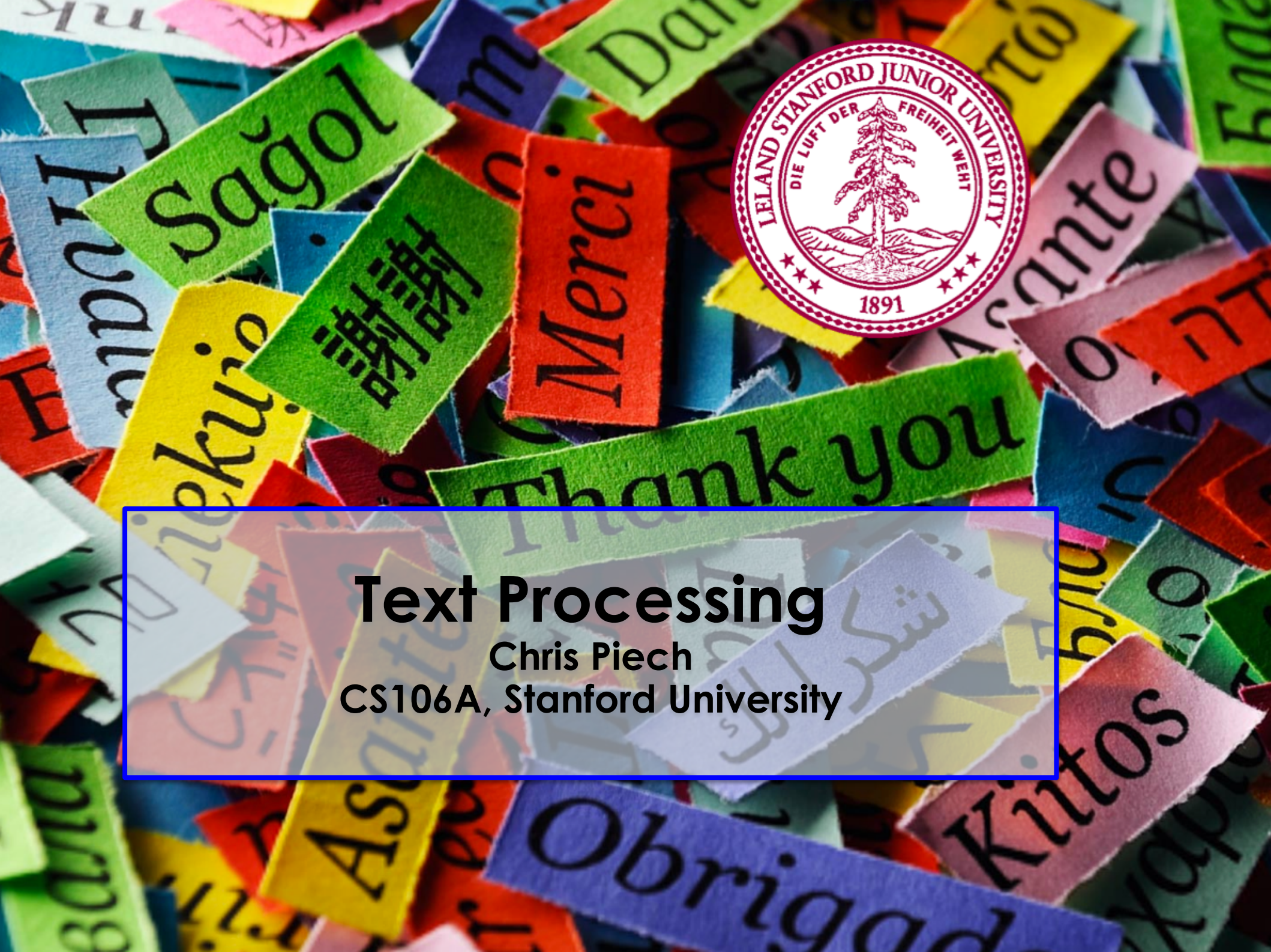


Text Processing

Chris Piech
CS106A, Stanford University

<Review>

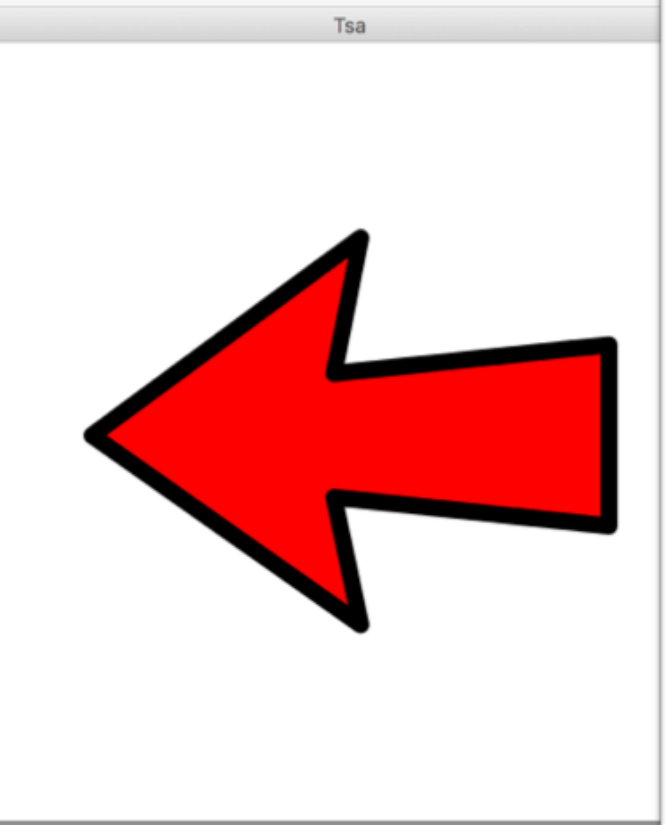
What does an object store?

Objects store addresses
(which are like URLs)

Primitives vs Objects

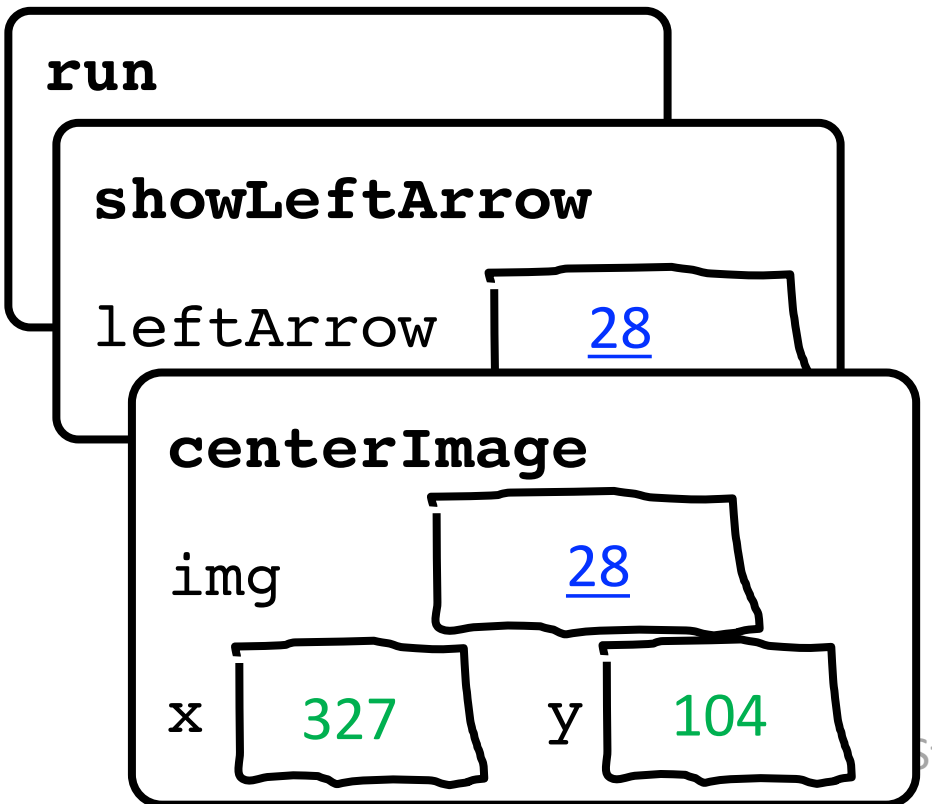
- **Primitives** store their *actual value* in their variable box. You can compare values with == and !=, and the original does not change when passed as a parameter and changed.
- **Objects** store their *address* in their variable box. You can't compare properties of an object via ==, and the original *does* change when passed as a parameter and changed.
- **Primitives,** when parameters, are *passed by value*, **Objects** are *passed by their address.*

Revisit our Programs

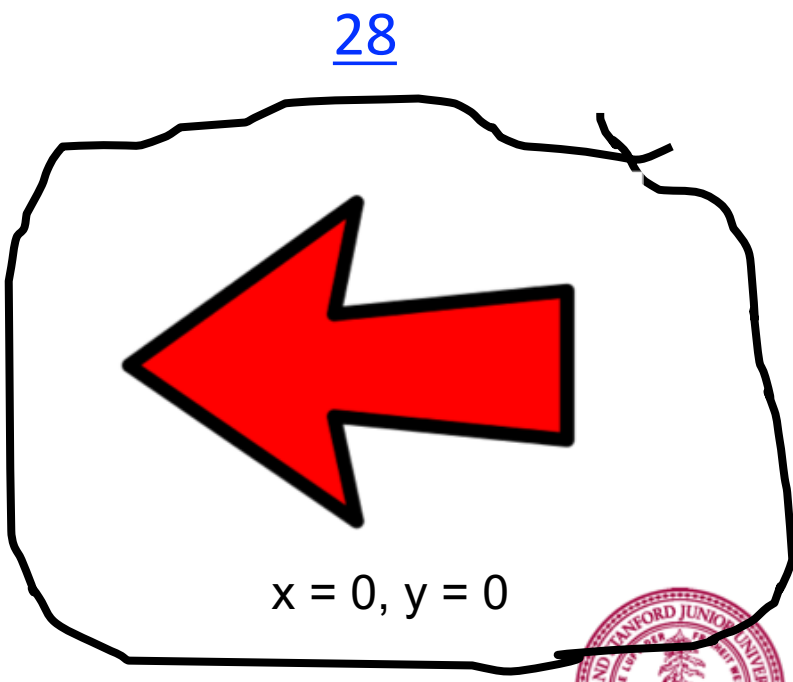



```
// displays a centered right arrow img
private void showRightArrow() {
    GImage rightArrow = new GImage("leftArrow.png");
    add(rightArrow);
    centerImage(rightArrow);
}

// displays a centered left arrow img
private void showLeftArrow() {
    GImage leftArrow = new GImage("rightArrow.png");
    add(leftArrow);
    centerImage(leftArrow);
}

// takes an image and adds it to the screen
private void centerImage(GImage img) {
    double w = img.getWidth();
    double h = img.getHeight();
    double x = (getWidth() - w) / 2;
    double y = (getHeight() - h) / 2;
    img.setLocation(x, y);
}
```

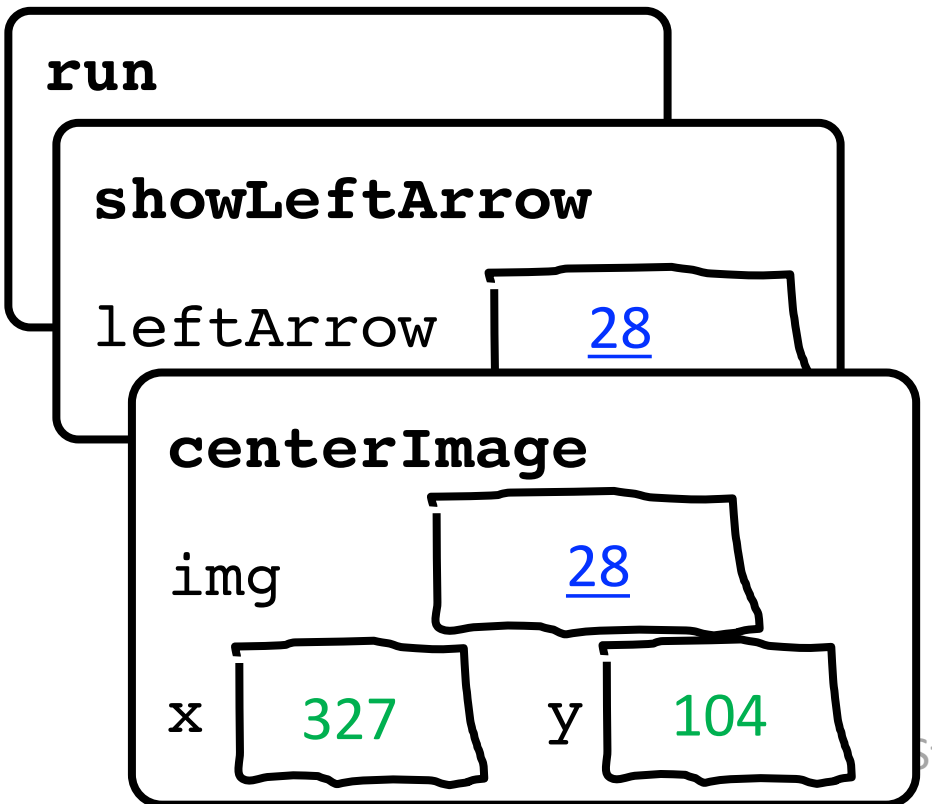
```
private void showLeftArrow() {
    GImage leftArrow = new GImage("rightArrow.png");
    add(leftArrow);
    centerImage(leftArrow);
}

// takes an image and adds it to the screen
private void centerImage(GImage img) {
    ...
    img.setLocation(x, y);
}
```



