tutorial OO 2
StringBuilder, Scanner, Exceptions

Pol Van Aubel

Radboud University

working with text

STRINGS

String

```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

String

```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String

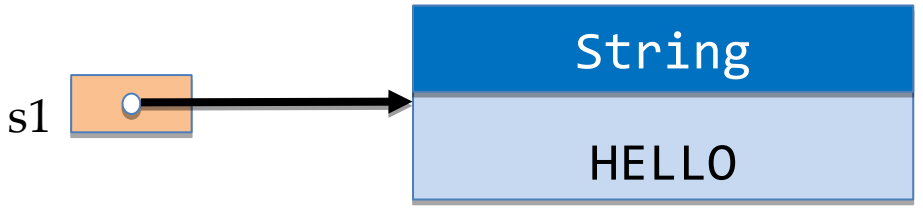
```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

shorthand for `new String`

- what will this method do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String



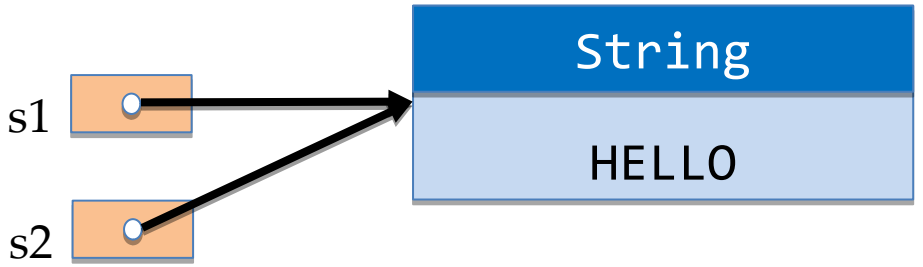

```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

shorthand for `new String`

- what will this method do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String



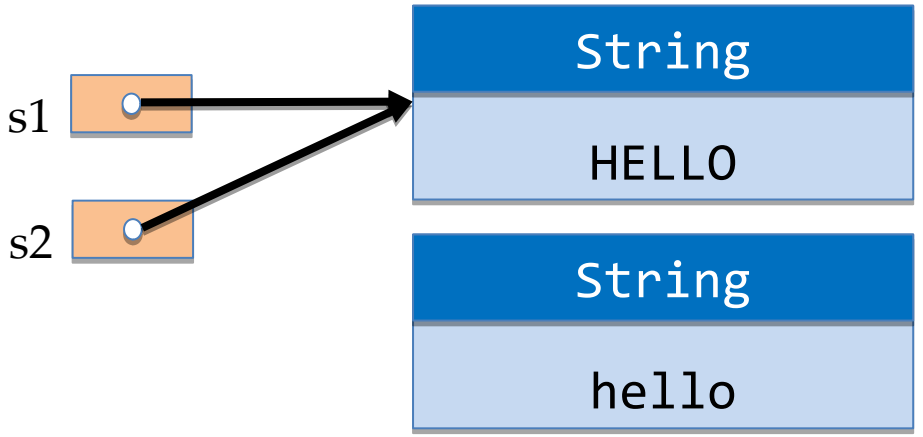

```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

shorthand for `new String`

- what will this method do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String



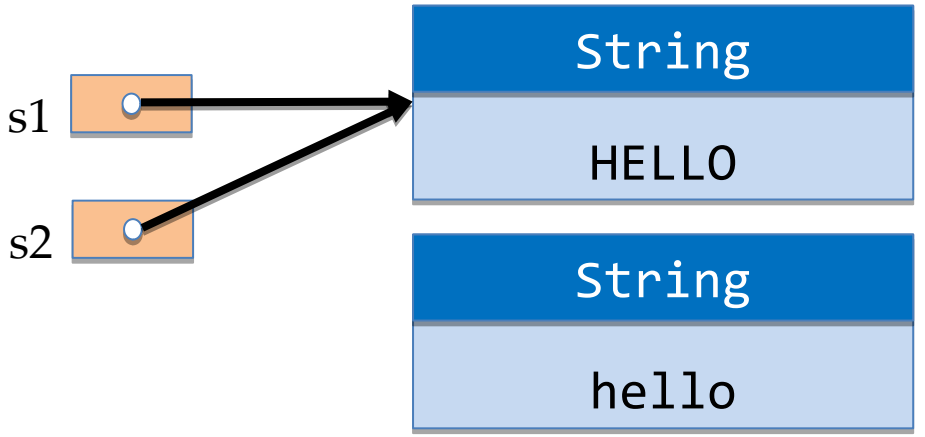

```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

shorthand for `new String`

- what will this method do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String




```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

shorthand for `new String`

- what will this method do?

A. A compiler error

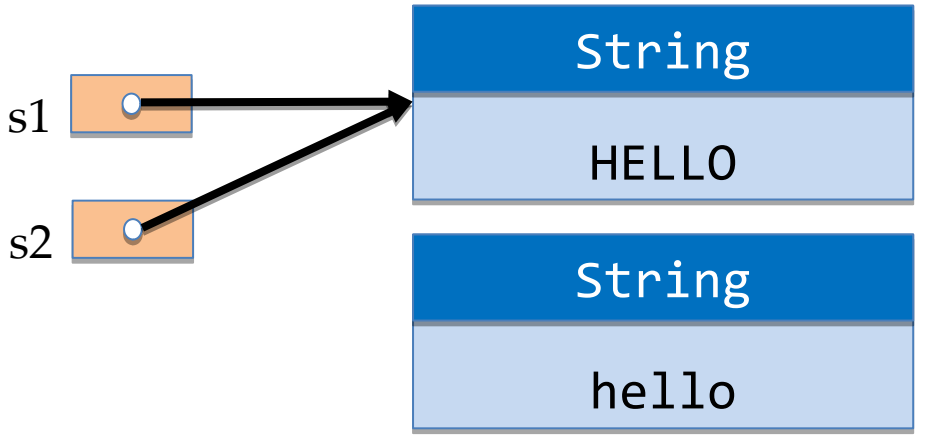
B. print `HELLO, HELLO, world!`

C. print `hello, hello, world!`

D. print `HELLO, hello, world!`

E. something else

String




```
private void stringDemo() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

shorthand for `new String`

- what will this method do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

strings are immutable objects

String

```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

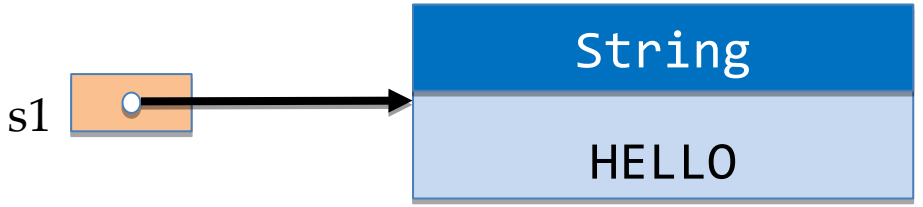
String

