

Manipulating Text

An Interesting Article

“How Revolutionary Tools
Cracked a 1700s Code”

<http://www.nytimes.com/2011/10/25/science/25code.html>

|oghnamōrλuvzīgriγm|érzueenjyra=rzr+h̄m̄h̄m̄z̄y|úλx
 zu f̄j̄r̄z̄īp̄h̄λu r̄h̄x̄ |n̄ōm̄ȳx̄l̄d̄a+l̄+ūīs̄ūst̄ |λ̄t̄x̄+w̄r̄p̄ēh̄ |h̄r̄p̄r̄r̄
 ππ̄ī|h̄x̄īd̄ūλ̄h̄p̄ȳz̄p̄h̄j̄d̄ōλ̄h̄j̄n̄ū|ōp̄h̄r̄λ̄m̄ōz̄λ̄ūh̄|ōt̄h̄n̄o=rzκ
 h̄c̄|z̄ūn̄:s̄c̄φ̄ |λ̄f̄π̄īz̄t̄ |λ̄n̄z̄ūv̄ōm̄z̄ȳπ̄r̄λ̄ūr̄z̄h̄|w̄ |λ̄f̄p̄īd̄c̄r̄p̄
 c̄p̄d̄ūoz̄p̄īn̄īs̄ḡ|p̄ȳḡōp̄d̄ē̄t̄d̄c̄h̄r̄h̄d̄ē̄ |λ̄b̄ȳ+r̄c̄p̄ |ūf̄x̄n̄z̄ḡt̄z̄h̄p̄ūh̄
 =r̄|j̄āz̄b̄h̄f̄|:z̄ē̄|s̄ū̄ȳx̄ūez̄r̄īḡ:z̄c̄ū̄x̄λ̄ḡm̄īz̄p̄r̄+̄r̄p̄ōz̄x̄h̄īḡp̄ȳ
 π̄r̄λ̄f̄z̄h̄āgr̄ē̄h̄λ̄āst̄h̄r̄z̄h̄ī=|s̄z̄ūt̄ |λ̄d̄j̄n̄īλ̄īx̄ḡz̄n̄|d̄j̄h̄t̄h̄oē
 • p̄ūoogz̄iō|ū+c̄n̄x̄x̄k̄m̄īc̄:k̄ȳr̄d̄+̄īr̄p̄h̄r̄z̄λ̄s̄λ̄c̄
 • x̄n̄z̄f̄s̄=h̄p̄m̄h̄d̄ūd̄s̄ā̄ |λ̄ūb̄+̄z̄h̄r̄p̄z̄ḡs̄x̄d̄=r̄λ̄ū̄|f̄īz̄ōḡūb̄h̄t̄c̄ū
 • x̄n̄m̄x̄īc̄d̄r̄λ̄ūv̄ |d̄l̄c̄īn̄h̄λ̄z̄ū̄z̄h̄n̄:ō̄j̄m̄ē.
 z̄p̄r̄v̄π̄īz̄ȳd̄īz̄īr̄z̄ā |λ̄b̄c̄p̄|λ̄f̄h̄h̄+d̄|ōv̄z̄h̄n̄:w̄p̄ūr̄x̄p̄|c̄:ū
 c̄=ḡp̄b̄c̄īh̄j̄c̄īr̄x̄|n̄f̄+̄)d̄f̄r̄īp̄o=r̄z̄b̄ḡh̄r̄z̄ū̄+̄l̄p̄ē̄r̄n̄z̄j̄ū̄+̄d̄n̄īū |λ̄
 t̄z̄ē̄ḡī+t̄r̄ī=|m̄f̄r̄īz̄n̄|k̄r̄x̄āb̄ |λ̄ō̄j̄n̄īλ̄c̄x̄m̄z̄f̄π̄ūḡ-λ̄h̄f̄
 λ̄h̄r̄d̄īj̄p̄ȳh̄ōλ̄āt̄+̄p̄d̄ē̄λ̄s̄z̄āz̄f̄p̄ȳj̄h̄d̄īz̄ūz̄ō |λ̄z̄ūp̄ōh̄īḡz̄īp̄r̄
 λ̄ūv̄z̄n̄+̄h̄λ̄f̄|d̄c̄:īr̄: |λ̄d̄ōz̄z̄h̄=|ū+̄īλ̄x̄z̄ūb̄m̄ōr̄λ̄ūf̄:
 • π̄p̄z̄ū̄j̄c̄ī=̄m̄ā̄l̄ū̄n̄p̄r̄x̄z̄z̄b̄|d̄c̄:n̄z̄h̄ē̄|ō̄j̄m̄s̄p̄ȳr̄ūd̄ḡh̄ō
 • +h̄īc̄|ḡh̄īḡz̄īz̄m̄p̄īc̄|c̄s̄=ḡē̄h̄z̄ūh̄p̄ō̄j̄f̄c̄p̄r̄c̄īr̄ūw̄ |ōo=̄m̄z̄h̄ȳm̄l̄
 x̄h̄r̄