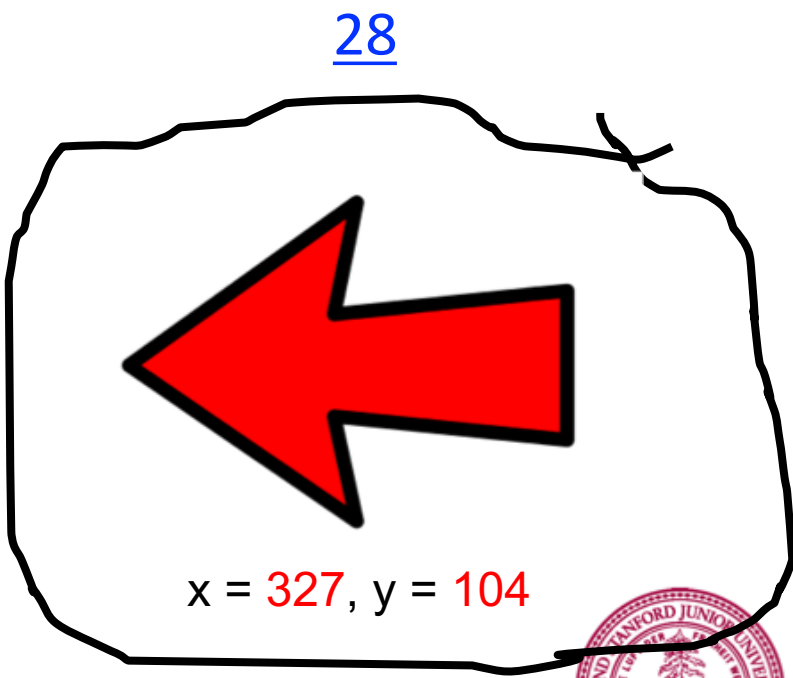
heap



```
private void showLeftArrow() {
    GImage leftArrow = new GImage("rightArrow.png");
    add(leftArrow);
    centerImage(leftArrow);
}

// takes an image and adds it to the screen
private void centerImage(GImage img) {
    ...
    img.setLocation(x, y);
}
```



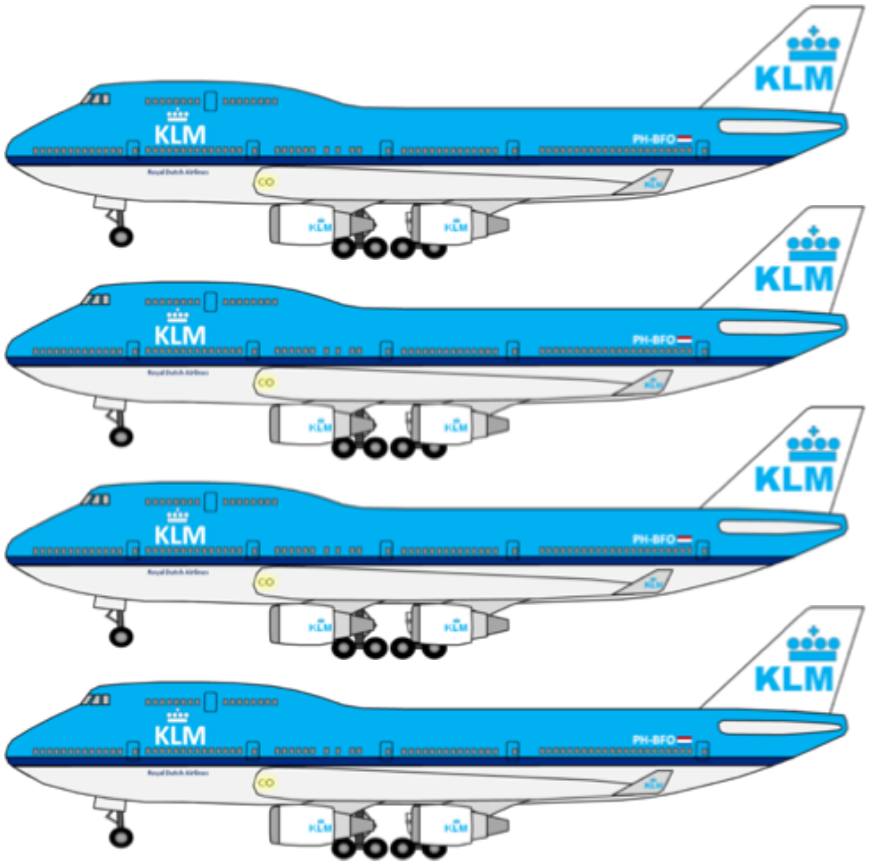
heap



</Review>

Fake Medicine is a Problem

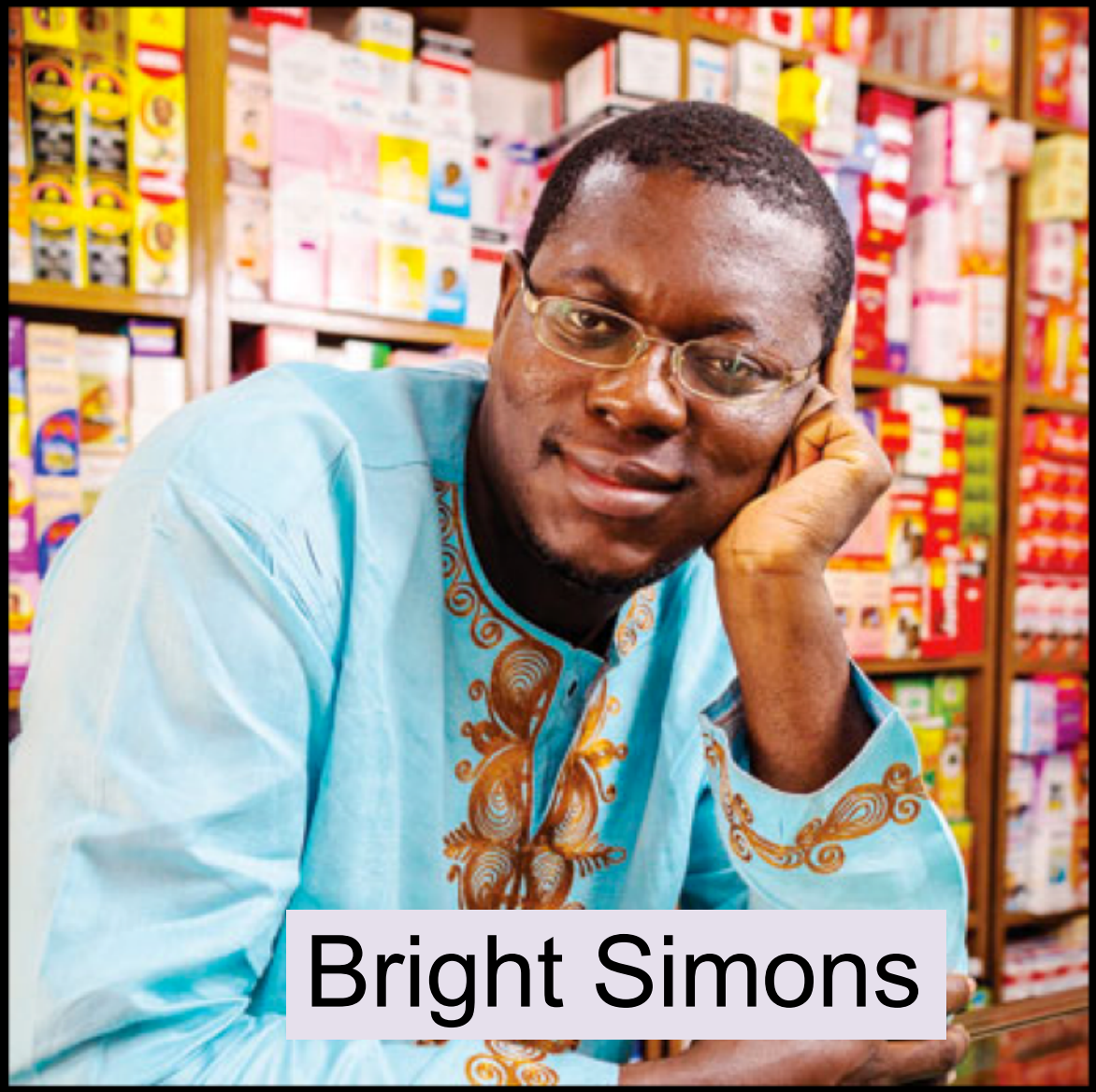
700,000 deaths a year from fake malaria and tuberculosis drugs [1]



Equivalent of this many crashes per day

[1] http://www.un.org/africarenewal/magazine/may-2013/counterfeit-drugs-raise-africa%E2%80%99s-temperature

Chris' Favorite Program



Underlying Puzzle

Counterfeiter

User

You (Distributor)

Underlying Puzzle

Counterfeiter

User

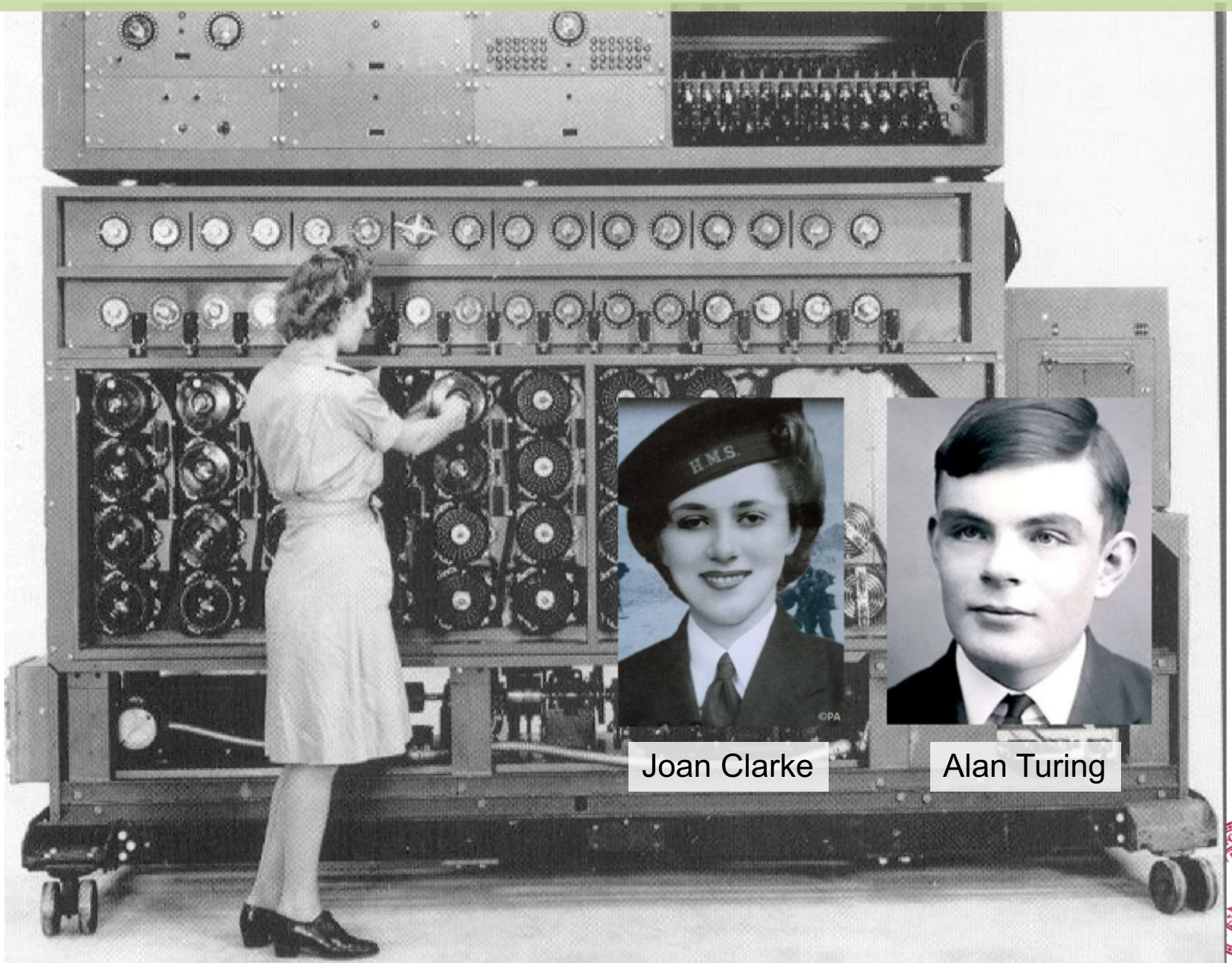
You (Distributor)

Revisit this on Wednesday

Learning Goals

1. Be able to perform math operations on chars
2. Be able to write string algorithms that loops over each character

Text Problem: Decryption



Text Problem: Translation

The spirit is willing but the flesh is weak.

↓

(Russian)

↓

The vodka is good but the meat is rotten.

