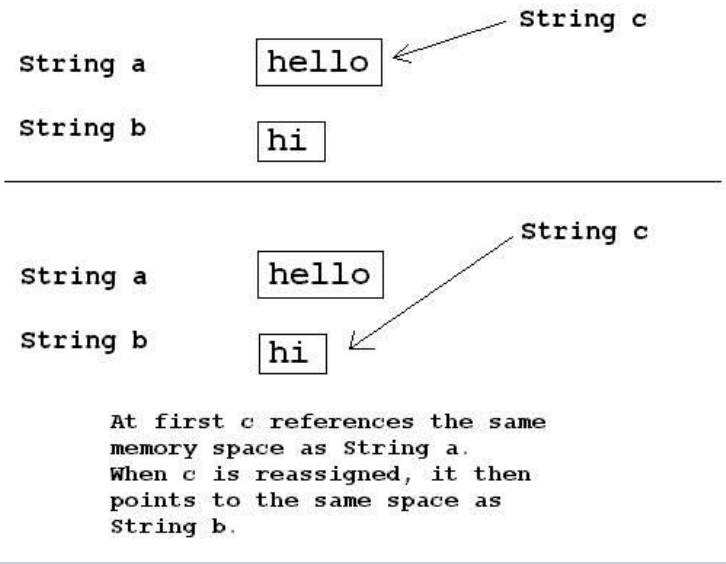
- Untuk menghemat tempat dan mengurangi kompleksitas, walaupun tiga string yang diinisialisasi, compiler Java hanya membuat satu ruang memori untuk menyimpan teks hello. Ruang memori digunakan bersama-sama oleh tiga obyek String, sedangkan variabel a, b, dan c hanya pointer yang menunjuk ke lokasi memori tersebut.

Penyimpanan dan Kekekalan String

Perubahan satu string jelas akan mempengaruhi dua string yang lain.
Karena itu dengan tidak diijinkannya manipulasi string secara langsung, Lingkungan Java mencegah perubahan satu string berdampak pada string yang lainnya.

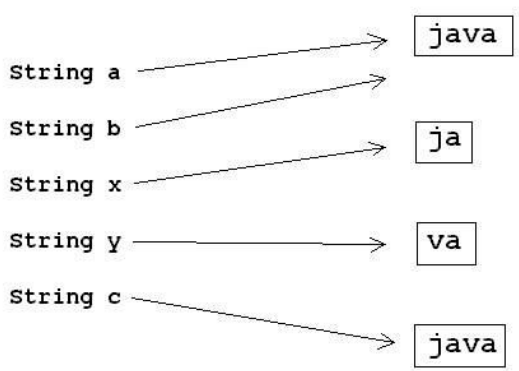
```
String a = "hello";
String b = "hi";
String c = a;
c = b;
```

Dua lokasi memori untuk menempatkan string. String pertama berisi hello String kedua berisi hi, Variabel String c menunjuk String pertama kemudian menunjuk ke String yang lainnya



Penyimpanan dan Kekekalan String

```
1.  public class Test {
2.     public static void main(String args[]) {
3.         String a = "java";
4.         String b = "java";
5.         String x = "ja";
6.         String y = "va";
7.         String c = x + y;
8.         if (a == b)
9.            System.out.println("a and b are the same object");
10.        else
11.           System.out.println("a and b are not the same object");
12.
13.        if (b == c)
14.           System.out.println("a and c are the same object");
15.        else
16.           System.out.println("a and c are not the same object");
17.    }
18. }
```



Because Strings a and b are allocated at compile time and have the same value, they will reference the same memory space. String c is allocated at run time and will be stored in a separate space.

Ketika source Java dieksekusi, isi variabel String a, b, dan c adalah sama : "java." Tetapi, Variabel a,b,dan c tidak akan menunjuk lokasi memori yang sama, Dan karena itu perbandingan kedua akan gagal. Output source code di atas :