ερελ
περ

• x̄h̄r̄āz̄p̄r̄ī=|r̄ȳr̄λ̄h̄d̄ūc̄ |Δ̄f̄h̄r̄x̄īp̄k̄ī=̄r̄ūēō̄h̄ū̄j̄ō̄f̄ō̄ūb̄d̄ō̄r̄h̄
 • p̄īz̄λ̄h̄|s̄ō̄m̄īp̄r̄īr̄d̄ō̄ḡl̄=̄m̄z̄l̄h̄īḡūh̄r̄t̄z̄p̄z̄āz̄x̄j̄ō̄h̄λ̄f̄īl̄:̄p̄ḡē̄z̄t̄p̄r̄ī
 • z̄v̄j̄n̄d̄ō̄r̄ḡū̄īḡūp̄m̄h̄t̄r̄ūd̄ |λ̄x̄p̄āz̄ḡc̄p̄r̄c̄h̄p̄ūc̄ |ōz̄x̄r̄d̄ū̄r̄:ū
 • b̄+̄q̄|:ēc̄.
 V̄z̄āz̄x̄t̄r̄īl̄:m̄īh̄p̄ū̄c̄ā̄h̄z̄ē̄λ̄ḡīh̄m̄d̄īc̄|z̄ōn̄:p̄ȳr̄ḡz̄ȳḡūf̄d̄p̄m̄:
 s̄ā̄īr̄ūf̄z̄īr̄w̄ |c̄=̄m̄π̄v̄z̄z̄ |λ̄x̄h̄r̄p̄λ̄h̄h̄t̄+r̄z̄h̄ī|f̄īn̄p̄h̄z̄ē̄n̄ |λ̄ī
 n̄īāc̄r̄ū̄|b̄ȳ|r̄p̄h̄īn̄h̄ȳ|s̄ōz̄īū̄m̄d̄āḡh̄z̄ū̄+̄ḡz̄|d̄r̄x̄r̄d̄s̄z̄ā̄ |ḡ |ōo
 p̄r̄d̄c̄q̄j̄ē̄λ̄b̄z̄h̄īk̄|p̄r̄ḡp̄h̄t̄|f̄ā̄l̄īz̄|ḡx̄ḡ|r̄r̄ūd̄x̄ā̄h̄ū̄ȳ+̄ūp̄r̄λ̄ū|s̄īλ̄-λ̄ȳ
 ōr̄ūām̄īō̄r̄h̄ī=̄r̄s̄p̄ūḡm̄īn̄h̄z̄ūb̄d̄ā̄j̄|p̄ḡz̄f̄p̄λ̄n̄|h̄ē̄x̄ū̄c̄|āz̄n̄z̄p̄d̄h̄z̄
 p̄λ̄h̄h̄t̄+r̄z̄h̄āz̄p̄r̄c̄īx̄p̄ȳ|ūē̄īm̄ḡ=ḡp̄n̄v̄r̄z̄īz̄|ḡs̄=̄b̄|ō̄h̄n̄p̄r̄ḡn̄h̄p̄
 z̄īp̄l̄ē̄n̄t̄īl̄:s̄ū̄f̄ō̄|r̄b̄īz̄=̄s̄x̄s̄ȳr̄ḡūīm̄z̄m̄t̄|λ̄p̄h̄t̄+̄d̄n̄ā̄p̄|ḡh̄c̄
 c̄ūd̄īr̄m̄ū̄|ūz̄ūc̄x̄c̄p̄d̄īx̄r̄p̄z̄z̄ō̄j̄f̄z̄īz̄h̄d̄ȳē̄j̄ūz̄ā |λ̄n̄p̄r̄t̄c̄ȳf̄
 |λ̄n̄z̄r̄ī+̄h̄λ̄ḡz̄īz̄c̄ |Δ̄m̄.
 P̄r̄īm̄ā̄ōōλ̄īz̄m̄λ̄h̄=̄c̄b̄
 d̄ē̄r̄:s̄ȳr̄ūd̄r̄ū̄īn̄|v̄ |λ̄.
 H̄ō̄īḡḡ-λ̄b̄m̄īz̄z̄f̄m̄h̄r̄λ̄c̄|d̄m̄īō̄h̄ē̄p̄h̄t̄īz̄=̄j̄p̄c̄ū̄r̄d̄p̄r̄:λ̄ō
 z̄h̄īp̄r̄īḡz̄n̄|p̄z̄ē̄d̄t̄ȳf̄:|ā̄b̄|s̄λ̄īd̄m̄c̄ī|ūḡx̄r̄z̄x̄t̄r̄ē̄h̄p̄ūc̄ |d̄ān̄:ā̄ī|ō̄ḡ