**This result cost billions of dollars (adjusted for inflation)*

How is text
represented?

The variable type **`String`**

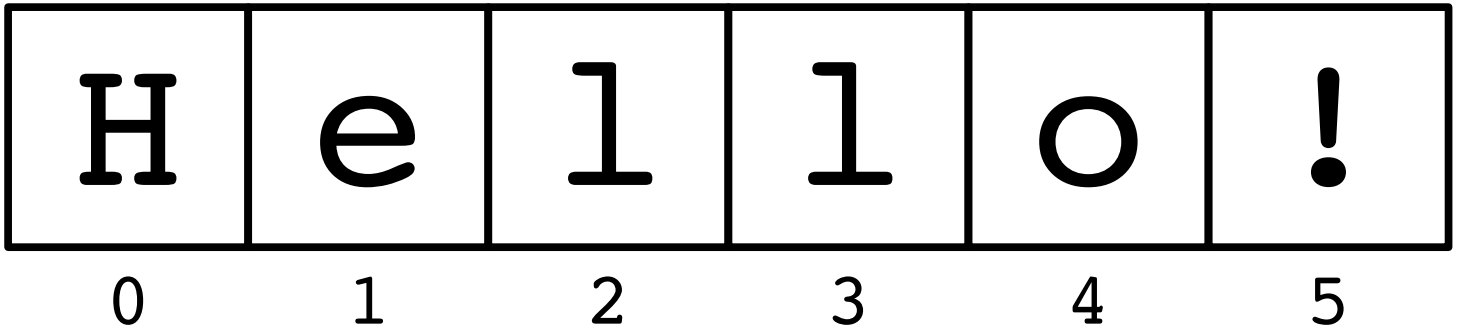
Text is stored using the variable type **`String`**.

A **`String`** is a sequence of characters.

```
public void run() {
    String text = "hello!";
    println(text);
}
```

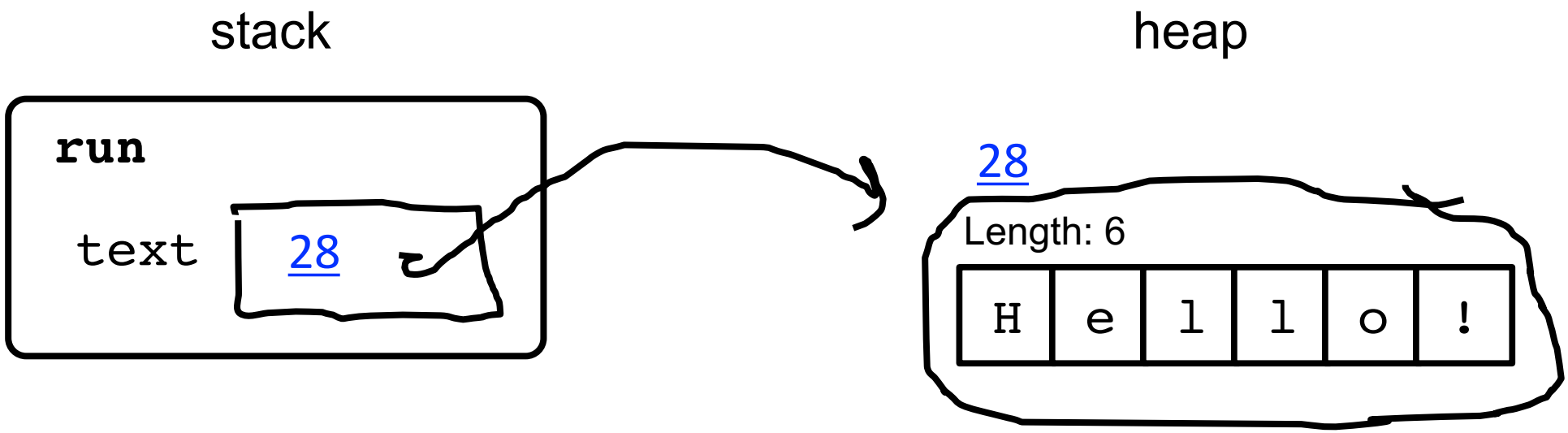

H e l l o !

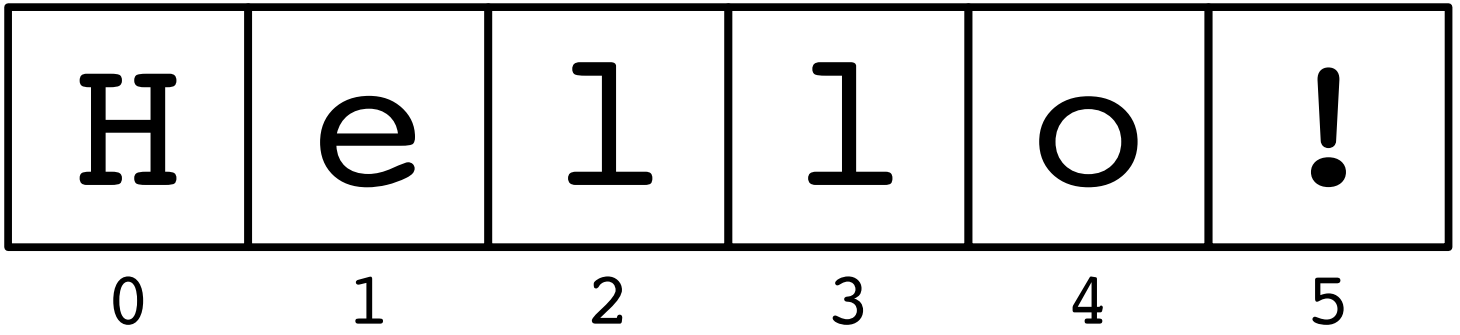

LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891


H
e
l
l
o
!
0
1
2
3
4
5

How it is actually stored

```
public void run() {
    String text = "hello!";
}
```





***text*.charAt(*index*)**

All characters in a string have an *index*.

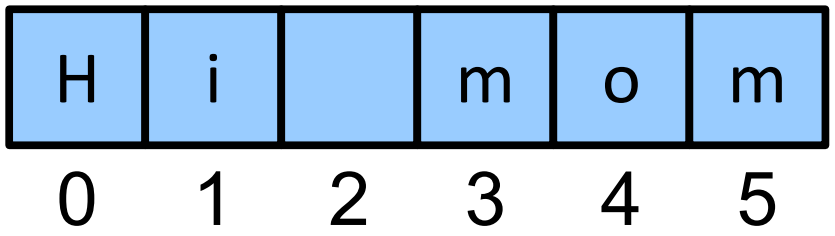
You can access a character in the sequence via its index

Looping with Strings

- Because each character in a string occupies a particular index, you can visit all the characters in a string one at a time using a loop.
- Canonical "loop over the characters in a string" loop:

```
for (int i = 0; i < string.length(); i++) {
    char ch = string.charAt(i);
    /* … process ch … */
}
```

- The *string*`.length()` method returns the number of characters in the string. This is one larger than the last valid index in the string.



(length 6)

numLs

How are characters represented?

The variable type **char**

- The primitive type **char** represents a single character or glyph.
- Some examples:

```
char letterA = 'A';
char plus    = '+'
char zero    = '0';
char space   = ' ';
char newLine = '\n'; // special
char first = text.charAt(0);
```

Enumeration

```
private static final int FROSH = 1;
private static final int SOPHOMORE = 2;
private static final int JUNIOR = 3;
private static final int SENIOR = 4;
private static final int OTHER = 5;

private int askForYear() {
  while (true) {
    int year = readInt("Enter class year: ");
    if (year >= FROSH && year <= OTHER) return year;
  }
}

private void printPopulation() {
   for(int year = FROSH; year <= SENIOR; year++) {
     printYear(year);
   }
}
```


Chars are just a giant enumeration. You can use math operators on char!

ASCII

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

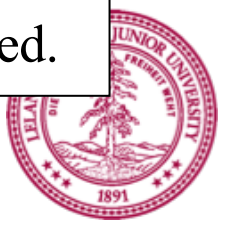
* This is only the first half of the table

The letter A, for example, has the ASCII value 65

‘A’ -> ‘Z’ are sequential.
‘a’ -> ‘z’ are sequential.
‘0’ -> ‘9’ are sequential.

Useful Character methods

`static boolean isDigit(char ch)` Determines if the specified character is a digit.
`static boolean isLetter(char ch)` Determines if the specified character is a letter.
`static boolean isLetterOrDigit(char ch)` Determines if the specified character is a letter or a digit.
`static boolean isLowerCase(char ch)` Determines if the specified character is a lowercase letter.
`static boolean isUpperCase(char ch)` Determines if the specified character is an uppercase letter.
`static boolean isWhitespace(char ch)` Determines if the specified character is **whitespace** (spaces and tabs).
`static char toLowerCase(char ch)` Converts **`ch`** to its lowercase equivalent, if any. If not, **`ch`** is returned unchanged.
`static char toUpperCase(char ch)` Converts **`ch`** to its uppercase equivalent, if any. If not, **`ch`** is returned unchanged.



Useful Character methods

```
public void run() {
    String str = readLine("Line: ");

    char ch = str.charAt(0);
    println("Original first char: " + ch);

    ch = Character.toUpperCase(ch);
    println("Uppercase first char: " + ch);

    if(Character.isLetter(ch)) {
       println("It's a letter!");
    }
}
```


numUppercase

Strings have some unique properties

Strings are often made through concatenation

```
public void run() {
    String s1 = "CS106";
    String s2 = "A";
    String s3 = "I got an " + s2 + " in " + s1 + s2;

    println(s3);
}
```

```
I got an A in CS106A
```

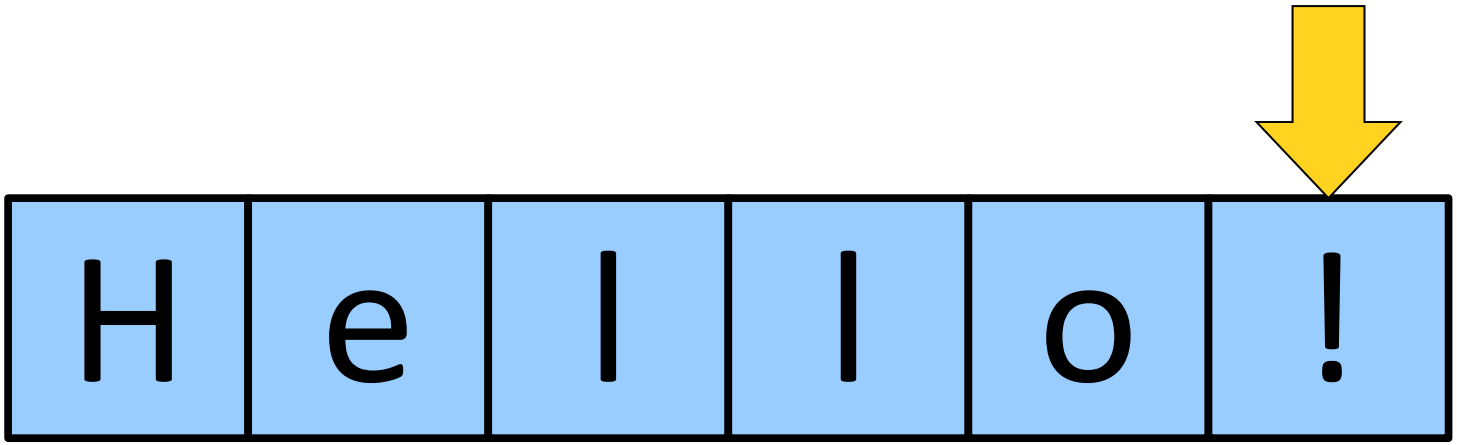

Strings are Immutable

- Java strings are ***immutable***: once a string has been created you cannot set characters.
- To change a string:
  - ***Create a new string*** holding the new value you want it to have via concatenation.
  - Reassigning the String variable (that's allowed).
- ***Important consequence:*** if you pass a String into a method, that method cannot modify that string.

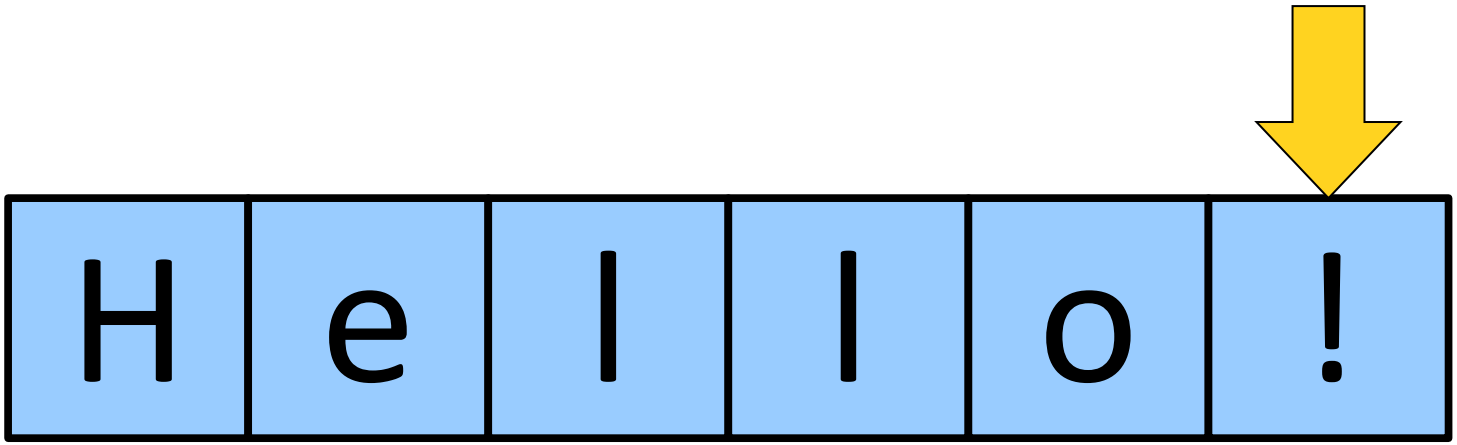

Can survive:
-300F to +300F
Massive radiation
The vacuum of space

Many string algorithms use the “loop and construct” pattern.

