STRING

PENGERTIAN STRING

- Obyek yang terdiri atas deretan karakter
- Tidak perlu ditentukan jumlah karakternya
- Inisialisasi :
  - Langsung tanpa kata kunci **new**
  - Langsung dengan kata kunci **new**

```
class TestStr1 {
   public static void main (String argv[]) {
       String s1;
       s1=”Halo”;

       String s2;
       s2= new String(”Ini juga bisa”);

       String s3 = ”Halo, apa khabar?”;
       System.out.println(s3);

       System.out.println(s1);
       System.out.println(s2);
       System.out.println(s3);

       s1 = s1 + ”, apa khabar?”;  //konkatenasi string
       System.out.println(s1);
   }
}
```

String null & String kosong

(tidak sama)

String s1;

s1 = null; → belum mempunyai alokasi memori untuk penempatan obyek tsb.

String s2;

s2 = ”” ; → String kosong (empty string) adalah String yg sudah diinisialisasi, namun tdk mpy karakter apapun.

Contoh beberapa metoda pada class String

int length() → **panjang dari string tsb**

char charAt(int index) → **karakter pada posisi index di string tsb**

```
public class Caristr {
  public static void main(String argv[]) {
    String str = "Ini adalah # String yang diproses";
    boolean found = false;
    int i;

    for (i = 0; i < str.length(); i++)
       if (str.charAt(i) == '#') {
          found = true;
          break;
       }

    if (found)
     System.out.println("Posisi # ada pada huruf ke " + i);
  }
}
```

int indexOf(char ch) → **posisi index pada String dimana karakter *ch* tsb ditemukan**

int indexOf(String s) → **posisi index pada String dimana String *s* ditemukan**

```
String str = ”Ini adalah # String” ;

int I = str.indexOf(’#’);
System.out.println(”Posisi=” + i);

int I = str.indexOf(”adalah”);
System.out.println(”Posisi=” + i);
```

Membandingkan String

```
String s1=”xxx”;
String s2=”yyy”;
```

```
s1 == s2
```
→ Membandingkan obyek

```
s1.equals(s2)
   atau
s2.equals(s1)
```
→ Membandingkan string

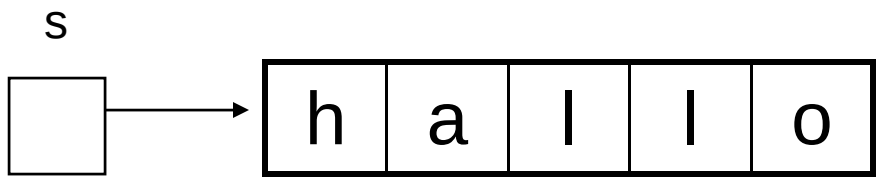
```
public class Compare {
   public static void main(String argv[]) {
   String tabel[] = {"amir", "bambang", "hasan"};
      String nama = "amir";

      for (int i = 0; i < tabel.length; i++) {
         if (tabel[i].equals(nama)) {
            System.out.println(nama + " terdaftar dalam
  tabel!");
         break;
         }
      }
   }
}
```

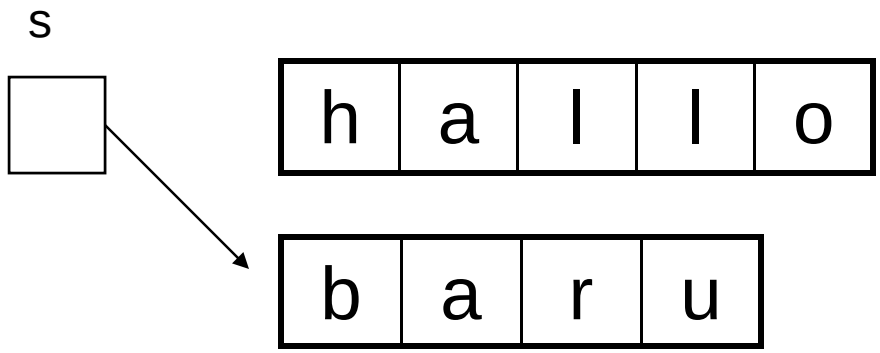
Modifikasi String

String adalah obyek yang bersifat “read only”, artinya tidak dapat diganti isinya.

```
String s = ”hallo”;
```




```
s = ”baru”;
```



Metoda untuk mengambil isi String :

String substring(int offset)
String substring(int offset, int endIndex)

```
public class SubStr {
   public static void main(String argv[]) {
      String s = "Kalimat ini akan disalin kemudian";

      String s1 = s.substring(5);
      String s2 = s.substring(8, 25);

      System.out.println("S1 = " + s1);
      System.out.println("S2 = " + s2);
   }
}
```

Konstruktor String

String dapat juga diciptakan melalui inisialisasi karakter array.

```
public class Strkonstruktor {
   public static void main(String argv[]) {
      char c_arr[] = {'h', 'a', 'l', 'o'};
      String s1 = new String(c_arr);

      String s2 = new String(c_arr, 2, 2);

      System.out.println("c_arr = " + c_arr);
      System.out.println("s1 = " + s1);
      System.out.println("s2 = " + s2);
   }
}
```

StringBuffer

- Karena class String adalah bersifat read-only, maka **class StringBuffer** digunakan untuk menampung byte yang dapat diubah isinya.
- Konstruktor :

  ```
  StringBuffer(int n)
  ```

  menciptakan Stringbuffer dengan array sebanyak n bytes
- Metoda :

  `append()` → menambah karakter pada stringBuffer

  `insert()` → menyelipkan potongan string pada sebuah String

```
public class StrBuf {
   public static void main(String argv[]) {
      String str = "Ini String Original";
      int len = str.length();
      StringBuffer strBuf = new StringBuffer(len);
      char ch;

      for (int i = (len-1); i >= 0; i--) {
         ch = str.charAt(i);
         strBuf.append(ch);
      }
      System.out.println(strBuf);
   }
}
```

Menyelipkan Teks

```
public class Insert {
   public static void main(String argv[]) {
      StringBuffer str = new
  StringBuffer("Minum Panas");

      str.insert(6, "Kopi ");
      System.out.println(str);
   }
}
```

Command Line Processor

- CLP adalah program yg menerima **input dari keyboard**
- Untuk membaca input dari keyboard dperlukan **BufferedReader.**
- Pertama definisikan dahulu obyek yg berasal dari **System.in** (keyboard) yg merupakan class dari **InputStream.** Agar dapat menggunakan **metoda readLine()** yg memberikan nilai balik String, obyek tsb perlu dibungkus dgn **BufferedReader.**

```
public class CLP {
   public static void main(String argv[]) {
      String command = "";
      java.io.InputStreamReader isr = new
                              java.io.InputStreamReader(System.in);
      java.io.BufferedReader buf = new
                              java.io.BufferedReader(isr);

      System.out.println("input : ");

         try {
            command = buf.readLine();
         }
         catch (java.io.IOException ex) {
            System.out.println("Error");
         }

      System.out.println("Yang dibaca : " + command);

   }
}
```