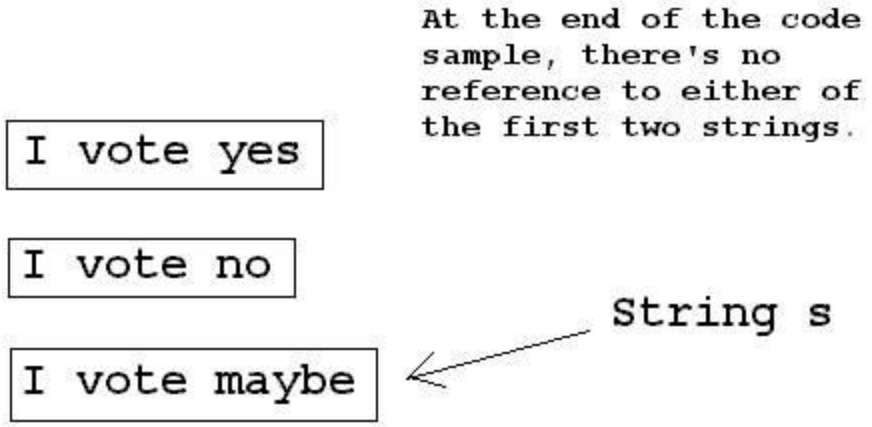
a and b are the same object
a and c are not the same object

Karena String a dan b didefinisikan saat compile dan isinya sama, maka referensi lokasi memorinya sama.String c dialokasikan saat run time dan akan Disimpan pada lokasi memori berbeda

Penyimpanan dan Kekekalan String

Pemakaian operator == tidak tepat dipakai untuk menyatakan kesamaan isi string. Seharusnya menggunakan method equals().
Programmer Java harus memperhatikan kebutuhan saat String dibuat.
Pada kasus dimana variabel string dipakai kembali, isi string mungkin saja hilang. contoh:

```
String y = "yes";
String n = "no";
String m = "maybe";
String s = "I vote " + y;
String s = "I vote " + n;
String s = "I vote " + m;
```



Konstruktor dan Method dari Kelas String

Tabel : Konstruktor yang dipakai untuk membuat obyek String

Syntax	Description	Example
String();	Inisialisasi dan membuat obyek String yang tidak berisi karakter. Sama dengan String(“”);	String();
String(String Value);	Inisialisasi dan membuat obyek String. Isikan string nya.	String(“Hello world.”);
String(StringBuffer buffer);	Isinya identik dengan serangkaian karakter yang disimpan pada obyek StringBuffer	String(new StringBuffer());

Konstruktor dan Method dari Kelas String

equals(String str)

Method equals () dari kelas String lebih dipilih untuk memeriksa kesamaan string. Method equals () mengembalikan nilai true jika parameter stringnya identik dengan sumber string jika tidak, memberikan nilai false.
Kesamaan string adalah case-sensitive, sehingga jika ada perbedaan akan dihasilkan false. Parameter null juga akan Memberikan nilai false.

```
String x = “hello”;
String y = “HellO”;
if (x.equals(y) == false) {
   System.out.println(“The two strings are not equal.”);
   System.out.println(“equals() is case sensitive”.);
}
```

equals() dan == untuk String

- Method equals() membandingkan content-nya
- == membandingkan alamatnya.

Konstruktor dan Method dari Kelas String

equalsIgnoreCase(String str)

Method equalsIgnoreCase () untuk menentukan kesamaan string tanpa memperhatikan penulisannya. Method mengembalikan nilai true jika kedua string sama, meskipun penulisannya berbeda. Jika tidak, method mengembalikan nilai false. Null argumen juga akan dikembalikan sebagai nilai false.

```
String x = "hello";
String y = "HellO";
if (x.equalsIgnoreCase(y)) {
   System.out.println("The two strings are equal.");
   System.out.println("equals() is case insensitive".);
}
```

Konstruktor dan Method dari Kelas String

compareTo(String str)

Method compareTo () digunakan untuk menempatkan string dalam urutan abjad. Jika string target lebih tinggi dalam urutan abjad dibandingkan string referensinya, method akan memberikan nilai negatif. Jika string target lebih rendah urutannya dalam alfabet, akan mengembalikan nilai positif. Jika string sama, maka akan mengembalikan nilai nol. Method ini umumnya digunakan dalam operasi penyortiran. Perbandingan didasarkan atas pernyataan karakter Unicode.

```
String s1 = “one”;
String s2 = “two”;
String s3 = “three”;
if (s1.compareTo(s2) < 0) {
   // Returns true, because the string “one” appears before “two”
   // in the alphabet.
   System.out.println(“one appears before two”);
}
if (s2.compareTo(s3) > 0) {
   // Returns true, because the string “two” appears after “three”
   // in the alphabet.
   System.out.println(“three appears before two”);
}
```

Konstruktor dan Method dari Kelas String

toUpperCase()