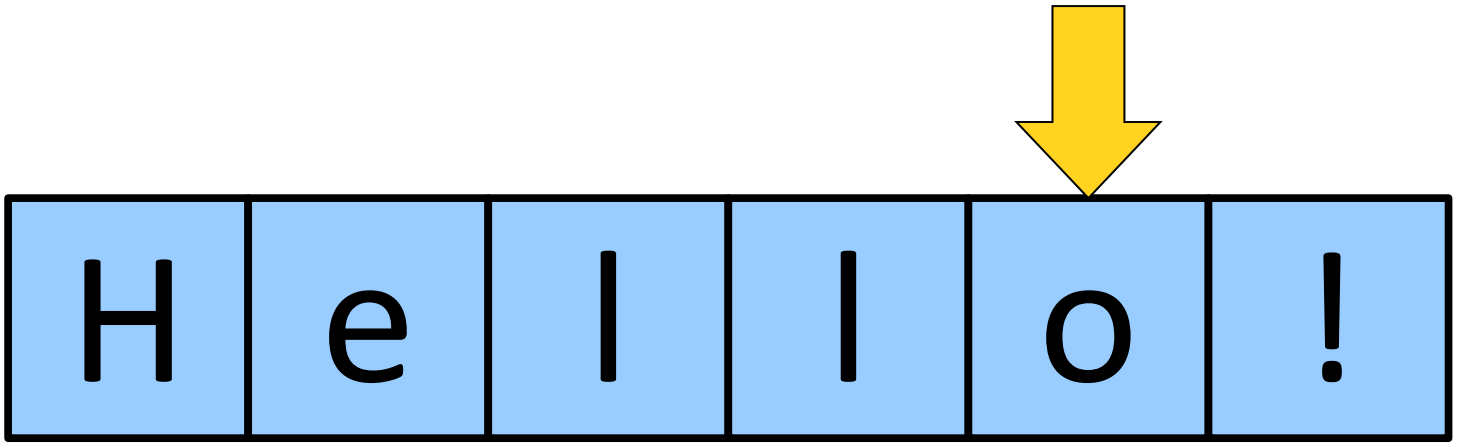
Reversing a String



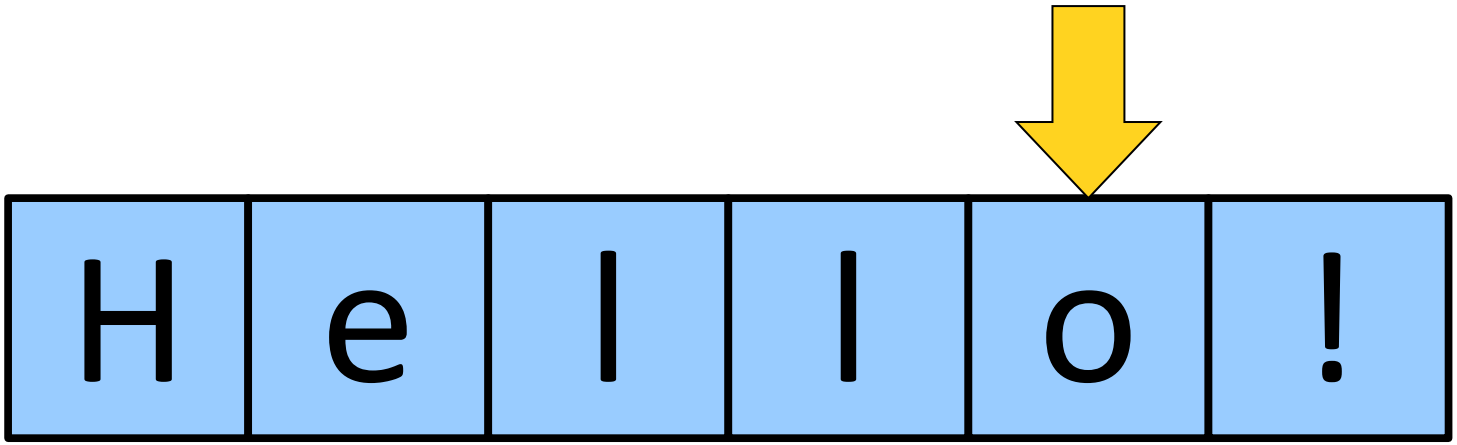
Reversing a String

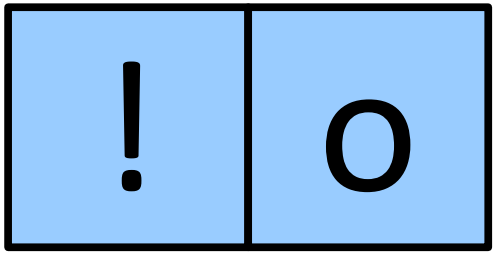


Reversing a String

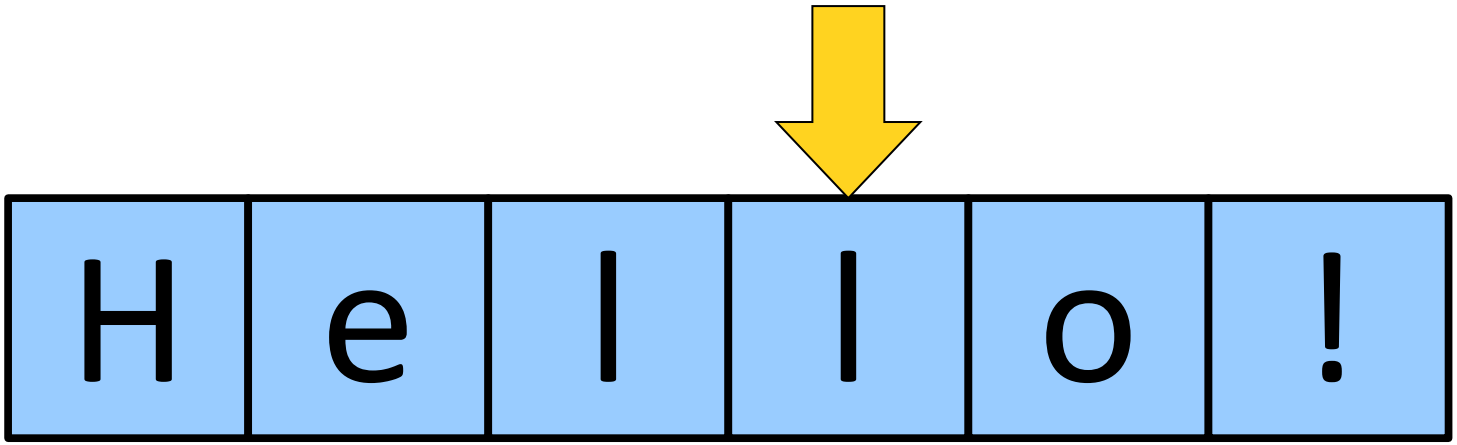


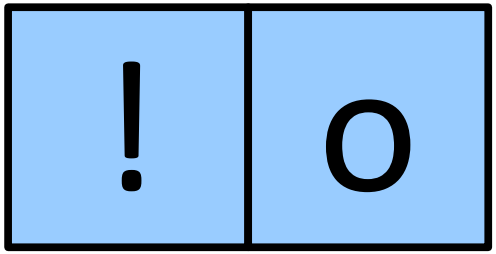
Reversing a String



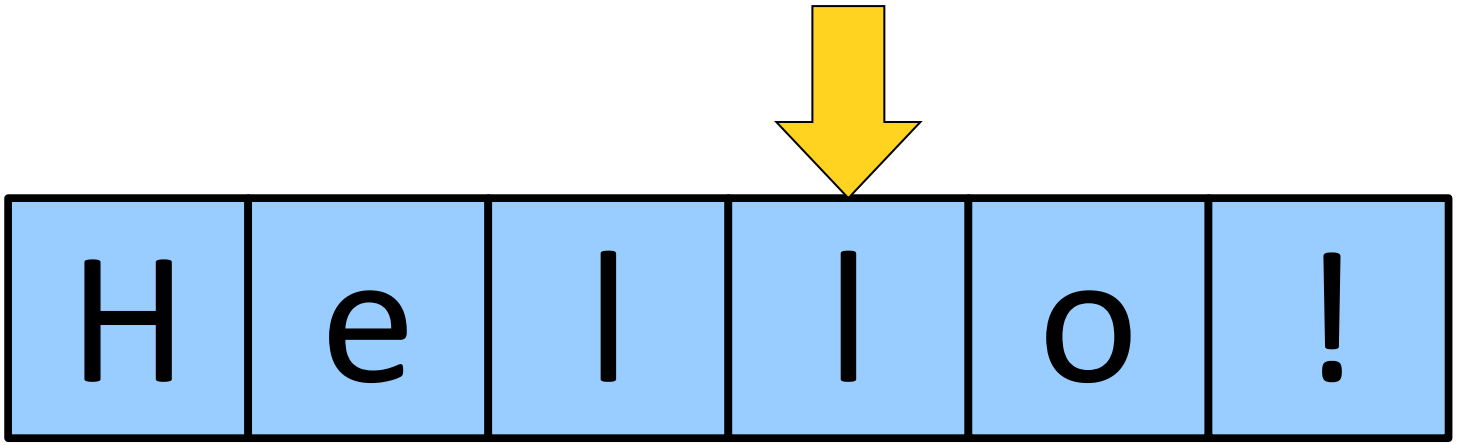


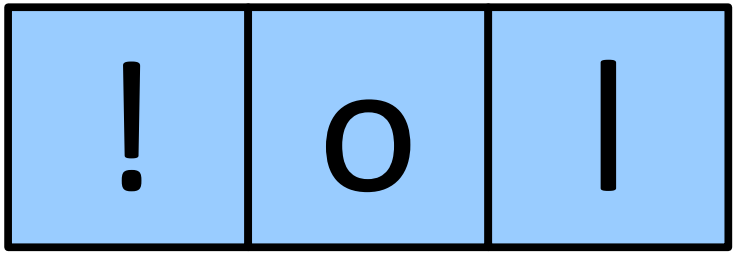
Reversing a String



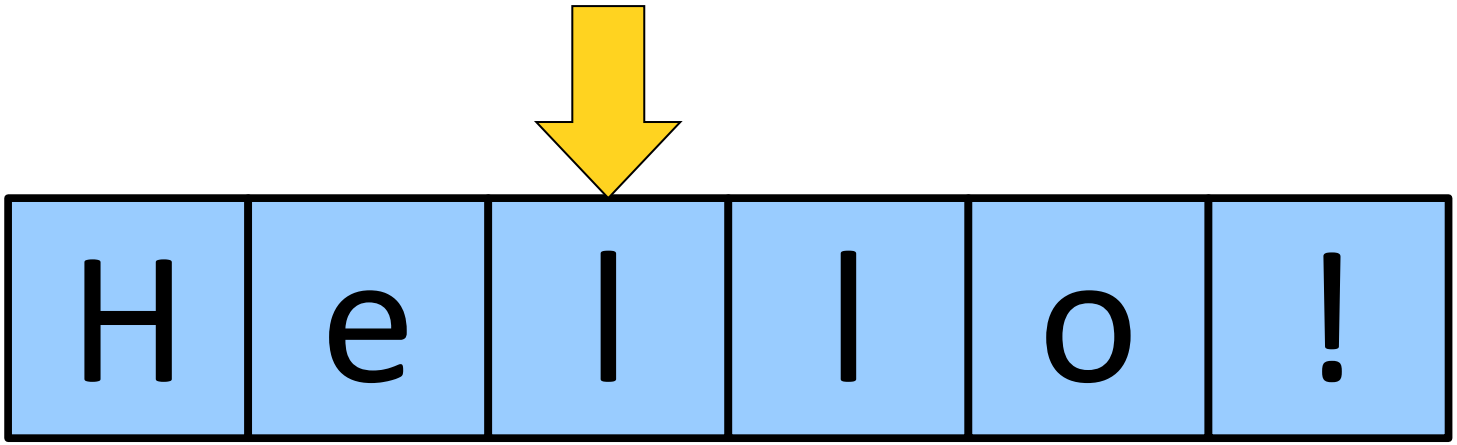


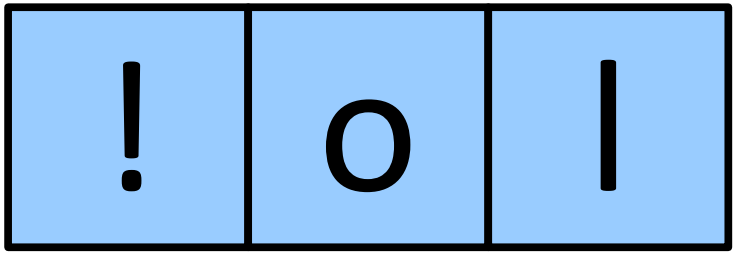
Reversing a String



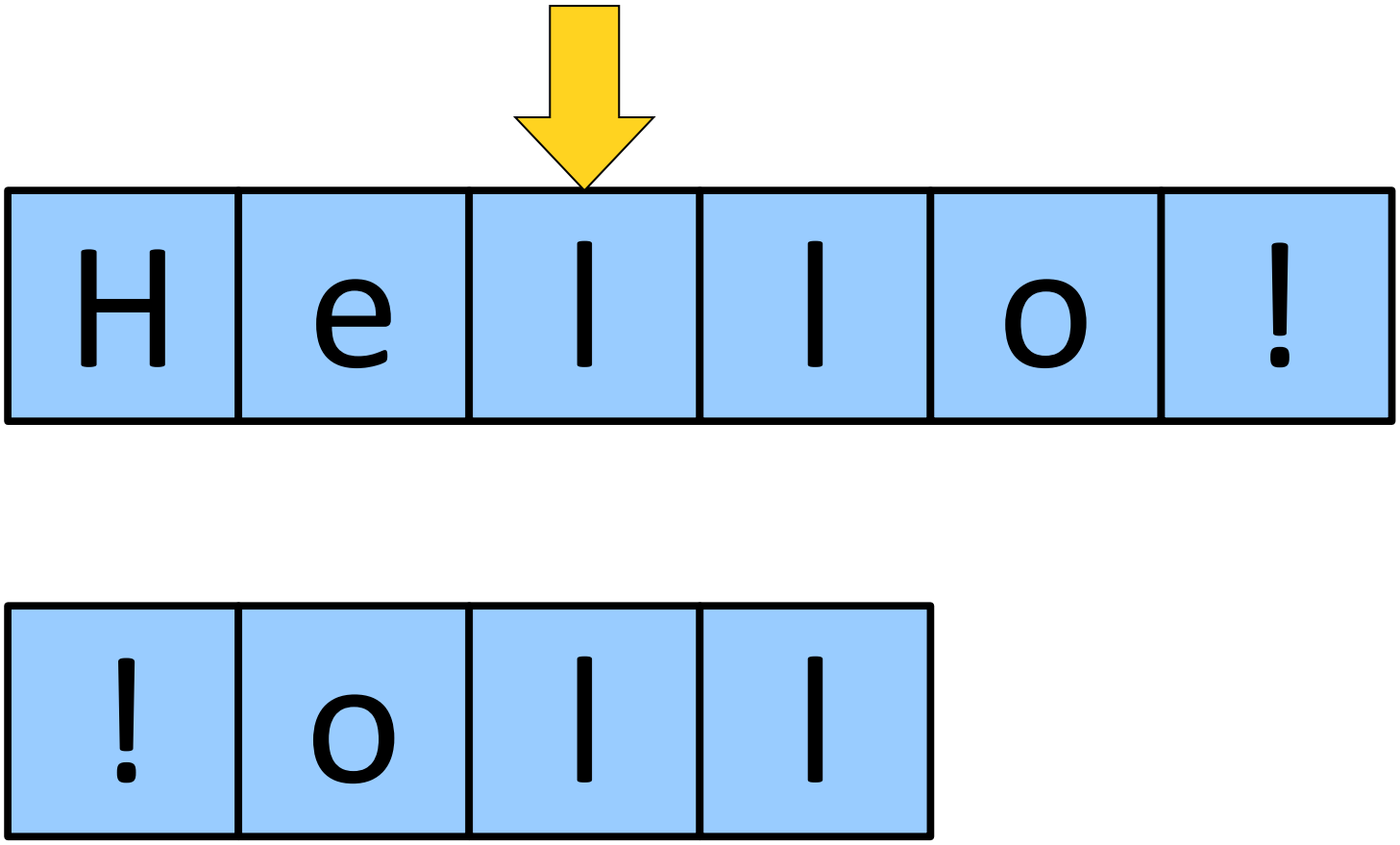


Reversing a String

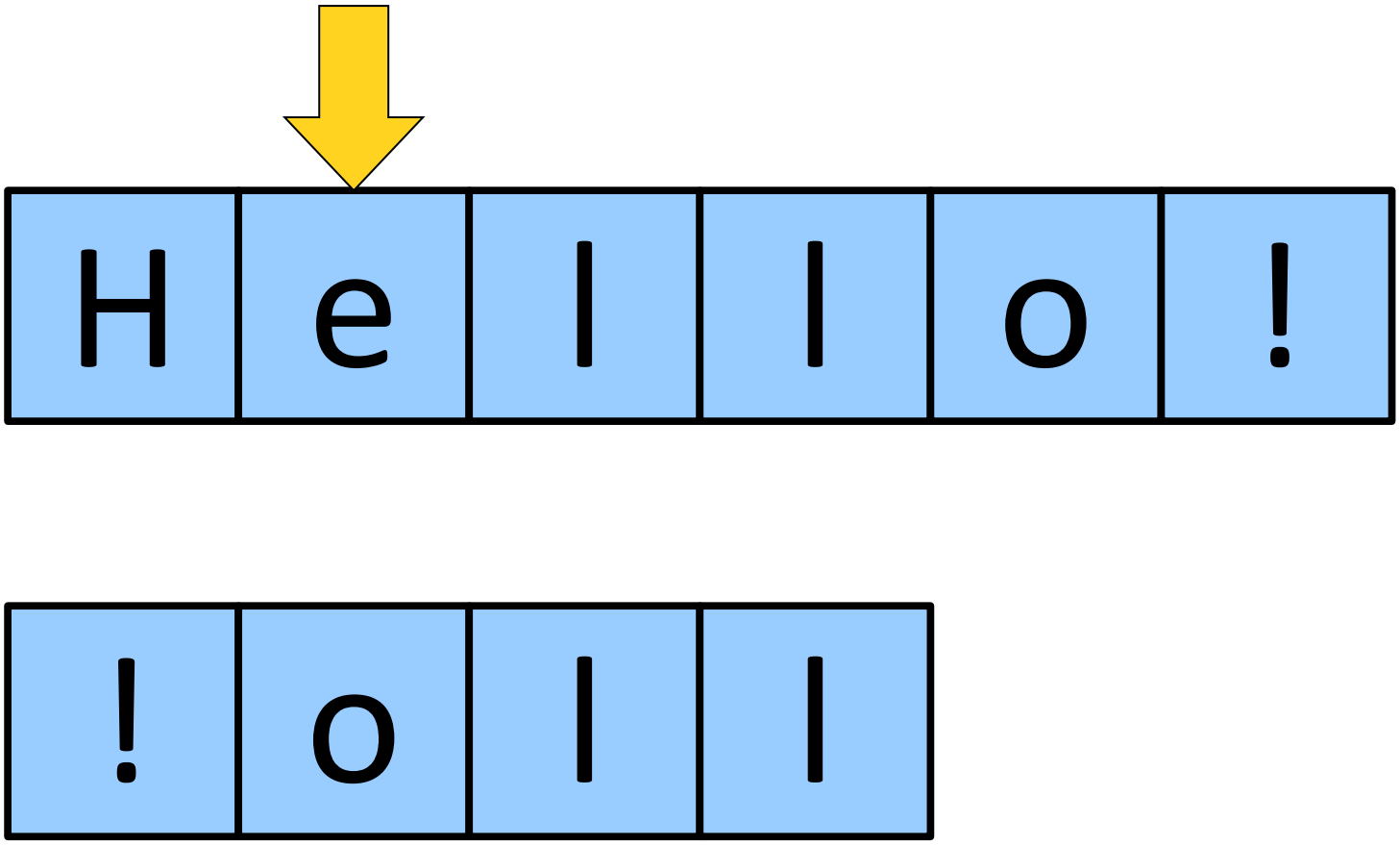




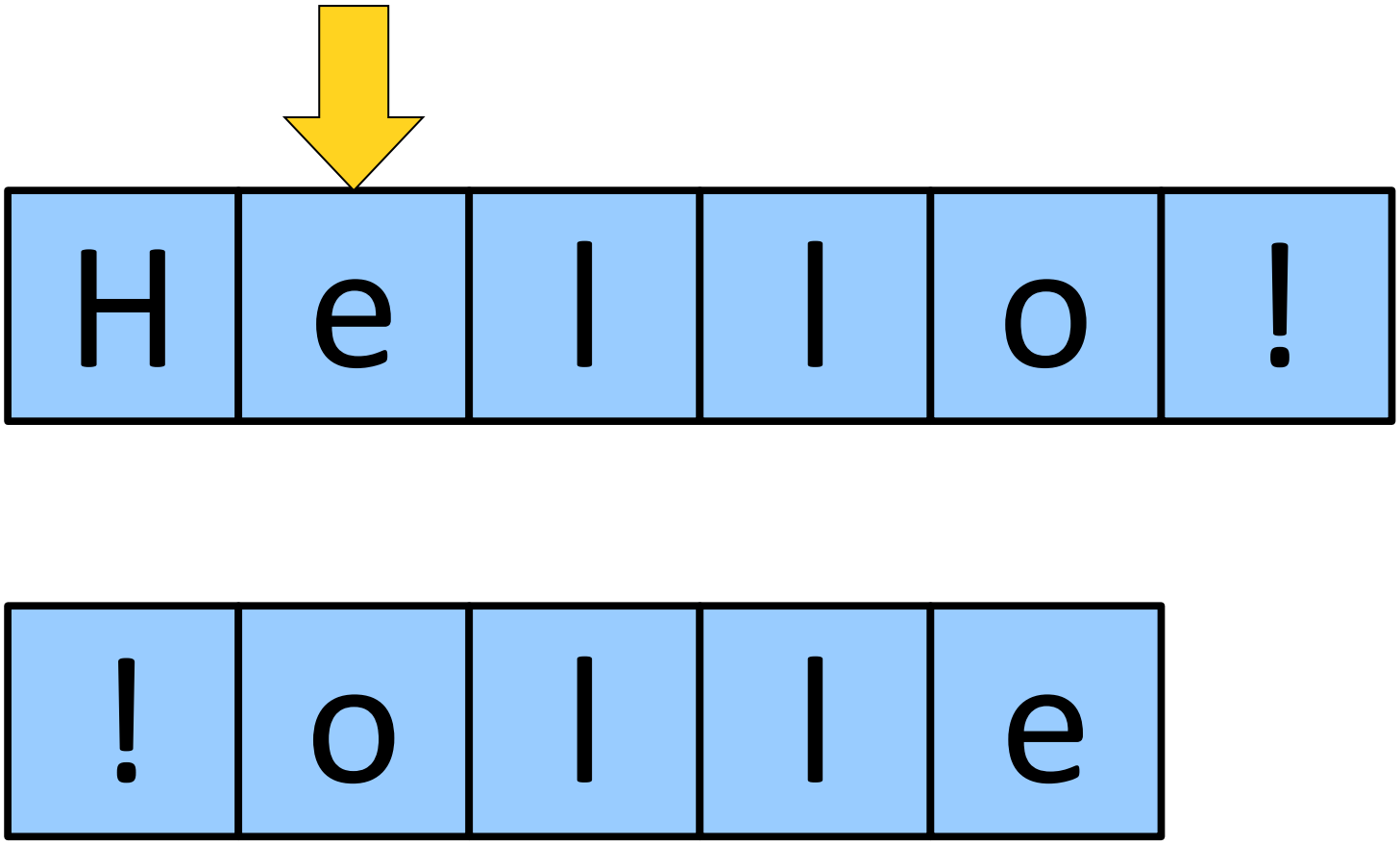
Reversing a String



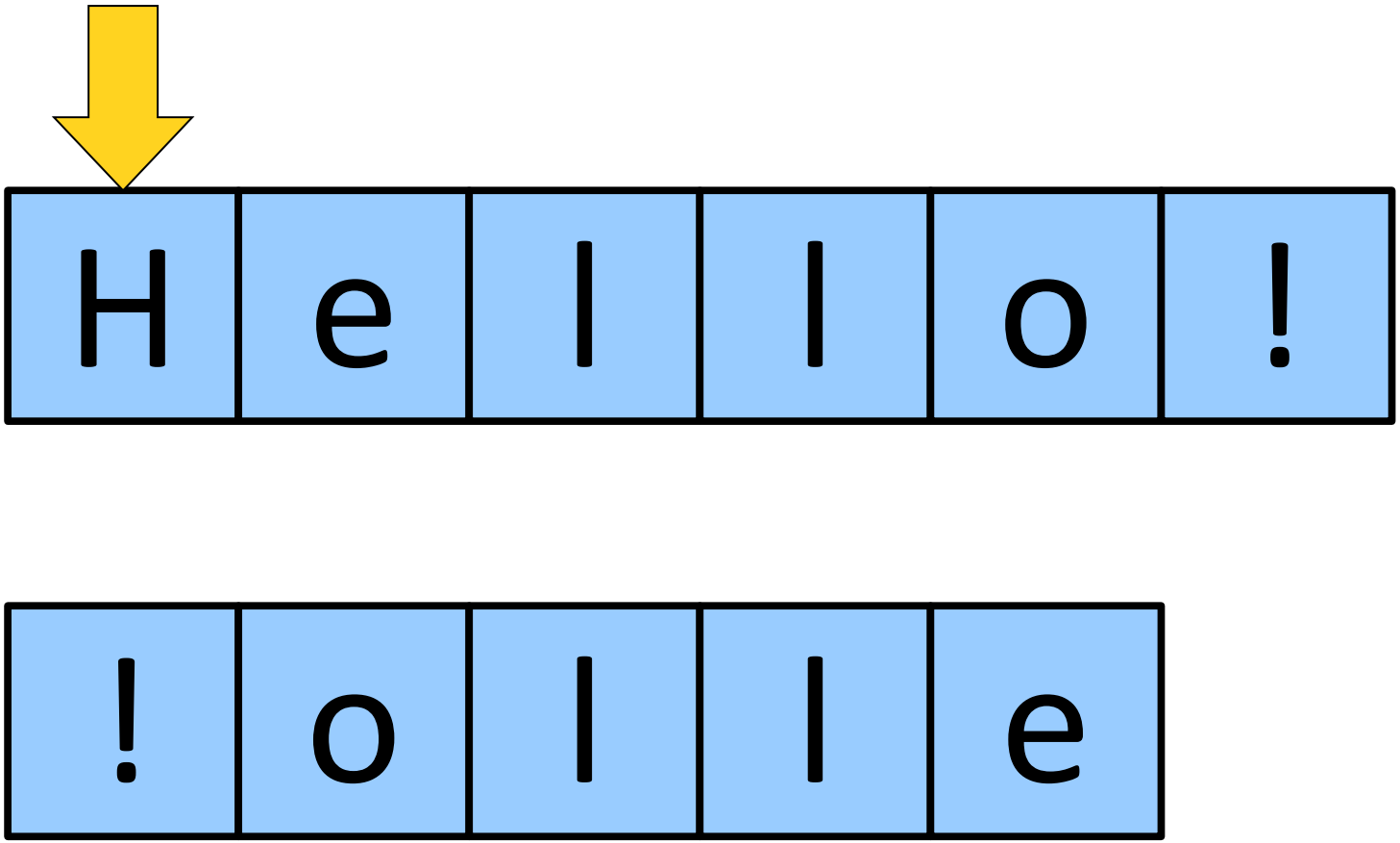
Reversing a String



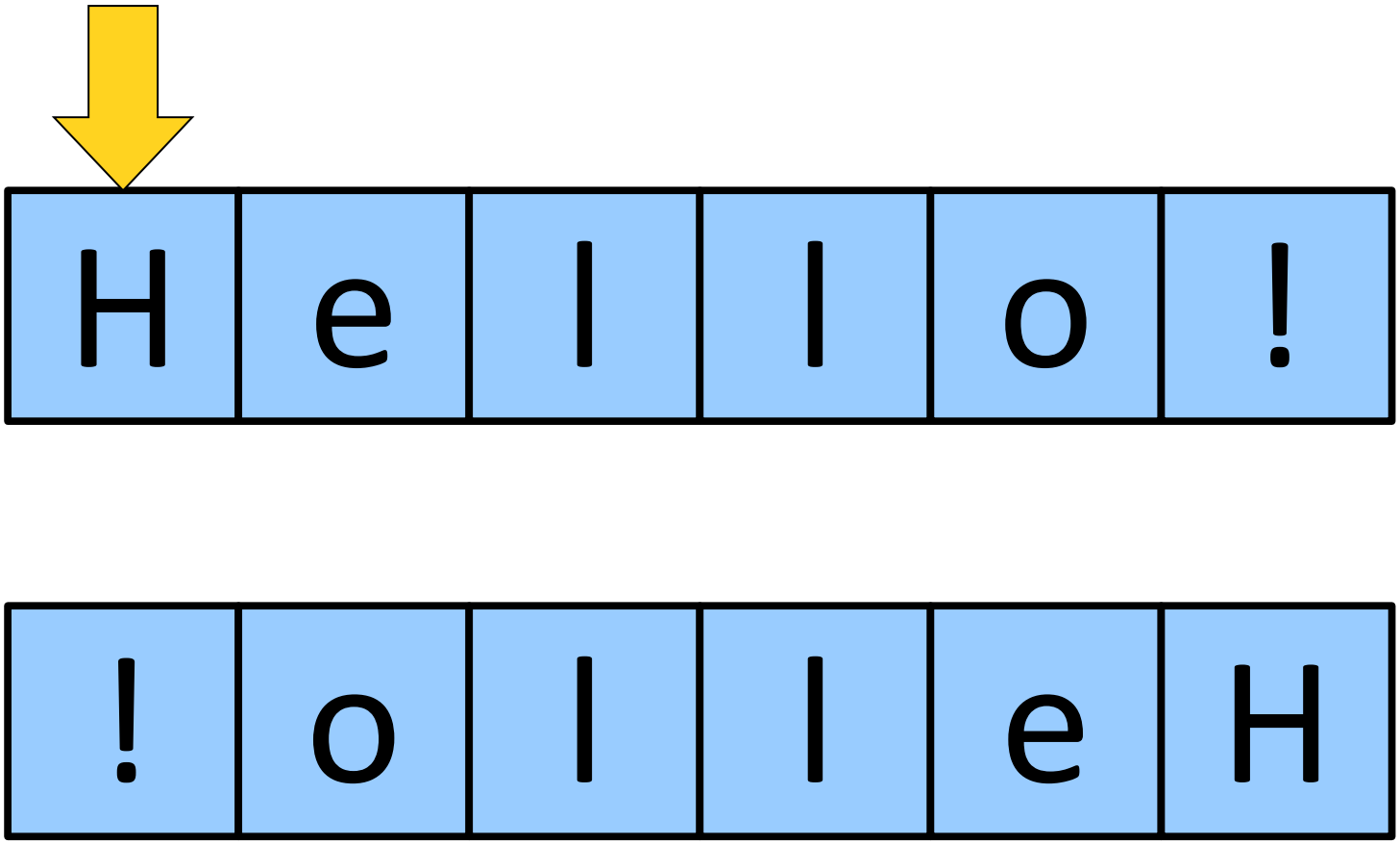
Reversing a String



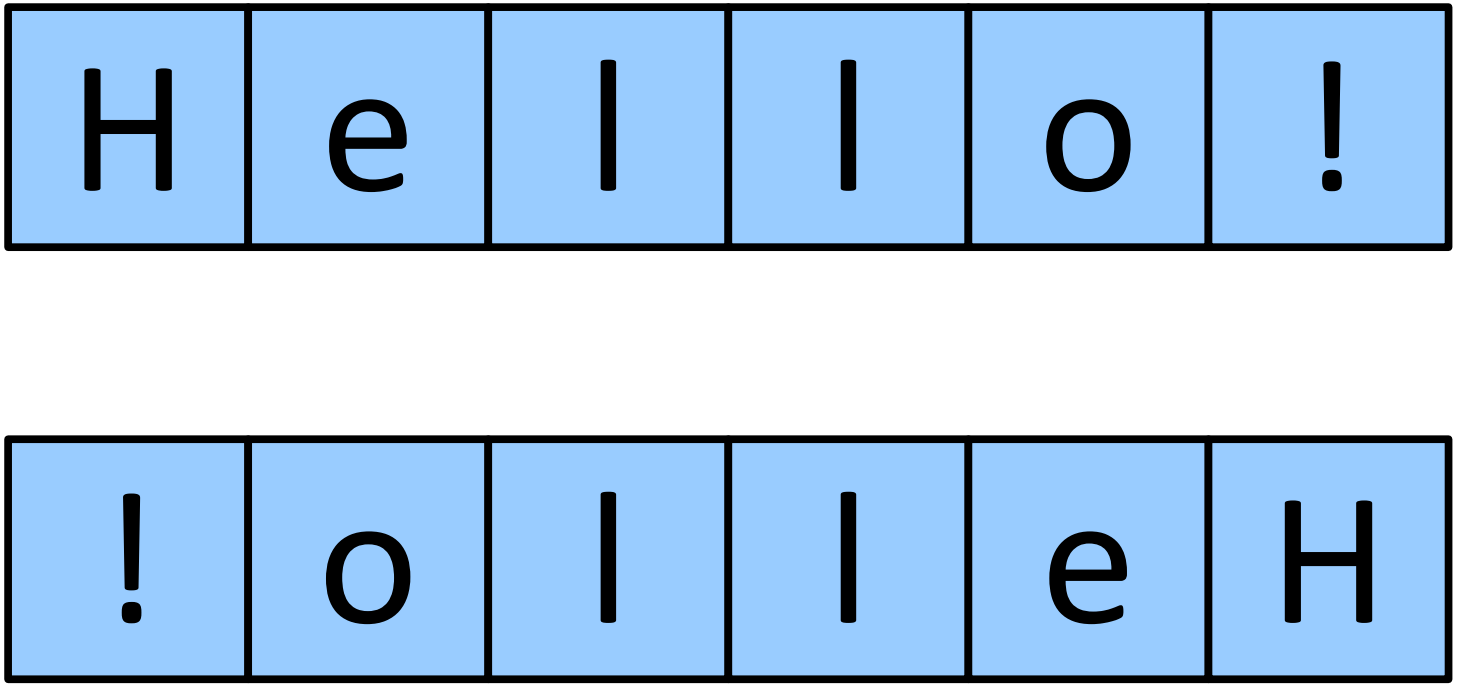
Reversing a String



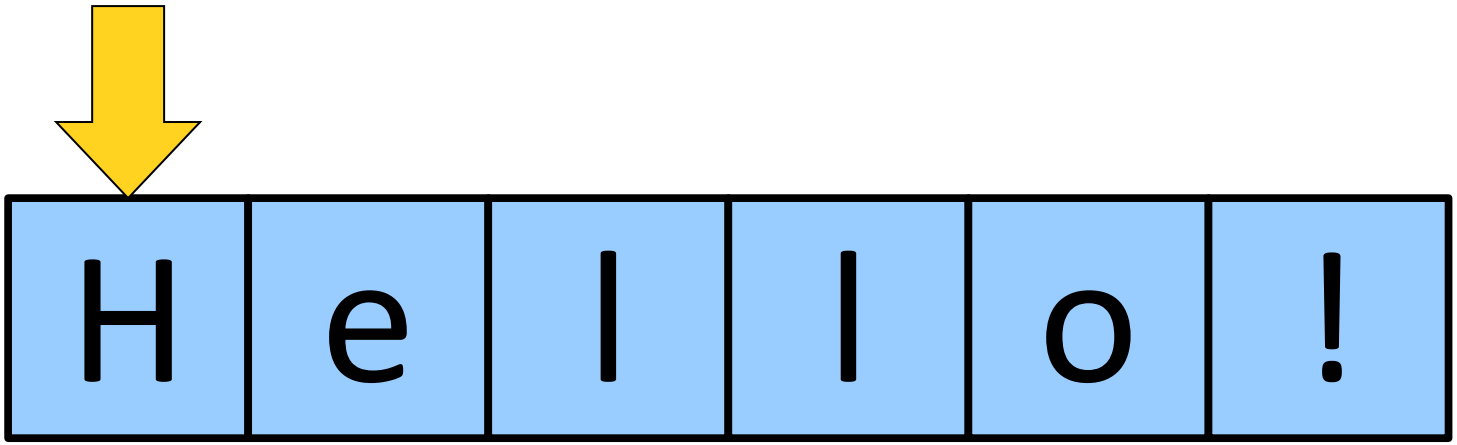
Reversing a String



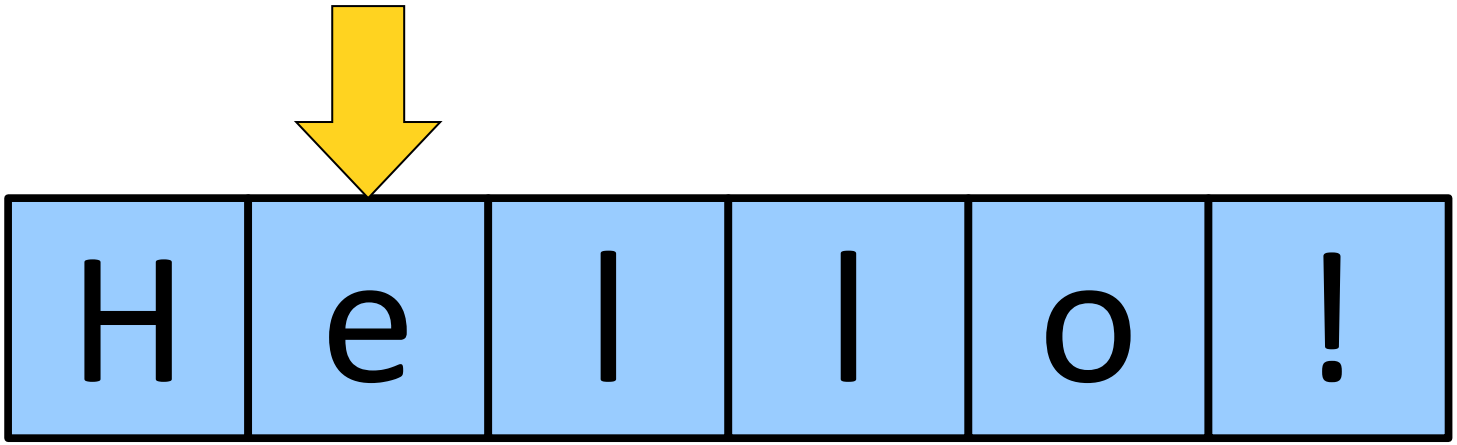
Reversing a String

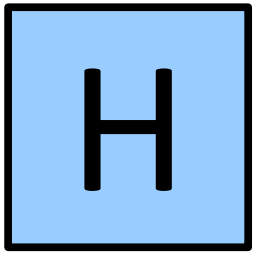


Reversing a String

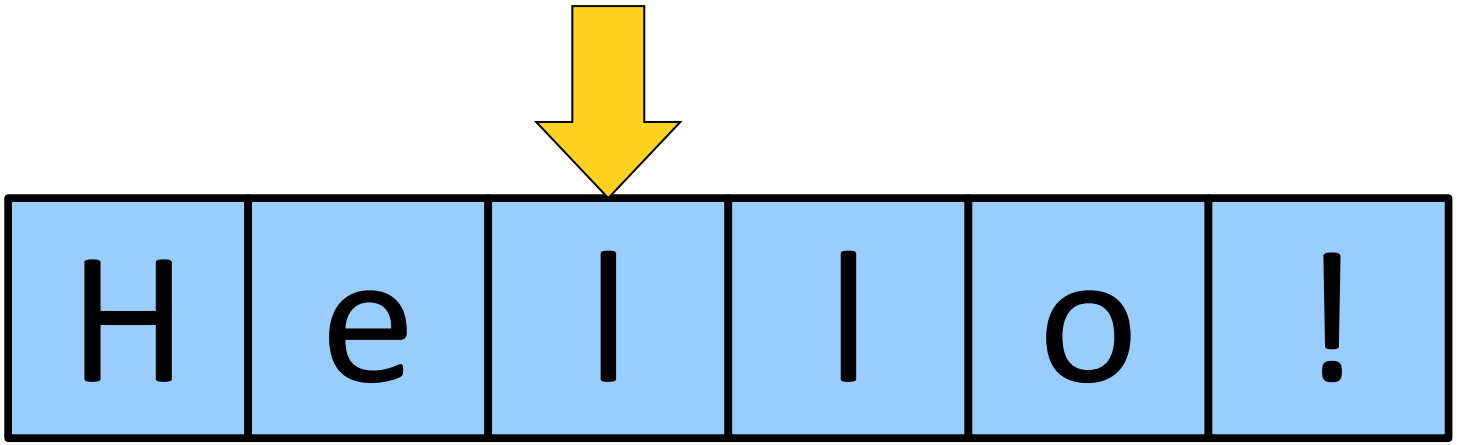


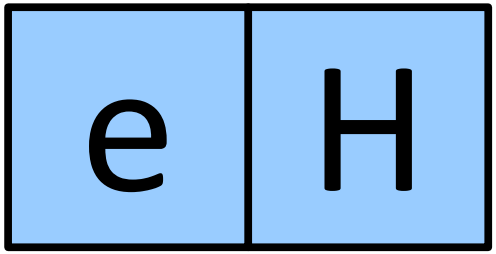