Announcements

- Breakout! due this Friday at 3:15PM.
 - Stop by the LaIR with questions!
 - Gil and I have office hours - please feel free to stop by!
- YEAH hours (assignment review) for Breakout! tonight in Herrin T-175, 7PM - 9PM.
 - Materials will be posted online.
 - Unfortunately, not recorded.

BRACE YOURSELF

MIDTERM IS COMING

Midterm Logistics

- Midterm is next **Monday, February 11** from **7PM - 10PM** (location TBA).
 - Open-book, open-note, closed-computer.
 - Covers material up through and including Wednesday's lecture.
- Practice exam available now; solutions will be released on Wednesday.
- If you need to take the exam at an alternate time, email Gil (**gilsho@stanford.edu**) no later than Wednesday at 12:50PM.
 - Gil will send out an email about this.

A **string** is a sequence of characters.





H e l l o !

H	e	l	l	o	!
---	---	---	---	---	---

0

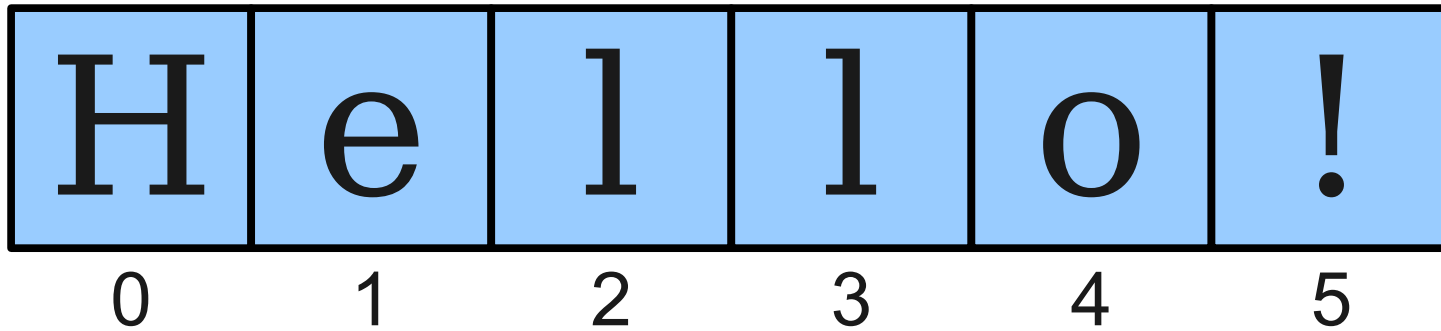
1

2

3

4

5



string.charAt (***index***)