Method toUpperCase () menghasilkan string yang identik dengan yang string asal, kecuali bahwa setiap karakter huruf kecil diubah menjadi huruf besar. String yang asli tidak berubah, hanya string yang dikembalikan berbeda. Catatan: Jika string asal sudah uppercase, maka string yang dikembalikan sama dengan string asal.

```
String s1 = "HELLO WORLD";
String s2 = "hello world";
if (s1.equals(s2.toUpperCase())) {
   System.out.println("String s1 is equal to uppercase s2");
}
```

Konstruktor dan Method dari Kelas String

toLowerCase()

Method toLowerCase () menghasilkan string yang identik dengan yang string asal,
kecuali bahwa setiap karakter huruf besar diubah menjadi huruf kecil.
String yang asli tidak berubah, hanya string yang dikembalikan berbeda.
Catatan:
Jika string asal sudah lowercase, maka string yang dikembalikan
sama dengan string asal.

```
String s1 = "HELLO WORLD";
String s2 = "hello world";
if (s2.equals(s1.toLowerCase())) {
   System.out.println("Lowercase string s1 is equal to string s2");
}
```

Konstruktor dan Method dari Kelas String

charAt(int index)

Method charAt () mengembalikan karakter pada posisi tertentu dalam string. Posisi dimulai dari nol, sehingga s.charAt (0) mengembalikan karakter pertama dalam string, s.charAt (1) adalah karakter yang kedua, dan seterusnya sampai s.charAt (s.length () -1) , yang mengembalikan karakter terakhir.

```
String s = "hello world";
char c1, c2, c3;
c1 = s.charAt(0);  // returns 'h'
c2 = s.charAt(6);  // returns 'w'
c3 = s.charAt(10); // returns 'd'
```

Konstruktor dan Method dari Kelas String

substring(int start)

Method substring () mengembalikan sebagian string dari string yang lain.
Method substring () memiliki dua bentuk.
Metode pertama menerima satu argumen mengembalikan sisanya
dari karakter yang ditunjukkan.

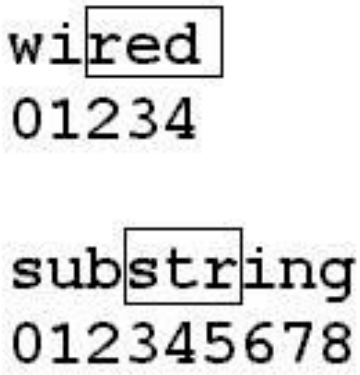
```
"nonfiction".substring(3); // returns "fiction"
```

Konstruktor dan Method dari Kelas String

substring(int start, int end)

method substring() yang kedua membutuhkan dua argumen dan Mengembalikan bagian string dari posisi argumen start s/d sebelum posisi argumen end. Hasilnya panjang string adalah = end-start.
Argumen zero based, jadi karakter pertama dicatat pada posisi 0, dan karakter terakhir dicatat pada posisi length()-1.

```
s1 = "wired".substring(2,4);  //  s1 is set to "red"
s2 = "substring".substring(3,5);  // s2 is set to "str"
```



"wired".substring(2,4) returns "red"

"substring".substring(3,5) returns "str"

Konstruktor dan Method dari Kelas String

indexOf(char ch),
indexOf(char ch, int index),
indexof(String s),
indexOf(String s, int index)

Method **indexOf()** mencari string, mengembalikan posisi pertama yang ditemukan dari string target. Seperti method yang lainnya zero based, posisi 0 menyatakan bahwa string target ditemukan pada posisi awal dari string asal. Jika karakter atau string target tidak ditemukan akan mengembalikan nilai **-1**. Pencarian adalah ***case sensitive***.

Method **indexOf()** juga dioverloaded untuk menerima parameter kedua yang menyatakan di mana posisi awal pencarian.Dapat juga dipakai untuk mencari string tertentu

```
String s = "Welcome to Java 2";
int x1, x2, x3, x4, x5;
x1 = s.indexOf("W");  // returns 0, first position
x2 = s.indexOf("e");  // returns 1, second position
// (e appears twice in the string, but IndexOf
// only returns the first occurrence)
x3 = s.indexOf("J");  // returns 11
x4 = s.indexOf("2"); // returns 16
x5 = s.indexOf("java"); // returns -1 because it's case-sensitive
```

Konstruktor dan Method dari Kelas String

lastIndexOf(char ch)