Reversing a String



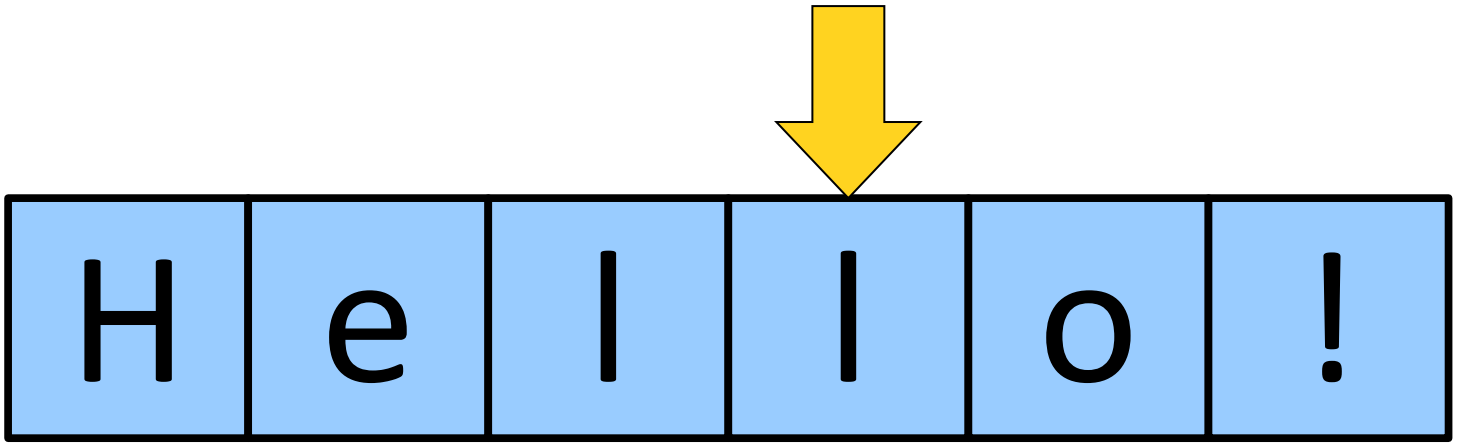


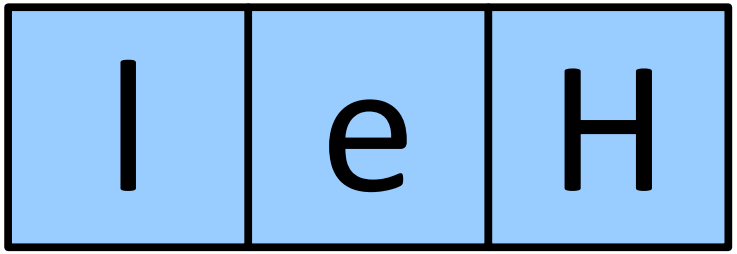
Reversing a String



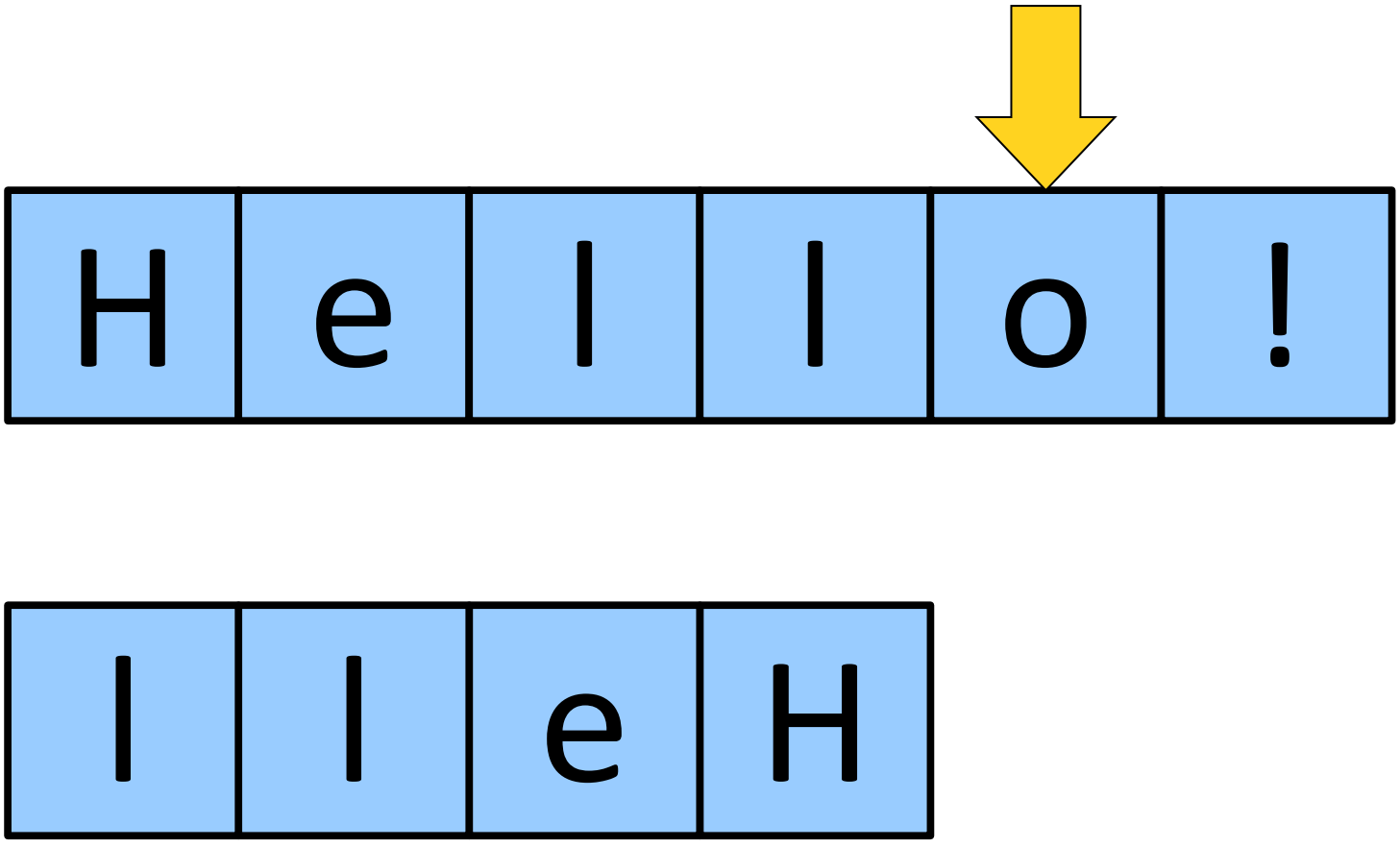


Reversing a String

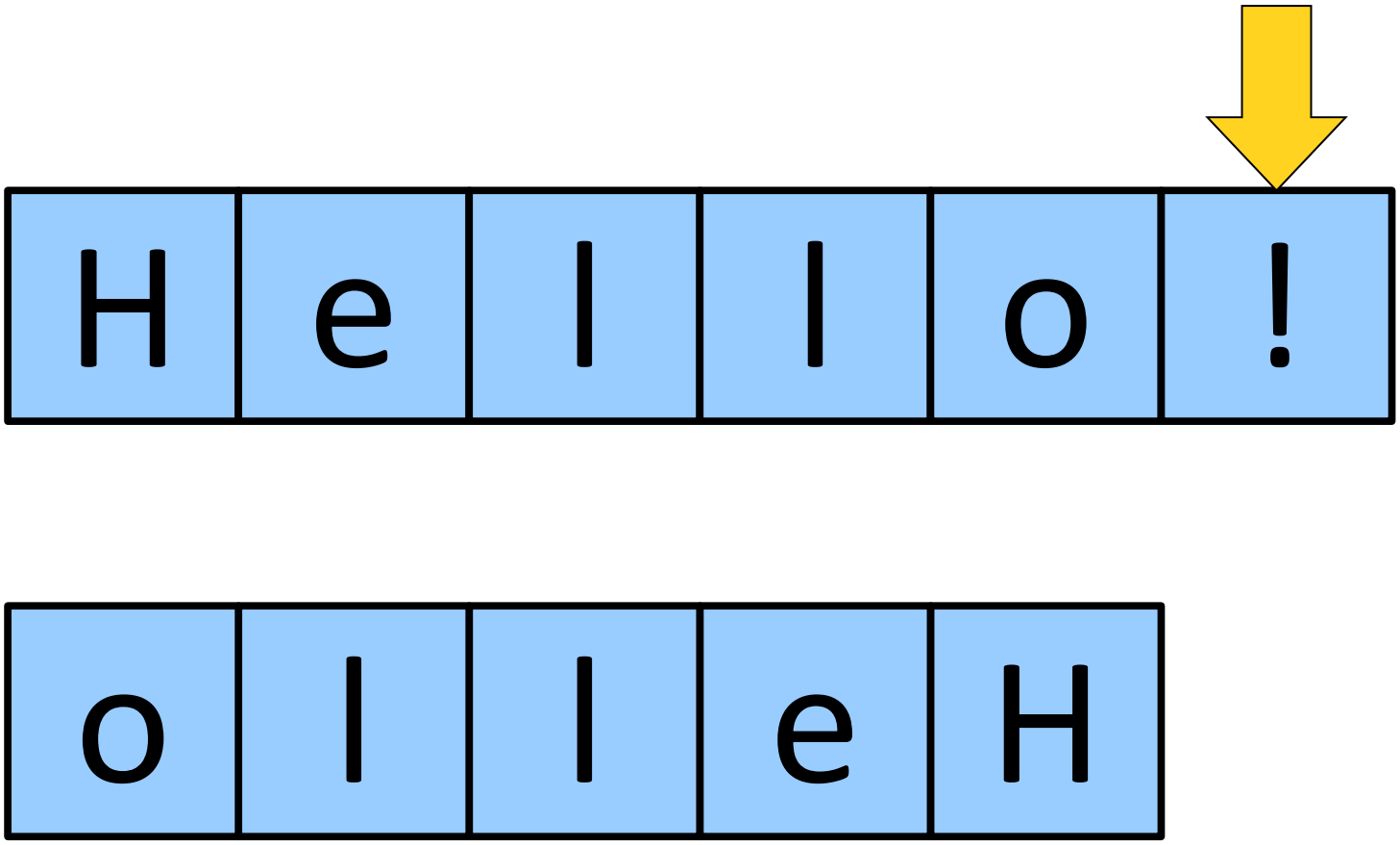




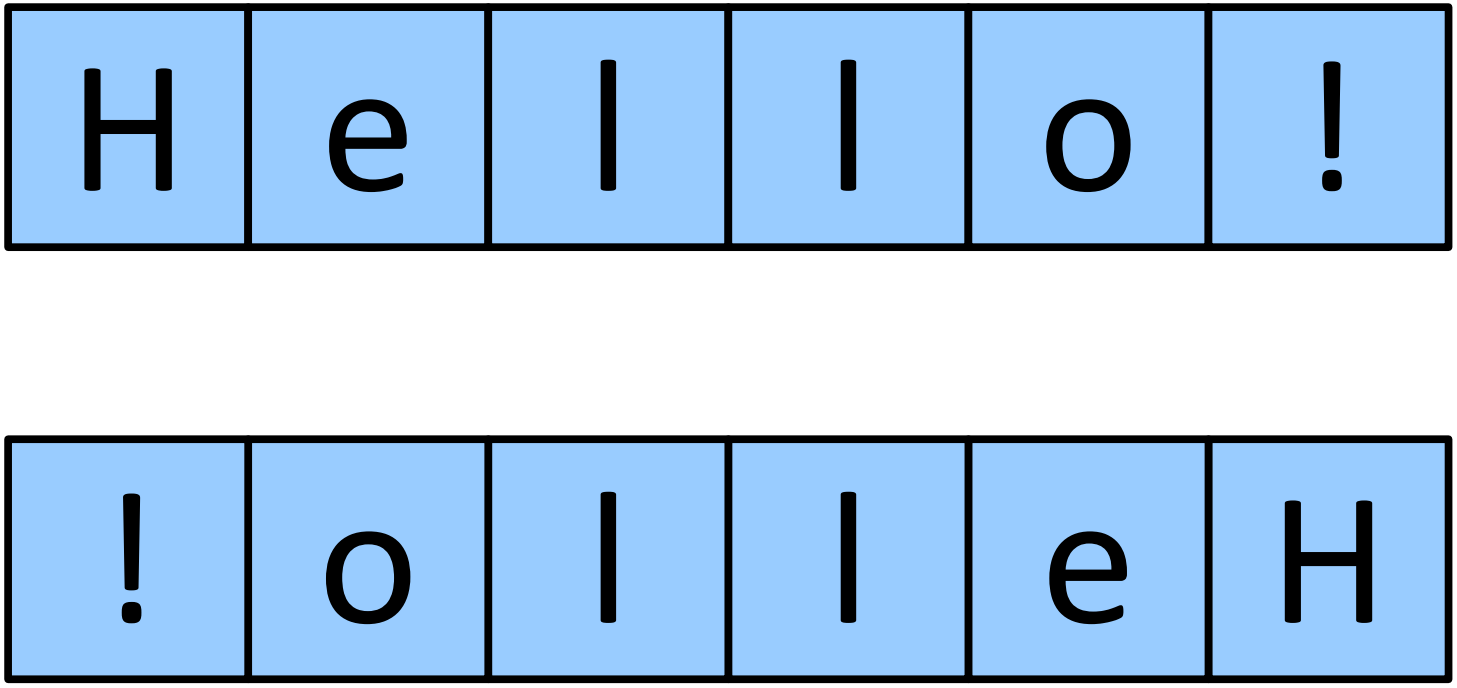
Reversing a String



Reversing a String



Reversing a String

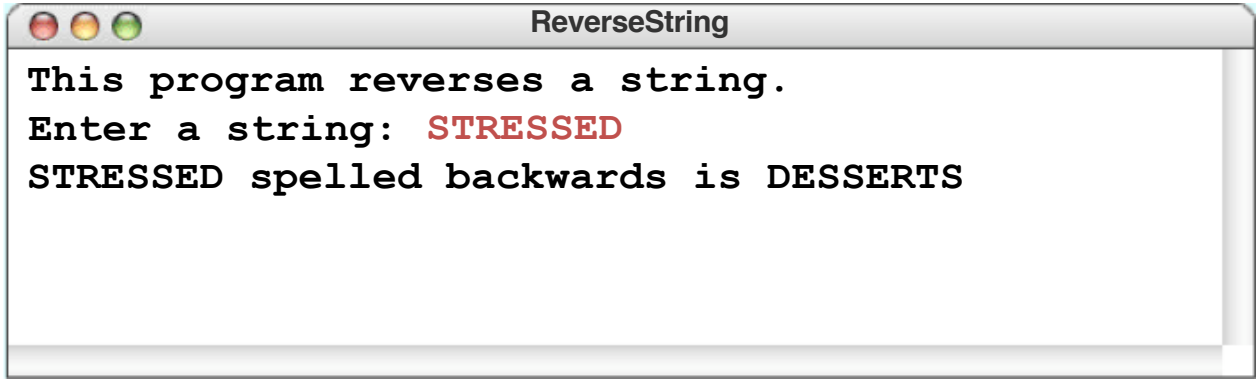


reverseString

```
public void run() {
```

```
private String reverseString(String str) {
    String result = "";
    for ( int i = 0; i < str.length(); i++ ) {
        result = str.charAt(i) + result;
    }
    return result;
}
```

result	str	i
DESSERTS	STRESSED	8

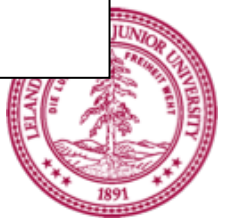


skip simulation

Useful String methods

`int length()` Returns the length of the string
`char charAt(int index)` Returns the character at the specified index. Note: Strings indexed starting at 0.
`String substring(int p1, int p2)` Returns the substring beginning at **`p1`** and extending up to but not including **`p2`**
`String substring(int p1)` Returns substring beginning at **`p1`** and extending through end of string.