The Data Type **char**

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

```
char letterA = 'A';
```

```
char plus = '+'
```

```
char zero = '0';
```


Escape Sequences

- An **escape sequence** is a sequence of characters in a program's source code that represents a single logical character.
- Examples:
 - `\t`: Horizontal tab
 - `\n`: Newline
 - `\'`: Single quote
 - `\"`: Double quote

Highlights from Character

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.

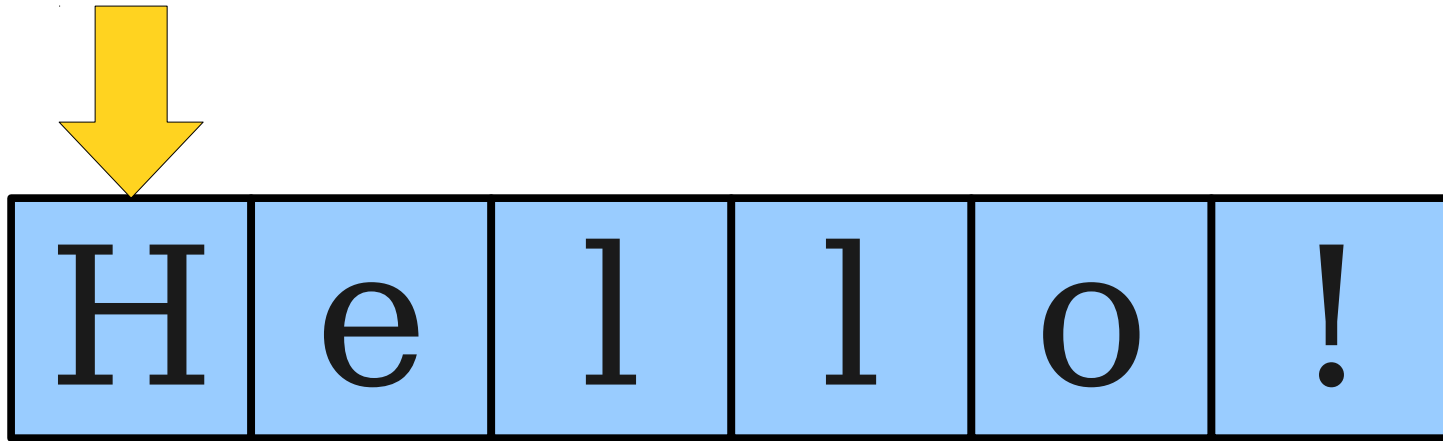
static char toUpperCase(char ch)

Converts **ch** to its uppercase equivalent, if any. If not, **ch** is returned unchanged.

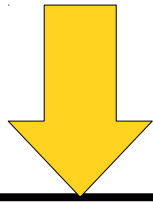
Strings are Immutable

- Java strings are **immutable**: once a string has been created, its contents cannot change.
- To change a string:
 - Create a new string holding the new value you want it to have.
 - Reassign the **String** variable to hold the new value.

Reversing a String



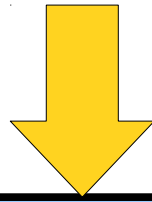
Reversing a String



H	e	l	l	o	!
---	---	---	---	---	---

H

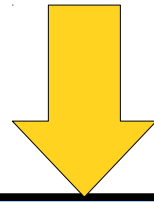
Reversing a String



H	e	l	l	o	!
---	---	---	---	---	---

e	H
---	---

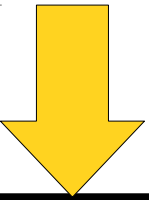
Reversing a String



H	e	l	l	o	!
---	---	---	---	---	---

l	e	H
---	---	---

Reversing a String



Character array: H e l l o !