```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String

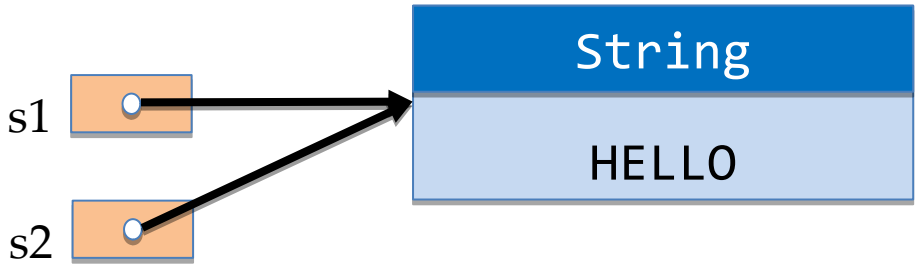



```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String

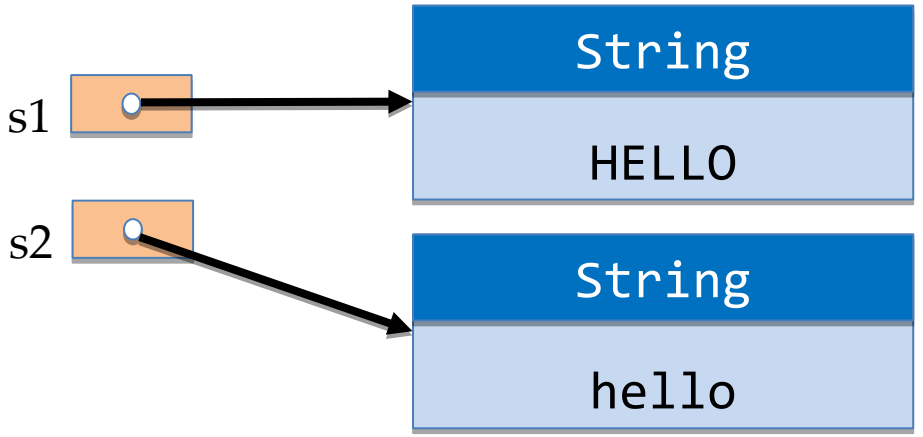



```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String

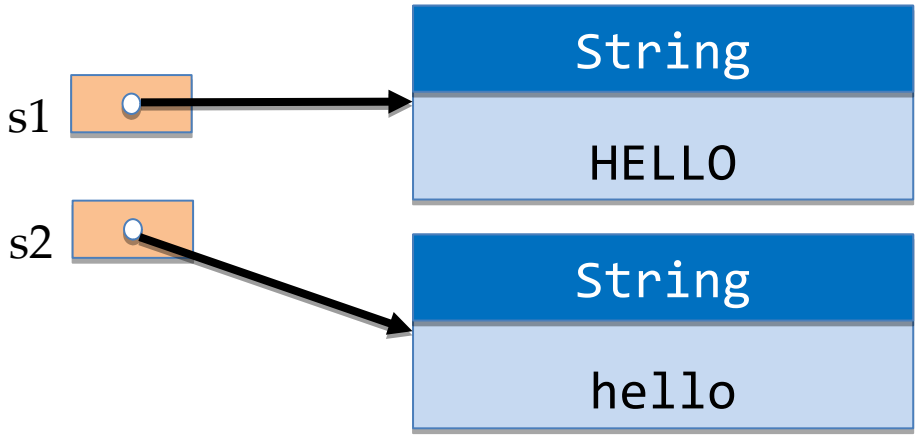



```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String

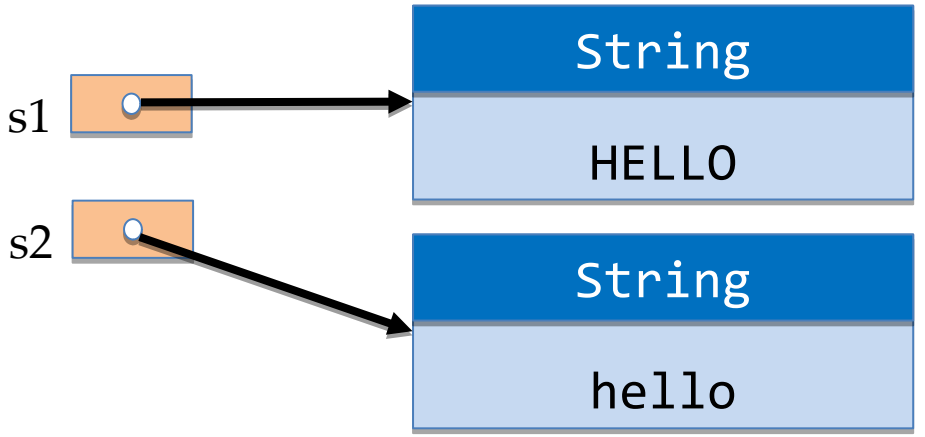



```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

String




```
private void stringDemo1() {
  String s1 = new String("HELLO");
  String s2 = s1;
  s2 = s2.toLowerCase();
  String s3 = "world!";
  System.out.println(s1 + ", " + s2 + ", " + s3);
}
```

- what will this method this do?

A. A compiler error
B. print `HELLO, HELLO, world!`
C. print `hello, hello, world!`
D. print `HELLO, hello, world!`
E. something else

to change 'Strings' we need something else

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
```

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
```

make a copy to change

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
  char[] c3 = Arrays.copyOf(c2, c2.length);
```

make a copy to change

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
  char[] c3 = Arrays.copyOf(c2, c2.length);
```

make a copy to change

from Java library

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
  char[] c3 = Arrays.copyOf(c2, c2.length);
  c3[0] = Character.toLowerCase(c3[0]);
  System.out.println(c3);
}
```

make a copy to change

from Java library

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
  char[] c3 = Arrays.copyOf(c2, c2.length);
  c3[0] = Character.toLowerCase(c3[0]);
  System.out.println(c3);
}
```

make a copy to change

from Java library

hi

char arrays

```
private void arrayDemo2() {
  char[] c1 = {'H', 'i'};
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1) {
    c2[i] = c1[i];
  }
  char[] c3 = Arrays.copyOf(c2, c2.length);
  c3[0] = Character.toLowerCase(c3[0]);
  System.out.println(c3);
}
```

hi

make a copy to change

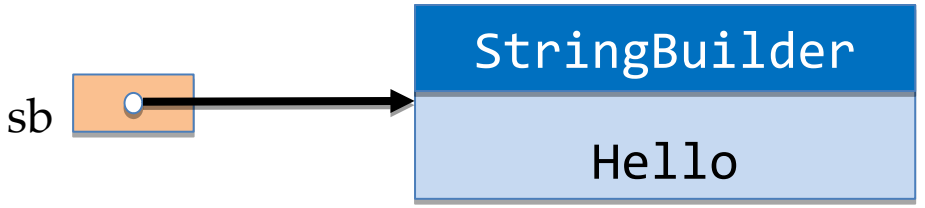
from Java library

clumsy low level, not the preferred solution

StringBuilder

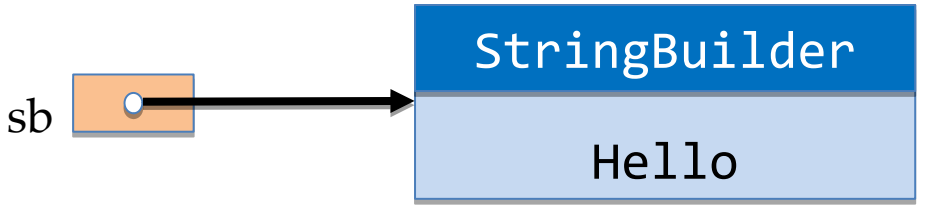
```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
```