`boolean equals(String s2)` Returns true if string **`s2`** is equal to the receiver string. This is case sensitive.
`int compareTo(String s2)` Returns integer whose sign indicates how strings compare in lexicographic order
`int indexOf(char ch)` *or* **`int indexOf(String s)`** Returns index of first occurrence of the character or the string, or -1 if not found
`String toLowerCase()` *or* **`String toUpperCase()`** Returns a lowercase or uppercase version of the receiver string

Using portions of slides by Eric Roberts

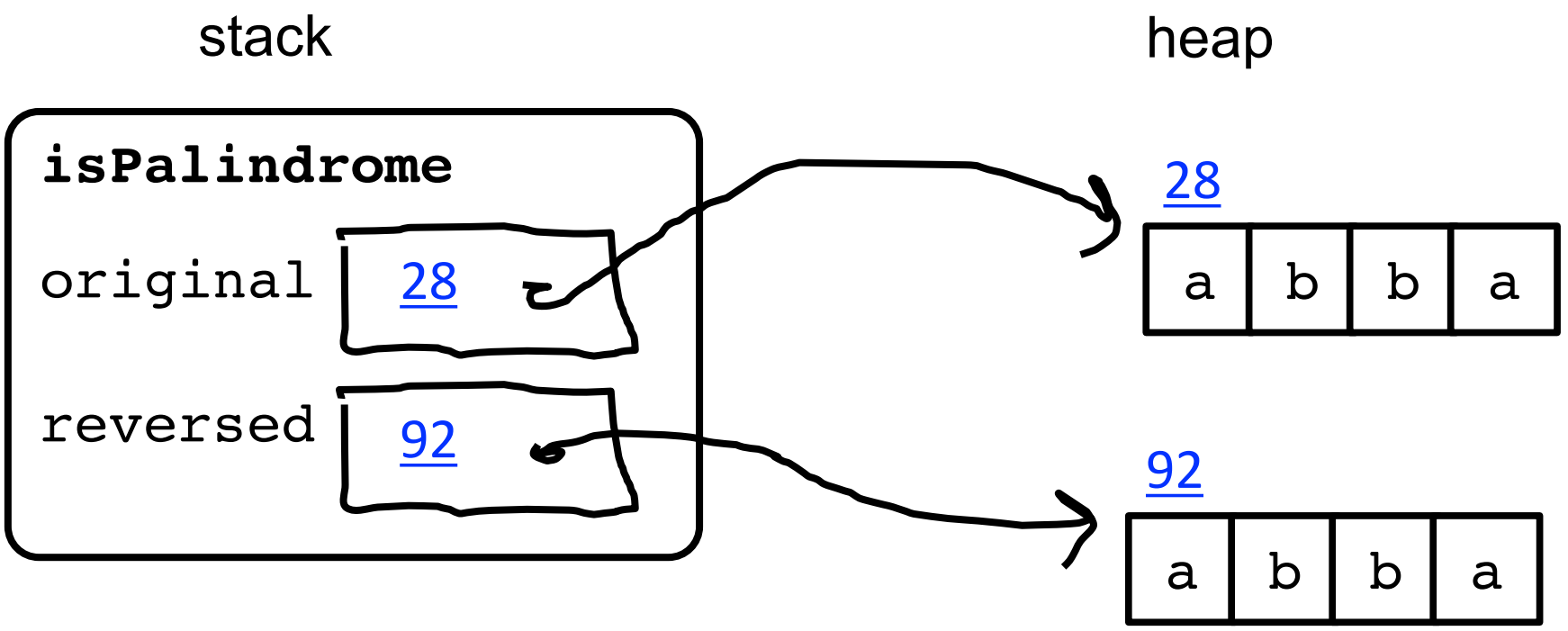


Palindrome

- A ***palindrome*** is a string that reads the same forwards and backwards.
- For example:
  - Abba
  - Racecar
  - Kayak
  - Mr. Owl ate my metal worm.
  - A man, a plan, a canal – panama!
  - Amo la pacífica paloma
  - Нажал кабан на баклажан
  - Elu par cette crapule

What went wrong?

```
private boolean isPalindrome(String original) {
    String reversed = reverse(original);
    return reversed == original;
}
```



Use .equals to compare strings, not ==

Some test cases

- Let's test our program on some examples:
  - Racecar
  - Kayak
  - Mr. Owl ate my metal worm.