Method ***lasIndexOf()*** seperti indexOf(), mencari string atau karakter jika ditemukan, akan mengembalikan posisi pertama dari karakter target ditemukan.
Tetapi tidak seperti indexOf() pencarian lastIndexOf() dimulai di posisi terakhir dari string. Sedangkan pada indexOf() dipanggil dengan parameter kedua untuk mulai mencari pada lokasi yang lain melanjutkan dari kanan ke kiri.

```
String s = "hello world";
int x1, x2;
x1 = s.lastIndexOf("l");  //  Returns 9, the position of last l
x2 = s.lastIndexOf("o",5);  // Returns 4, the o in hello
```

Konstruktor dan Method dari Kelas String

startsWith(String str)

Method startsWith() menguji apakah string diawali dengan spesifik karakter tertentu. Alternatif pemanggilan method memulai pengujian dengan spesifik alamat dan mirip dengan substring(). Perhatikan contoh di bawah menunjukkan dua pemanggilan startsWith() yang keduanya bernilai true.

```
public class Test {
   public static void main(String args[]) {
      String s = "knowledge";
      if (s.startWith("know")) {  // returns true
         System.out.println("knowledge begins with know");
      }
      if (s.startsWith("led",4)) {  // returns true
         System.out.println("starting at position 4 it begins led");
      }
   }
}
```

Konstruktor dan Method dari Kelas String

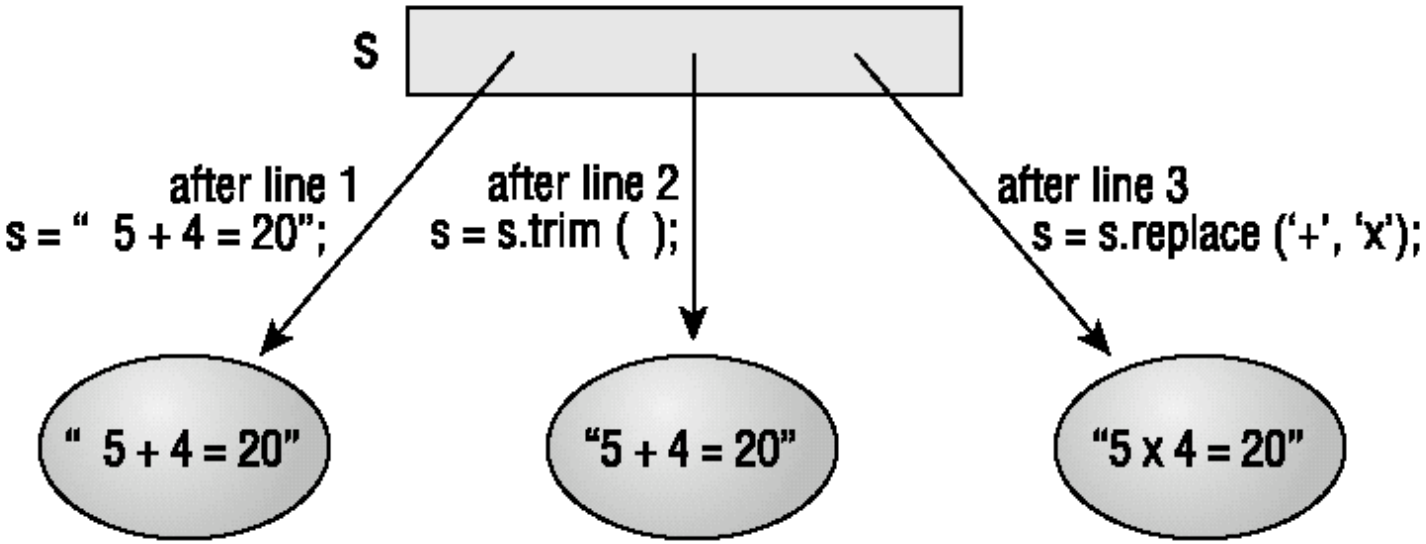
trim()

Method trim() digunakan untuk membuang spasi dari awal s/d akhir string. Tidak hanya spasi tetapi juga semua karakter control karakter ASCII (tabs/form feeds). String yang dihasilkan akan berisi spasi di dalam string, hanya saja awal dan akhir spasi akan dihilangkan.

```
String s1 = "   remove leading and trailing white space     ";
String s2 = s1.trim();
System.out.println(s2);
//  this line will print "remove leading and trailing white space"
```

Trimming and replacing

```
1. String s = " 5 + 4 = 20";
2. s = s.trim(); // "5 + 4 = 20"
3. s = s.replace('+', 'x'); // "5 x 4 = 20"
```