StringBuilder




```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
```

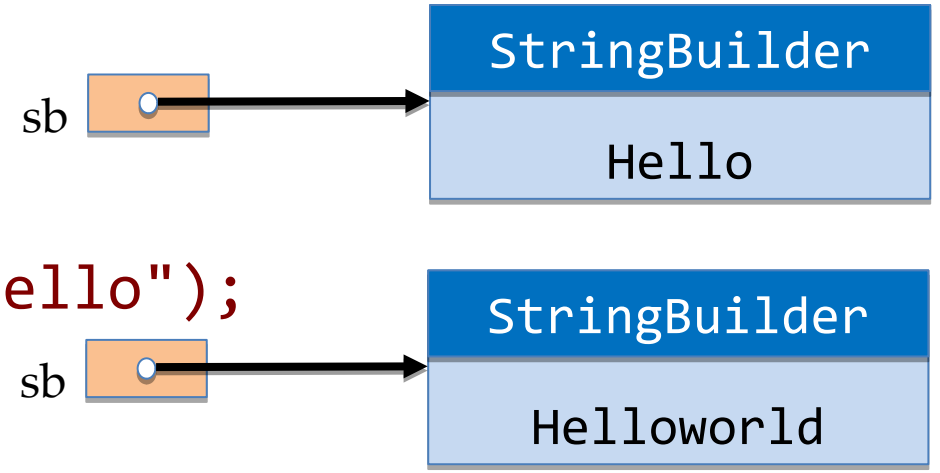
StringBuilder



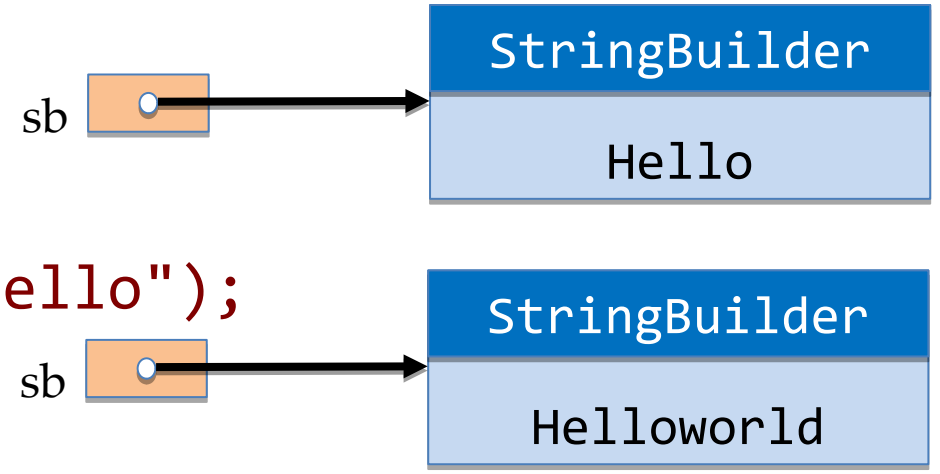

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
```

StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
```



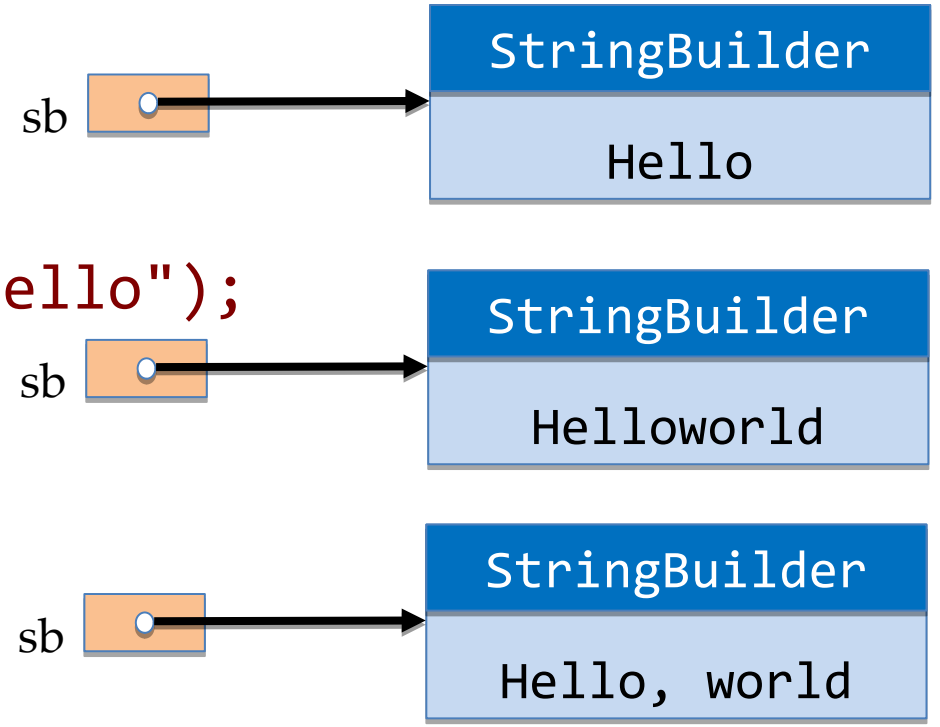
StringBuilder




```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
```