  - Go hang a salami! I'm a lasagna hog.
- Will it work?

Stress Test

A man, a plan, a caret, a ban, a myriad, a sum, a lac, a liar, a hoop, a pint, a catalpa, a gas, an oil, a bird, a yell, a vat, a caw, a pax, a wag, a tax, a nay, a ram, a cap, a yam, a gay, a tsar, a wall, a car, a luger, a ward, a bin, a woman, a vassal, a wolf, a tuna, a nit, a pall, a fret, a watt, a bay, a daub, a tan, a cab, a datum, a gall, a hat, a tag, a zap, a say, a jaw, a lay, a wet, a gallop, a tug, a trot, a trap, a tram, a torr, a caper, a top, a tonk, a toll, a ball, a fair, a sax, a minim, a tenor, a bass, a passer, a capital, a rut, an amen, a ted, a cabal, a tang, a sun, an ass, a maw, a sag, a jam, a dam, a sub, a salt, an axon, a sail, an ad, a wadi, a radian, a room, a rood, a rip, a tad, a pariah, a revel, a reel, a reed, a pool, a plug, a pin, a peek, a parabola, a dog, a pat, a cud, a nu, a fan, a pal, a rum, a nod, an eta, a lag, an eel, a batik, a mug, a mot, a nap, a maxim, a mood, a leek, a grub, a gob, a gel, a drab, a citadel, a total, a cedar, a tap, a gag, a rat, a manor, a bar, a gal, a cola, a pap, a yaw, a tab, a raj, a gab, a