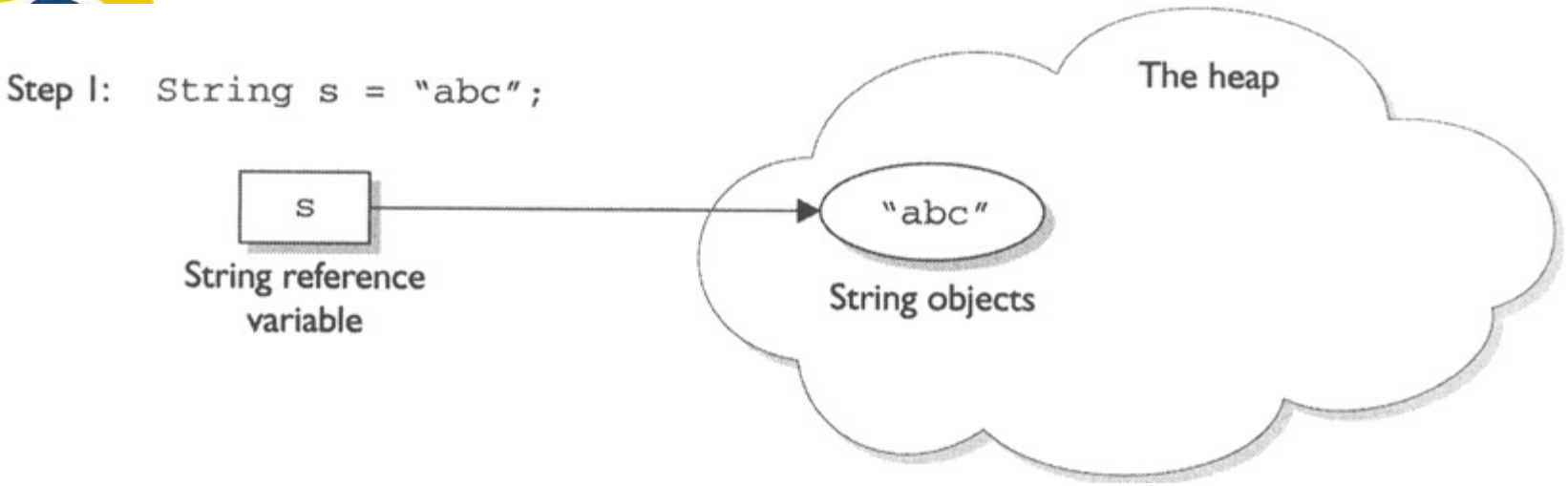
Membuat String di Java

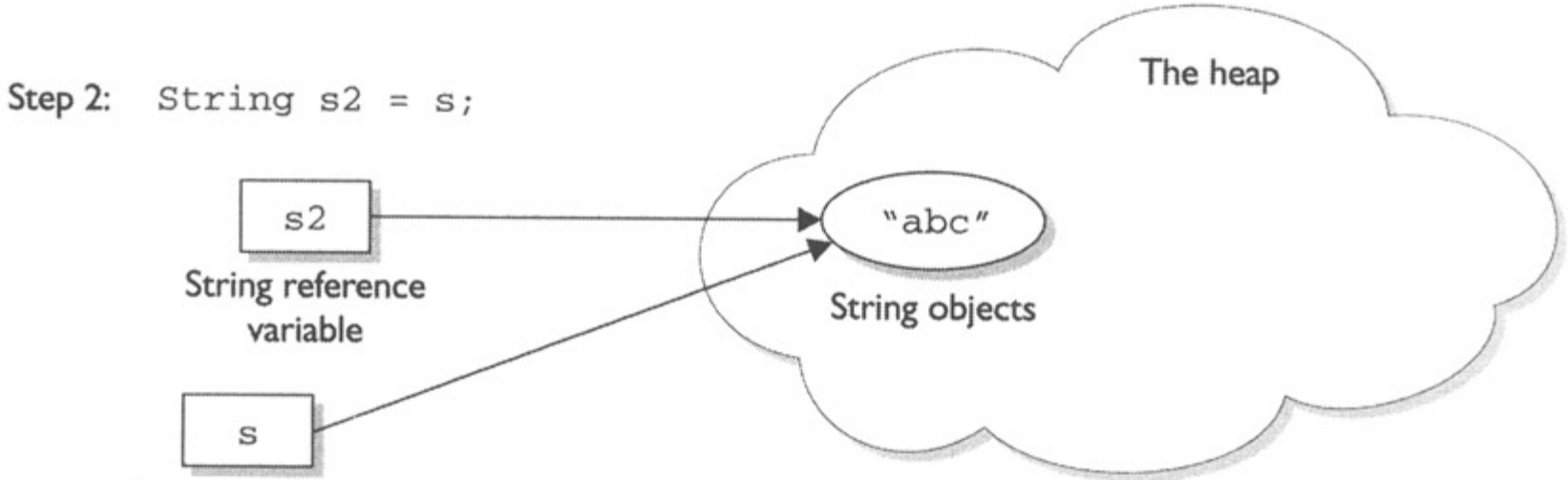
- method concat () : untuk menggabungkan string.

```
String a = "Halo ";
String b = "Nana";
String c = a.concat(b); // c = "Halo Nana"
```