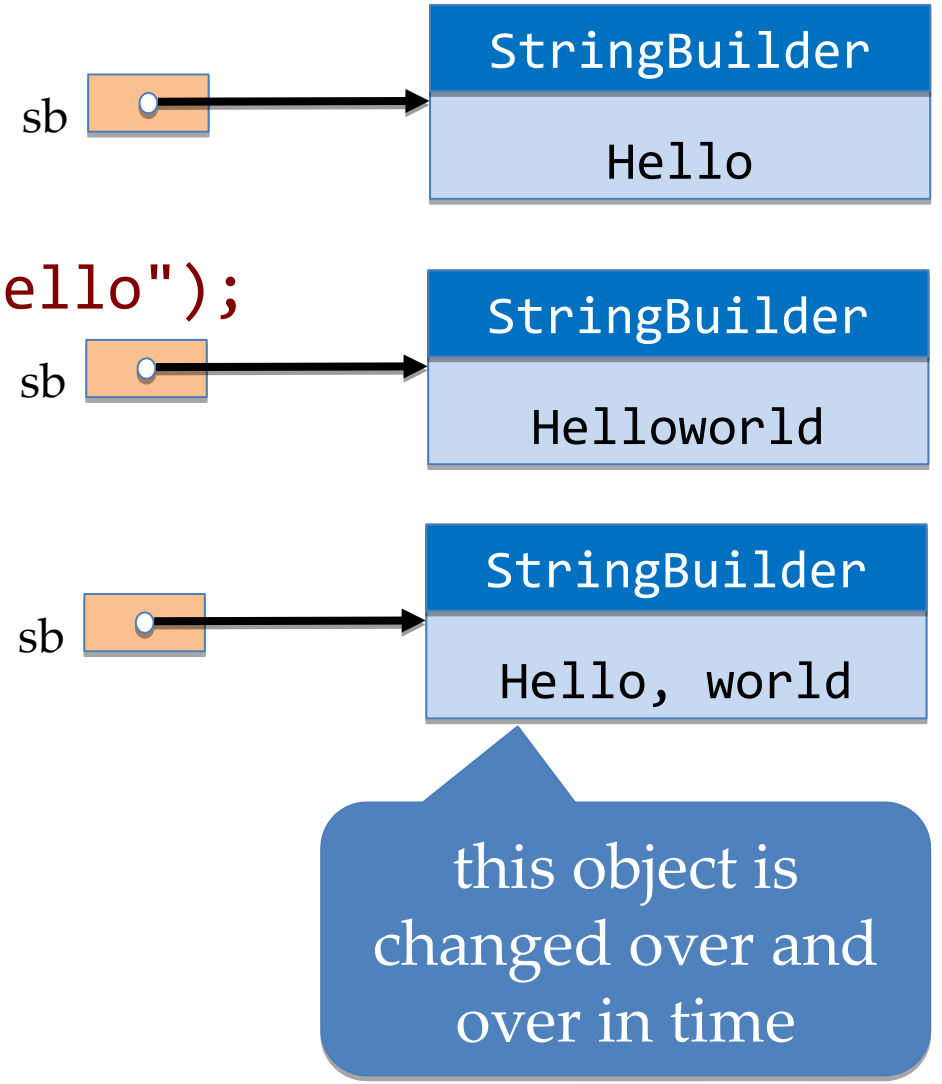
StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
```



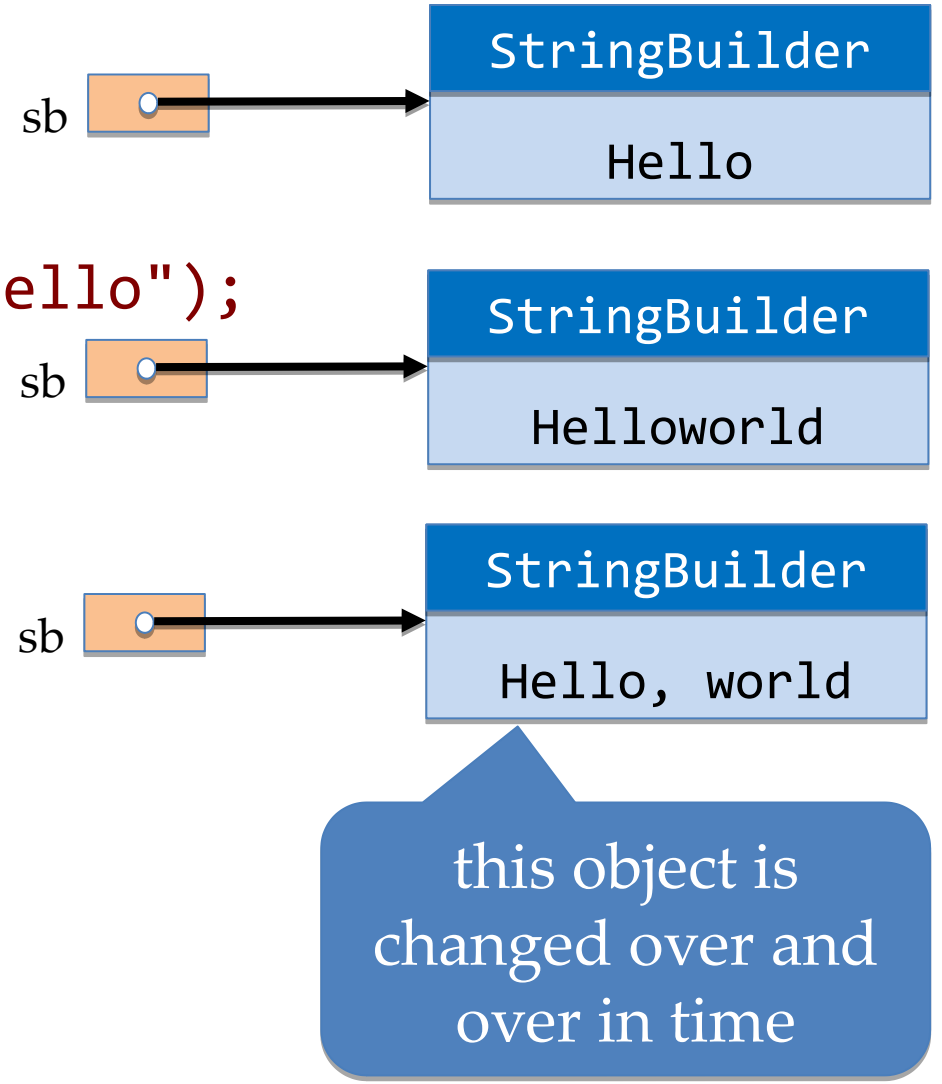
StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
```



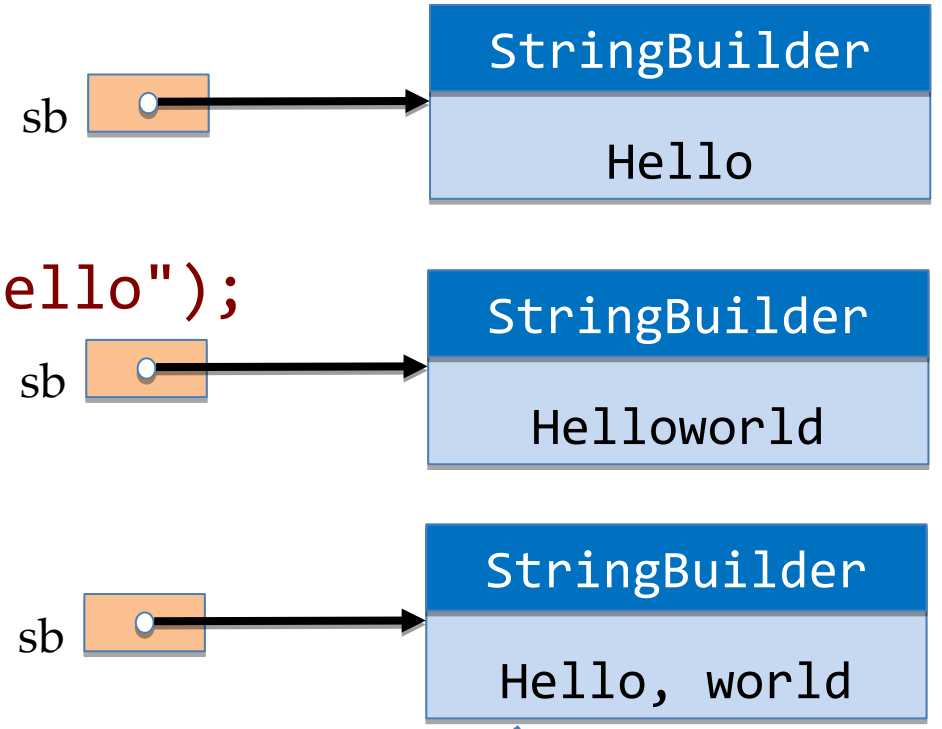
StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
  System.out.println(sb);
}
```



StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
  System.out.println(sb);
}
```