nag, a pagan, a bag, a jar, a bat, a way, a papa, a local, a gar, a baron, a mat, a rag, a gap, a tar, a decal, a tot, a led, a tic, a bard, a leg, a bog, a burg, a keel, a doom, a mix, a map, an atom, a gum, a kit, a baleen, a gala, a ten, a don, a mural, a pan, a faun, a ducat, a pagoda, a lob, a rap, a keep, a nip, a gulp, a loop, a deer, a leer, a lever, a hair, a pad, a tapir, a door, a moor, an aid, a raid, a wad, an alias, an ox, an atlas, a bus, a madam, a jag, a saw, a mass, an anus, a gnat, a lab, a cadet, an em, a natural, a tip, a caress, a pass, a baronet, a minimax, a sari, a fall, a ballot, a knot, a pot, a rep, a carrot, a mart, a part, a tort, a gut, a poll, a gateway, a law, a jay, a sap, a zag, a tat, a hall, a gamut, a dab, a can, a tabu, a day, a batt, a waterfall, a patina, a nut, a flow, a lass, a van, a mow, a nib, a draw, a regular, a call, a war, a stay, a gam, a yap, a cam, a ray, an ax, a tag, a wax, a paw, a cat, a valley, a drib, a lion, a saga, a plat, a catnip, a pooh, a rail, a calamus, a dairyman, a bater, a canal – Panama!

Remember!

Counterfeiter

You (Distributor)

User

Can you solve it?