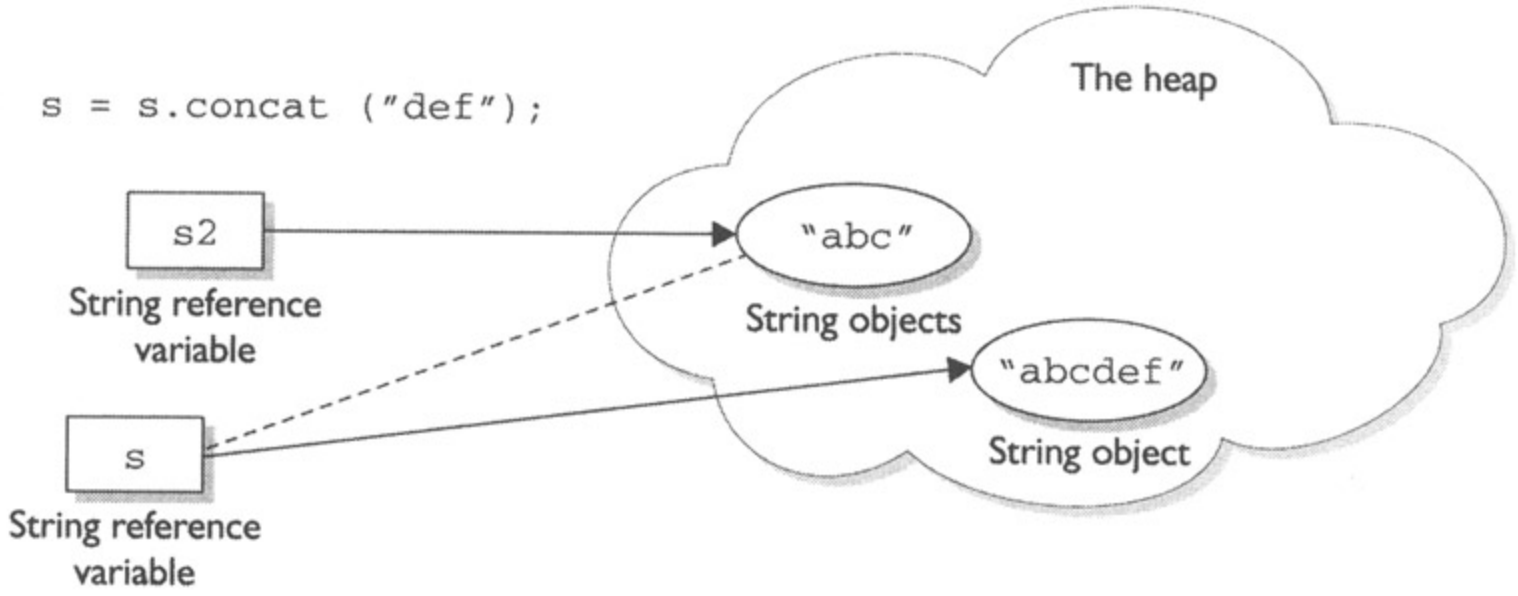
- Java menyediakan penulisan pendek untuk penggabungan string. Menggunakan overloading operator + (plus). Contoh kode ini memiliki hasil yang sama dengan a.concat(b).

```
String a = "hello ";
String b = "Pill Bug";
String c = a + b;
```


Step 1: String s = "abc";
The heap
s
"abc"
String reference variable
String objects


Step 2: String s2 = s;
The heap
s2
"abc"
String reference variable
String objects
s

Step 3: `s = s.concat ("def");`



Pemakaian Class StringBuffer

Masalah utama dengan class String adalah sifat kekekalannya. Lebih mudah dan efisien jika sebuah string dapat langsung diubah. Java menyediakan alternatif class yaitu StringBuffer, untuk mengatasi masalah ini. StringBuffer adalah string yang dapat dimodifikasi. StringBuffers digunakan secara internal untuk mengimplementasikan method-method yang ada dalam class String.