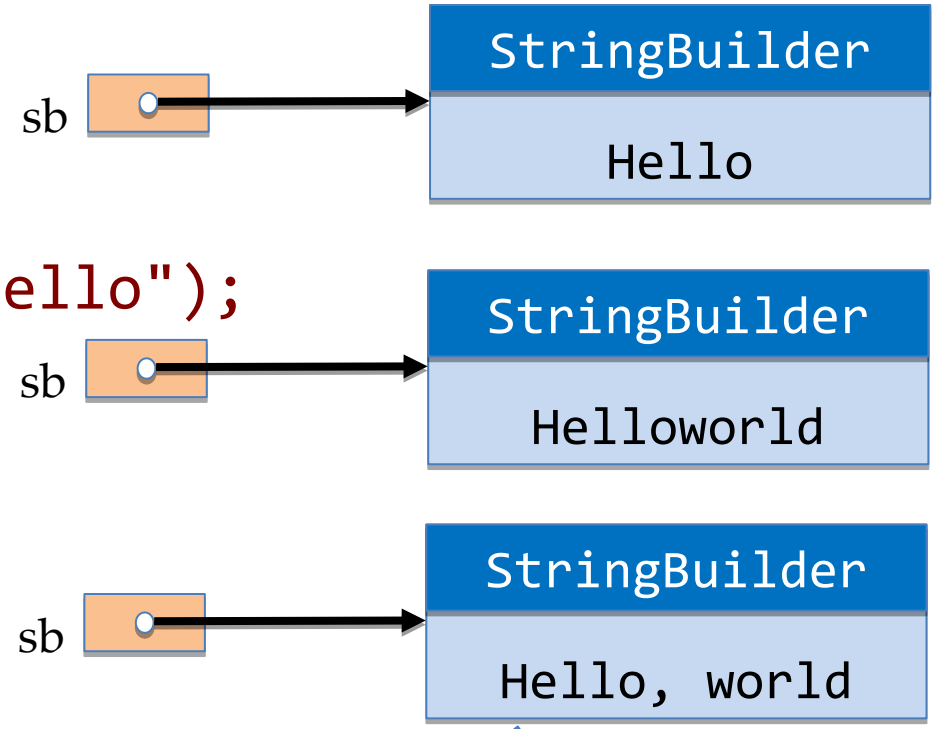


here `toString()` is added automatically by Java

this object is changed over and over in time

StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
  System.out.println(sb);
}
```



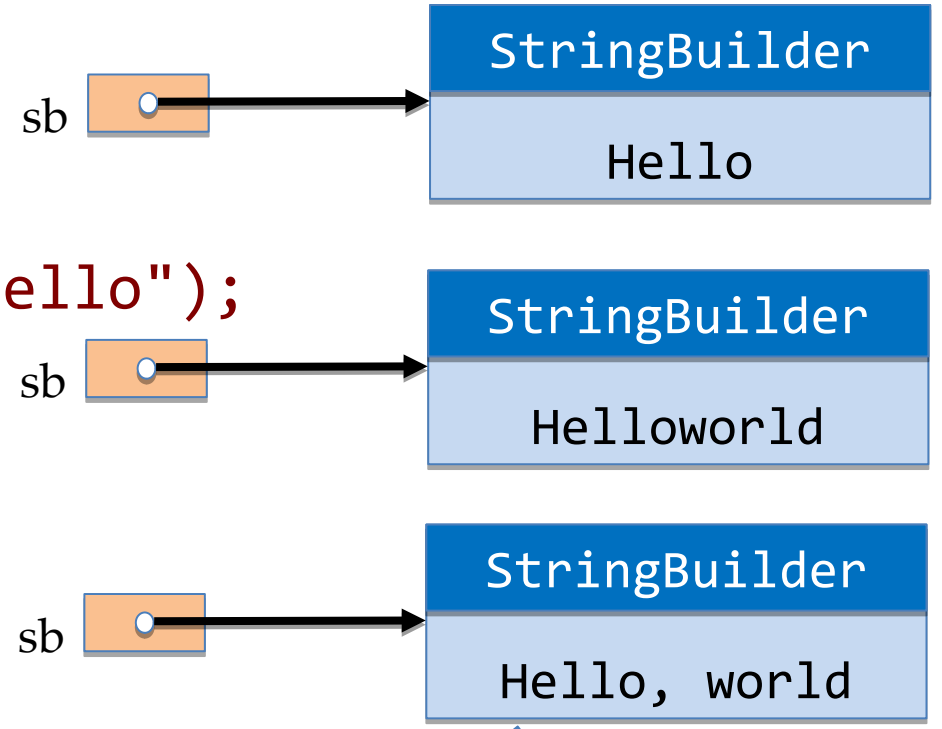
here `toString()` is added automatically by Java

this object is changed over and over in time

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

StringBuilder

```
private void stringBuilderDemo() {
  StringBuilder sb = new StringBuilder("Hello");
  sb.append("world");
  sb.insert(sb.indexOf("w"), ", ");
  System.out.println(sb.toString());
  System.out.println(sb);
}
```



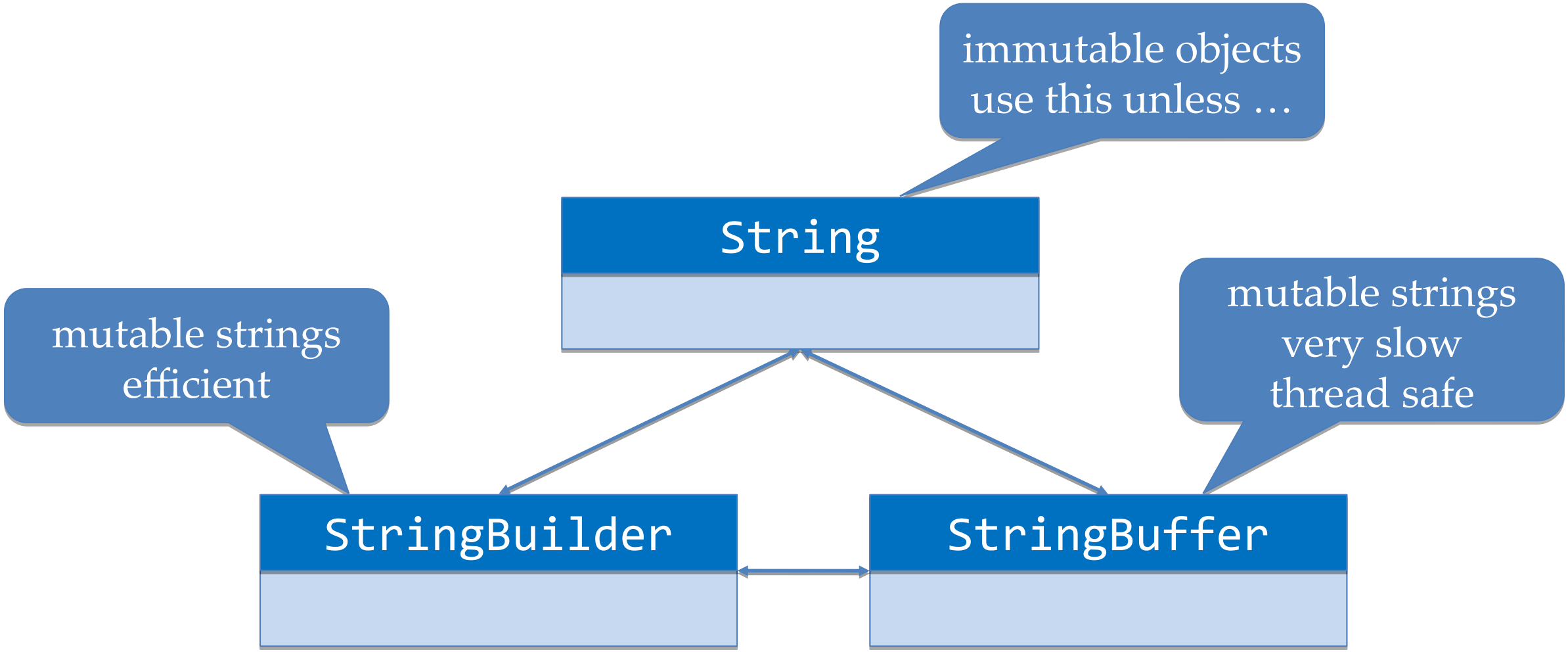
here `toString()` is added automatically by Java

the preferred way to handle changing strings

this object is changed over and over in time

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

String - StringBuilder – StringBuffer



some useful StringBuilder methods