StringBuffer dapat membawa sejumlah karakter pada penciptaannya. Hal ini dikenal sebagai kapasitas. StringBuffer memiliki default kapasitas 16 karakter, tapi biasanya pemrogram mendefinisikan kapasitas pada penciptaan. Misalnya, untuk menciptakan StringBuffer kosong dengan kapasitas 100 karakter sbb :

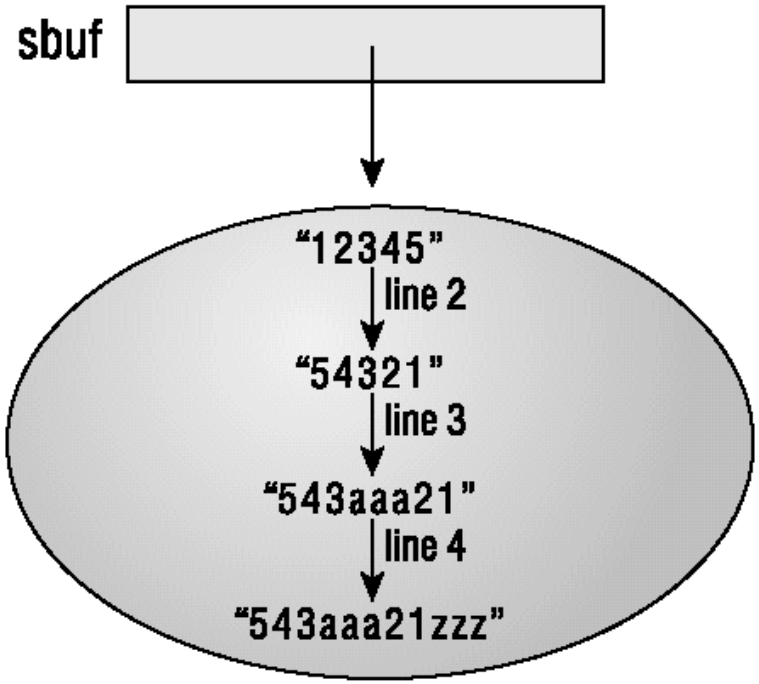
```
StringBuffer b = new StringBuffer(100);
```

StringBuffer dapat juga dibuat dari String yang sudah ada sbb :

.

Modifying a string buffer

```
1. StringBuffer sbuf = new StringBuffer("12345");
2. sbuf.reverse(); // "54321"
3. sbuf.insert(3, "aaa"); // "543aaa21"
4. sbuf.append("zzz"); // "543aaa21zzz"
```



Pemakaian Class StringBuffer

StringBuffer b = new StringBuffer(“hello”);

- Cara diatas akan membuat StringBuffer dengan kapasitas 21 (16 ditambah dengan panjang string argument) yang berisi string ”hello”.
- Kapasitas StringBuffer tidak terlalu penting, karena kapasitas StringBuffer
- bisa bertambah pada saat karakter ditambahkan walaupun melebihi kapasitas

Pemakaian Class StringBuffer

Sintak	Deskripsi	Contoh
StringBuffer()	Menginisialisasi dan membuat objek StringBuffer kosong dengan kapasitas default 16 karakter. Sama seperti jika kita membuat dengan sintak StringBuffer(16)	StringBuffer();
StringBuffer(int capacity)	Menginisialisasi dan membuat objek StringBuffer dengan kapasitas yang ditentukan	StringBuffer(100);
StringBuffer(String value)	Menginisialisasi dan membuat objek StringBuffer. Parameter berupa String dengan kapasitas 16 ditambah dengan panjang string.	StringBuffer(“Hello world.”);

Methods StringBuffer

capacity()

method capacity() dari class StringBuffer mengembalikan kapasitas objek StringBuffer pada saat ini.

```
String s = "hello world";
int c;
StringBuffer buf = new StringBuffer(s);
c = buf.capacity();  // will be set to 27
```

reverse()

methode reverse() untuk membalik isi dari objek StringBuffer.

```
StringBuffer buf = new StringBuffer("hello");
buf.reverse();
System.out.println(buf.toString());  // prints out the text 'olleh'
```

Class StringBuffer mempunyai method toString() untuk mengubah objek StringBuffer menjadi objek String.