```
StringBuilder()
StringBuilder(int capacity)
StringBuilder(String str)

append(int i)                          // similar methods for other Java types
insert(int offset, char c)             // similar methods for other Java types
setCharAt(int index, char ch)
char charAt(int index)
int indexOf(String str)                          // also for char
int indexOf(String str, int fromIndex) // -1 if char does not occur
```

complete API at:

docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

reading input

SCANNING

Scanner

```
private void scan1() {
  Scanner scan = new Scanner(" Hello, world! ");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

Scanner

instead of `System.in`

```
private void scan1() {
  Scanner scan = new Scanner(" Hello, world! ");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

Scanner

instead of `System.in`

```
private void scan1() {
  Scanner scan = new Scanner(" Hello, world! ");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- what will this method do?

A. Compiler error
B. Runtime error
C. print `Token: "Hello, world!"`
D. print `Token: "Hello," Token: "world!"`
E. print `Token: "Hello" Token: "world"`
F. something else

Scanner

instead of `System.in`

```
private void scan1() {
  Scanner scan = new Scanner(" Hello, world! ");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- what will this method do?

A. Compiler error
B. Runtime error
C. print `Token: "Hello, world!"`
D. print `Token: "Hello," Token: "world!"`
E. print `Token: "Hello" Token: "world"`
F. something else

```
Token: "Hello,"
Token: "world!"
```

Scanner

instead of `System.in`

```
private void scan1() {
  Scanner scan = new Scanner(" Hello, world! ");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- what will this method do?

A. Compiler error
B. Runtime error
C. print `Token: "Hello, world!"`
D. print `Token: "Hello," Token: "world!"`
E. print `Token: "Hello" Token: "world"`
F. something else

```
Token: "Hello,"
Token: "world!"
```

how to get rid of non-alpha's ?

scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- pattern: `"[^a-zA-Z]+"`

scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- pattern: `"[^a-zA-Z]+"`

[ ] group

scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- pattern: `"[^a-zA-Z]+"`

[ ] group

^ not

scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- pattern: `"[^a-zA-Z]+"`

[ ] group

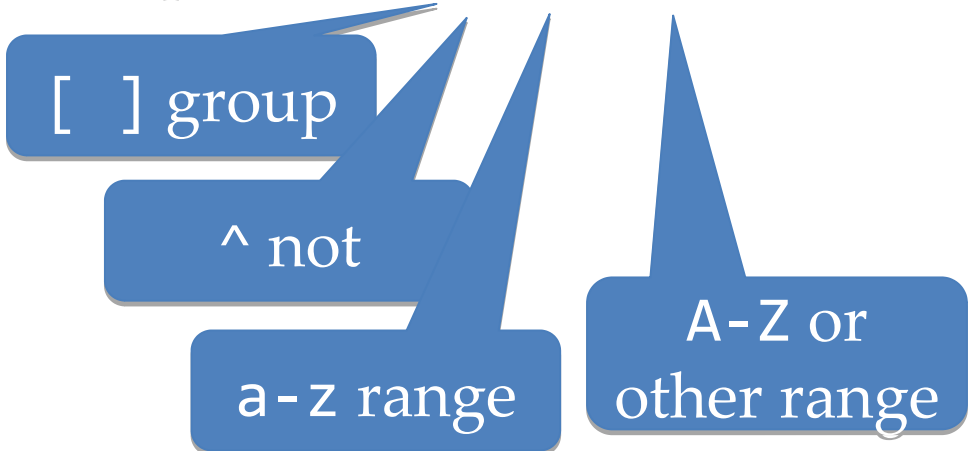
^ not

a-z range

scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

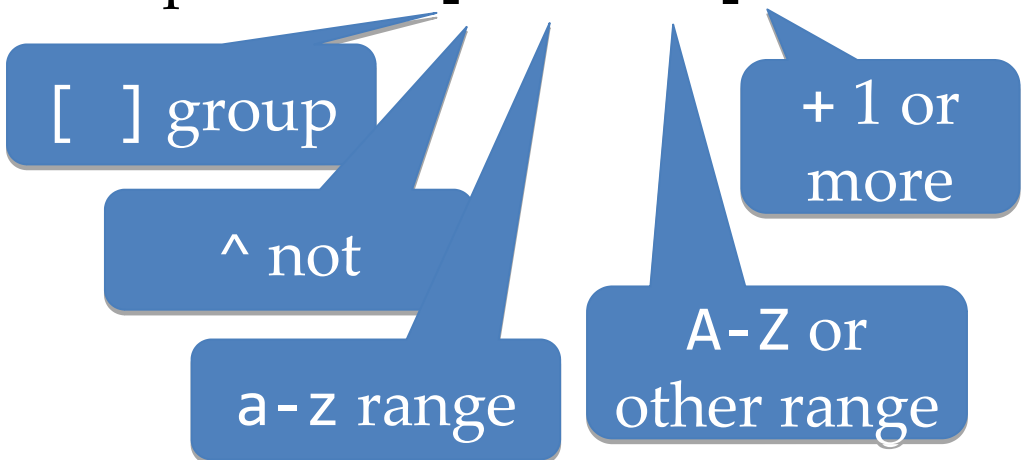
- pattern: `"[^a-zA-Z]+"`



scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

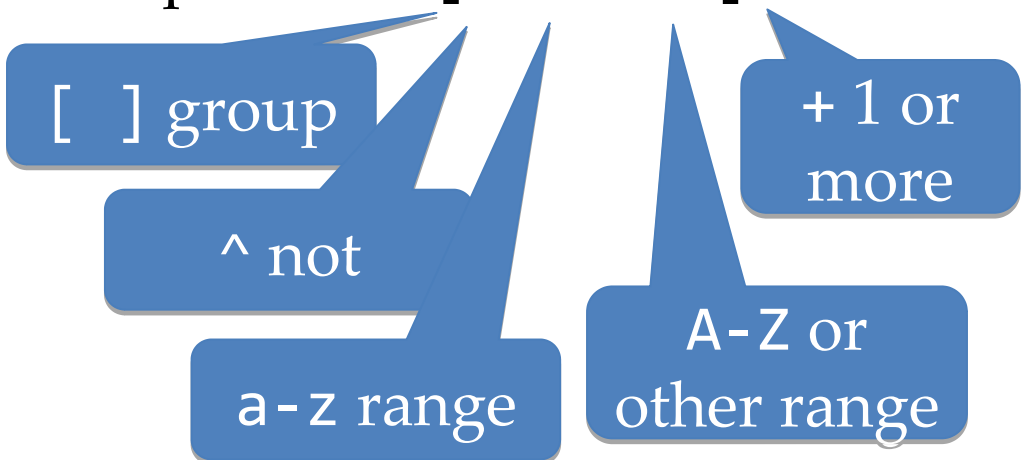
- pattern: `"[^a-zA-Z]+"`



scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- pattern: `"[^a-zA-Z]+"`



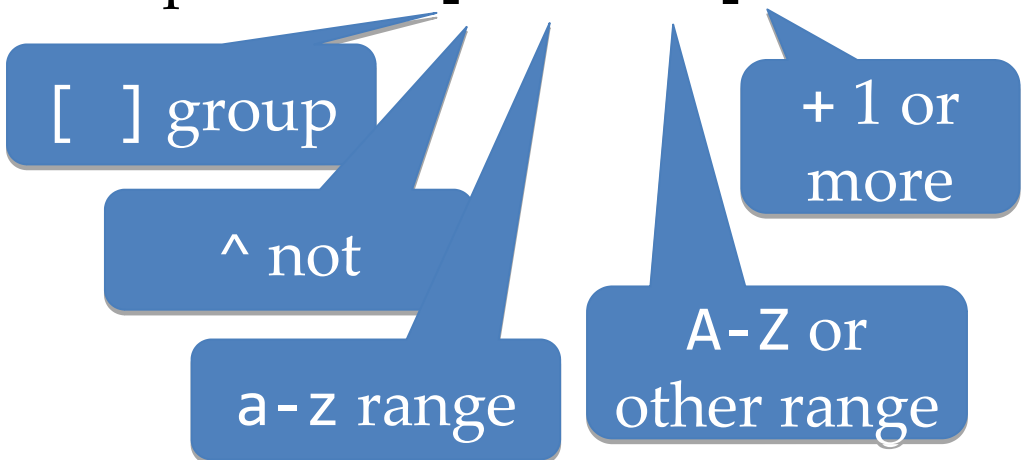
- what will this method do?

A. Compiler error
B. print `Token: "Hello, world"`
C. print `Token: "Hello," Token: "world!"`
D. print `Token: "Hello" Token: "world"`
E. something else

scanner with tailor-made delimiters

```
private void scan2() {
  Scanner scan = new Scanner("Hello, world! ");
  scan.useDelimiter("[^a-zA-Z]+");
  while (scan.hasNext()) {
    System.out.println("Token: \"" + scan.next() + "\"");
  };
}
```

- pattern: `"[^a-zA-Z]+"`



- what will this method do?

A. Compiler error
B. print `Token: "Hello, world"`
C. print `Token: "Hello," Token: "world!"`
D. print `Token: "Hello" Token: "world"`
E. something else

```
Token: "Hello"
Token: "world"
```

patterns

- patterns are regular expressions
  - given by string
  - or `Pattern` object

docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

- apart from setting the delimiter we can use
  - `hasNext("\\p(Digit)")`
  - `next("\\d+")`
  - `nextInt()`
  - etc.

Delimiter = \p{javaWhitespace}+

```
private void delimiter() {
  Scanner scan = new Scanner("");
  System.out.println("Delimiter = " + scan.delimiter());
}
```

reading numbers

```
private void scan3() {
  Scanner scan = new Scanner(" 42 is the answer.");
  int a = scan.nextInt();
  System.out.println("a = " + a);
}
```

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

reading numbers

```
private void scan3() {
  Scanner scan = new Scanner(" 42 is the answer.");
  int a = scan.nextInt();
  System.out.println("a = " + a);
}
```

- what will this method do?

A. Compiler error
B. Runtime error
C. print `a = 0`
D. print `a = 4`
E. print `a = 42`

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

reading numbers

```
private void scan3() {
  Scanner scan = new Scanner(" 42 is the answer.");
  int a = scan.nextInt();
  System.out.println("a = " + a);
}
```

- what will this method do?

```
a = 42
```

A. Compiler error
B. Runtime error
C. print `a = 0`
D. print `a = 4`
E. print `a = 42`

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

reading numbers 2

```
private void scan4() {
  Scanner scan = new Scanner("The answer is 42.");
  int a = scan.nextInt();
  System.out.println("a = " + a);
}
```

reading numbers 2

```
private void scan4() {
  Scanner scan = new Scanner("The answer is 42.");
  int a = scan.nextInt();
  System.out.println("a = " + a);
}
```

- what will this method do?

A. Compiler error
B. Runtime error
C. print `a = 0`
D. print `a = 4`
E. print `a = 42`

reading numbers 2

```
private void scan4() {
  Scanner scan = new Scanner("The answer is 42.");
  int a = scan.nextInt();
  System.out.println("a = " + a);
}
```

- what will this method do?

A. Compiler error
B. Runtime error
C. print `a = 0`
D. print `a = 4`
E. print `a = 42`

```
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at ootutorial02.OOtutorial02.scan4(OOtutorial02.java:88)
    at ootutorial02.OOtutorial02.main(OOtutorial02.java:30)
```

Exceptions

- Exceptions indicate a runtime problem in a Java program
  - also used in many other languages
- Exceptions are more flexible than stopping the program with an error code
  - we can catch the exception and do something to handle the error