Methods StringBuffer

setCharAt(int index, char ch)

- method setCharAt() mengubah sebuah karakter pada posisi tertentu menjadi karakter lain.
- Karakter pertama pada StringBuffer dimulai dari 0.
- ```
  StringBuffer buf = new StringBuffer("java");
  buf.setCharAt(0,'J');
  buf.setCharAt(2,'V');
  System.out.println(buf.toString());  // prints
  "JaVa"
  ```
  ```

Methods StringBuffer

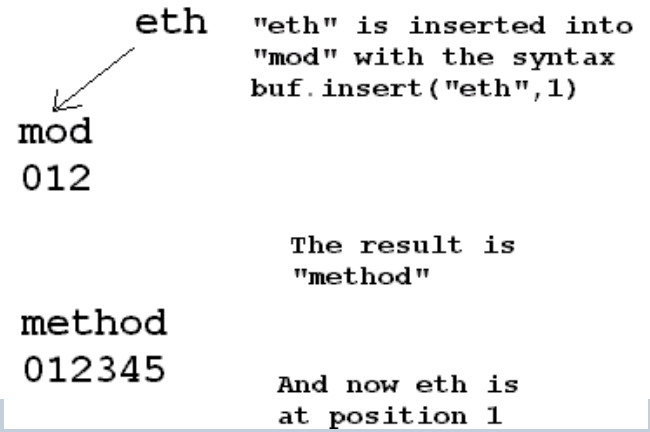
append(Object obj)

method append() untuk menambahkan String diakhir objek StringBuffer.

```
StringBuffer buf = new StringBuffer("Test");
buf.append("ing");
System.out.println(buf.toString());  // prints "Testing"
buf.append(123);
System.out.println(buf.toString());  // prints "Testing123"
```

insert(int index, Object obj)

method insert() dari class StringBuffer untuk menyisipkan string ke objek StringBuffer pada posisi tertentu.




```
StringBuffer buf = new StringBuffer("mod");
buf.insert(1,"eth");
System.out.println(buf.toString());  // prints "method"
```

Methods StringBuffer

delete(int start, int end)

method delete() untuk menghapus karakter-karakter pada StringBuffer.
Method ini mempunyai dua argumen yaitu posisi awal dan akhir.
Posisi akhir adalah karakter sebelum karakter pada posisi akhir yang akan dihapus.

```
public class Test {
   public static void main(String args[]) {
      StringBuffer buf = new
         StringBuffer("We'll delete characters from this sentence");
      buf.delete(0,13);
      buf.delete(10,99);
      System.out.println(buf.toString());  // displays "characters"
   }
}
```

buf.del(0,13) deletes the first thirteen characters.

```
We'll delete characters from this sentence
012345678901234567890123456789012345678901
000000000011111111112222222222333333333344
```

buf.del(10,99) deletes from position 10 to the end of the String.

```
characters from this sentence
012345678901234567890123456789
000000000011111111112222222222
```

Methods StringBuffer

length()

method length() mengembalikan panjang dari objek StringBuffer.
Jika objek StringBuffer kosong maka mengembalikan nilai 0.

```
StringBuffer buf = new StringBuffer(“”);
System.out.println(“The length is “ + buf.length());
// prints “The length is 0”
```

StringBuilder

- Class StringBuilder ditambahkan di Java 5
- Fungsi-fungsi yang ada di StringBuffer sama seperti fungsi yang ada di StringBuilder.
- Java Sun merekomendasikan menggunakan StringBuilder dibandingkan dengan StringBuffer karena StringBuilder dapat berjalan lebih cepat.
- Tapi StringBuilder ini tidak aman untuk multiple thread karena tidak mendukung sinkronisasi

Penggunaan StringBuilder dan StringBuffer

- ```
  String x = "abc"; x.concat("def");
  System.out.println("x = " + x);
  // output is "x = abc"
  ```

- ```
  String x = "abc"; x = x.concat("def");
  System.out.println("x = " + x);
  // output is "x = abcdef"
  ```

- ```
  StringBuffer sb = new StringBuffer("abc");
  sb.append("def"); System.out.println("sb = " +
  sb);
  // output is "sb = abcdef"
  ```

- ```
  StringBuilder sb = new StringBuilder("abc");
  sb.append("def").reverse().insert(3, "---");
  System.out.println( sb );
  // output is "fed --- cba"
  ```