- we will discuss this in detail later in this course
  - for now it is good to know what exceptions are
  - it is good to be able to read the stack trace
  - you should use them now to ensure you don't crash on malformatted input!

```
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at ootutorial02.OOtutorial02.scan4(OOtutorial02.java:88)
    at ootutorial02.OOtutorial02.main(OOtutorial02.java:30)
```

catching the nextInt exception

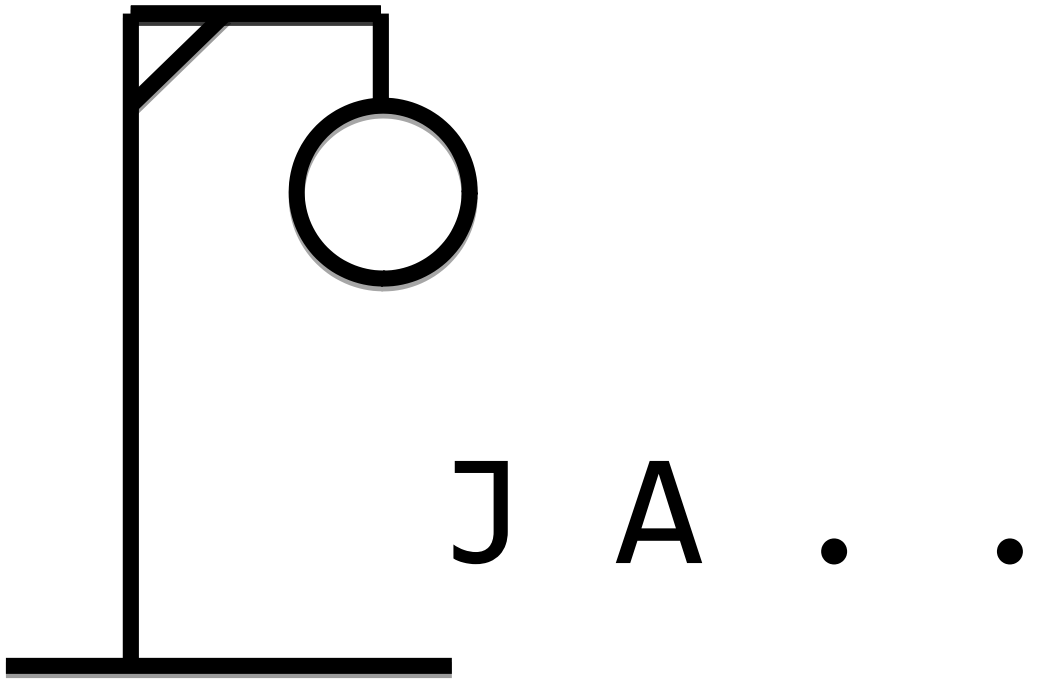
```
private void scan5() {
  Scanner scan = new Scanner("The answer is 42.");
  int a = safeNextInt(scan);
  System.out.println("a = " + a);
}

private static int safeNextInt(Scanner scan) {
  int result = 0;
  try {
    result = scan.nextInt();
  } catch (InputMismatchException e) {
    System.out.print("O dear ");
    e.printStackTrace(System.out);
  }
  return result;
}
```

catch only used if the exception occurs

always

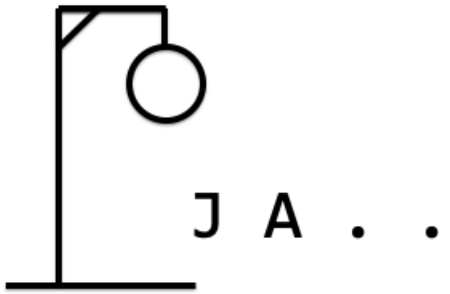
```
O dear java.util.InputMismatchExcep
    at java.util.Scanner.throwFor(S
    at java.util.Scanner.next(Scann
    at java.util.Scanner.nextInt(Sc
    at java.util.Scanner.nextInt(Sc
    at ootutorial02.OOtutorial02.sa
    at ootutorial02.OOtutorial02.sc
    at ootutorial02.OOtutorial02.ma
a = 0
```



assignment

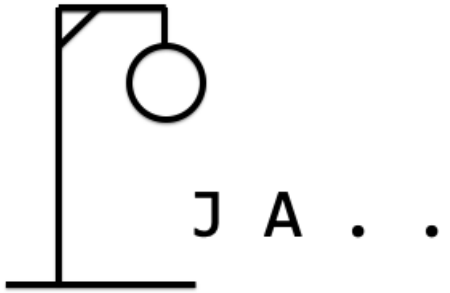
HANGMAN

Model? View?



- Separation of concerns
  - keeping track of the state of the program (Model)
  - presenting that state to the user (View)
  - acting on user actions (Controller, ViewModel, ...)
- Model
  - models the problem being solved
  - includes logic to manipulate data in the model
- View
  - separate class(es) to present the model to the user
- Don't worry about cleanly separating Control from View just yet
  - but you could, e.g., have a separate class handling the input gotten on System.in from the class handling output on System.out

hints



- OO design
  - classes needed (important nouns in assignment)
  - attributes (properties of these objects)
  - methods (actions of the objects; verbs)
- what do we need for the administration (state, model)
  - there might be a reason to introduce `stringBuilder`
- all I/O in separate class (view)
  - use a scanner
- there is a given class `WordReader` to read words from a file
  - you are most welcome to look at the implementation
  - the reuse idea of OO is that knowing the interface must be enough to use it

define a class even if your program needs only one instance

do as we say, not as we do

- all I/O in separate class (view)
  - sometimes we show code that breaks our own rules
  - e.g. all logic in main,
  - I/O in Model...

```
public static void mainPrim() {

   int i = 10;
   int j = 20;


   System.out.println( i );
   System.out.println( j );


   j = i;
   i = 50;


   System.out.println( i );
   System.out.println( j );

}
```

```
public class CreditCard {
    private final String name, date;
    private final int number, cvv;

    public CreditCard( String name, String date, int number, int cvv ) {
        this.name   = name;
        this.date   = date;
        this.number = number;
        this.cvv    = cvv;
    }

    public void pay( double amount ) {
        System.out.format( "Paid %1.2f with card %d of %s\n", amount, numb
    }

}
```

these are demonstrations!

- Used to illustrate what we teach
- Impossible to show the “correct” way without losing focus on what matters
- Large class hierarchies tend to not fit on slides
- Implementing `CreditCard.pay(double)` not relevant to teaching goal!

- You do not have these liberties

printf debugging

- Sometimes need to inspect state of program while running
- Dirty option: `System.out.println(state.toString());`
  - Okay while developing, but take them out before submitting!
- Better option: `System.err.println(state.toString());`
  - Processes have three default I/O "pipes":
    1) Standard Input (stdin): `System.in`
    2) Standard Output (stdout): `System.out`
    3) Standard Error (stderr): `System.err`
  - Can "redirect" these pipes to/from other processes, files, network sockets…
- Best option: use a debugger!
  - debug → debug project / ctrl+F5
  - breakpoints (left-click on line number) pause execution when the line is reached
  - stepping allows you to execute line-by-line

System.err output

- Sensible error output on System.err can be left in submission
  - E.g. printing the trace of unexpected / uncommon Exceptions ("Exception caught: ...")
    - (but not the trace of an InputMismatchException when a user enters a non-integer…)
- But not:
  - "Entered method X"
  - "Student created"
  - "Student name changed"
  - "s == 42"
  - …
- If in doubt, take it out before submitting

finally

- `while`, `for`, `if` etc. allow you to omit `{ }` if the body is only a single line

```
private char[] charCopy(char[] c1) {
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1)
    c2[i] = c1[i];
  return c2;
}
```

NEVER DO THIS!

why not?

- Future you will hate your guts:

```
private char[] charCopyUntil(char[] c1, char stp) {
  char[] c2 = new char[c1.length];
  for (int i = 0; i < c1.length; i += 1)
    c2[i] = c1[i];
    if (c2[i] == stp) {
      return c2;
    }
  return c2;
}
```

designing and building assignment 1

IDE DEMO – NARRATED CODING