Character array: l l e H

Reversing a String

↓

H	e	l	l	o	!
---	---	---	---	---	---

o	l	l	e	H
---	---	---	---	---

Reversing a String

H	e	l	l	o	!
---	---	---	---	---	---

!	o	l	l	e	H
---	---	---	---	---	---

Palindromes

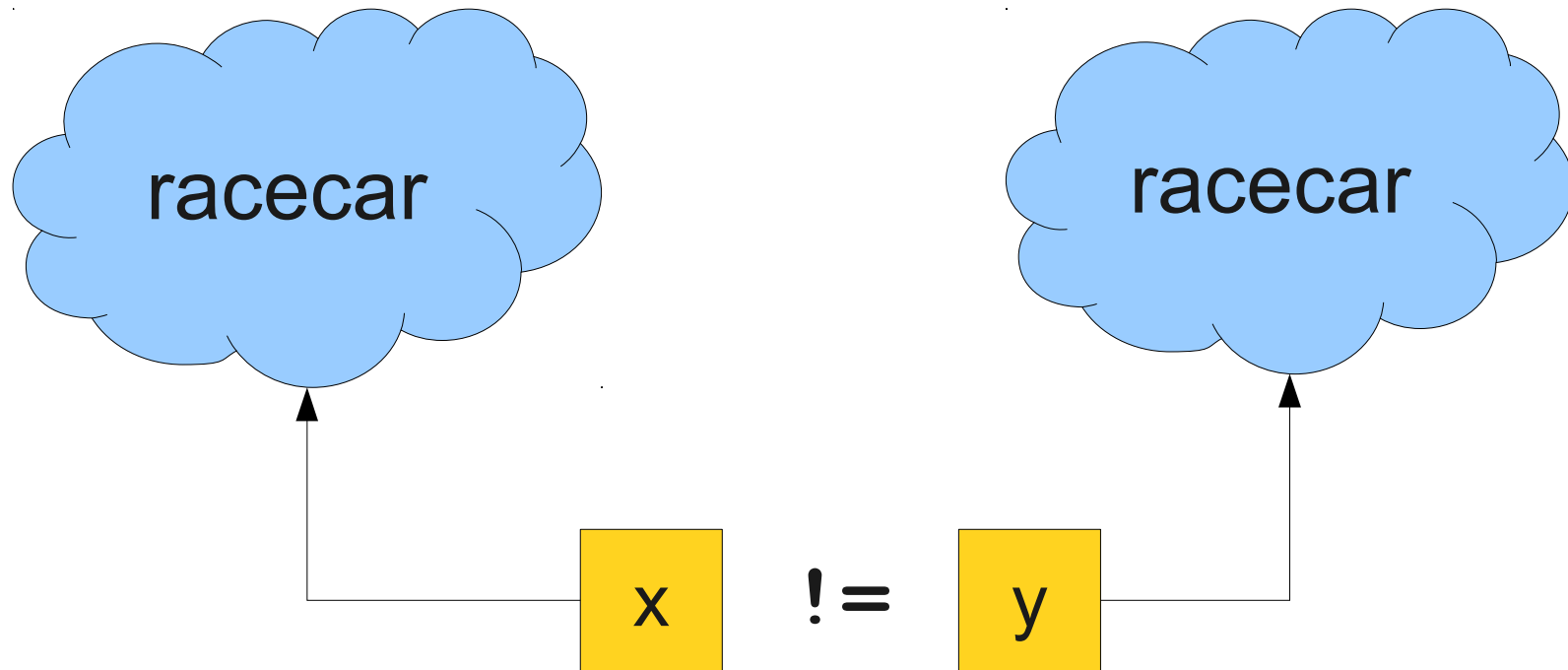
- A **palindrome** is a string that reads the same forwards and backwards.
- For example:
 - Racecar
 - Kayak
 - Mr. Owl ate my metal worm.
 - Go hang a salami! I'm a lasagna hog.

Checking for Palindromes

What Went Wrong?

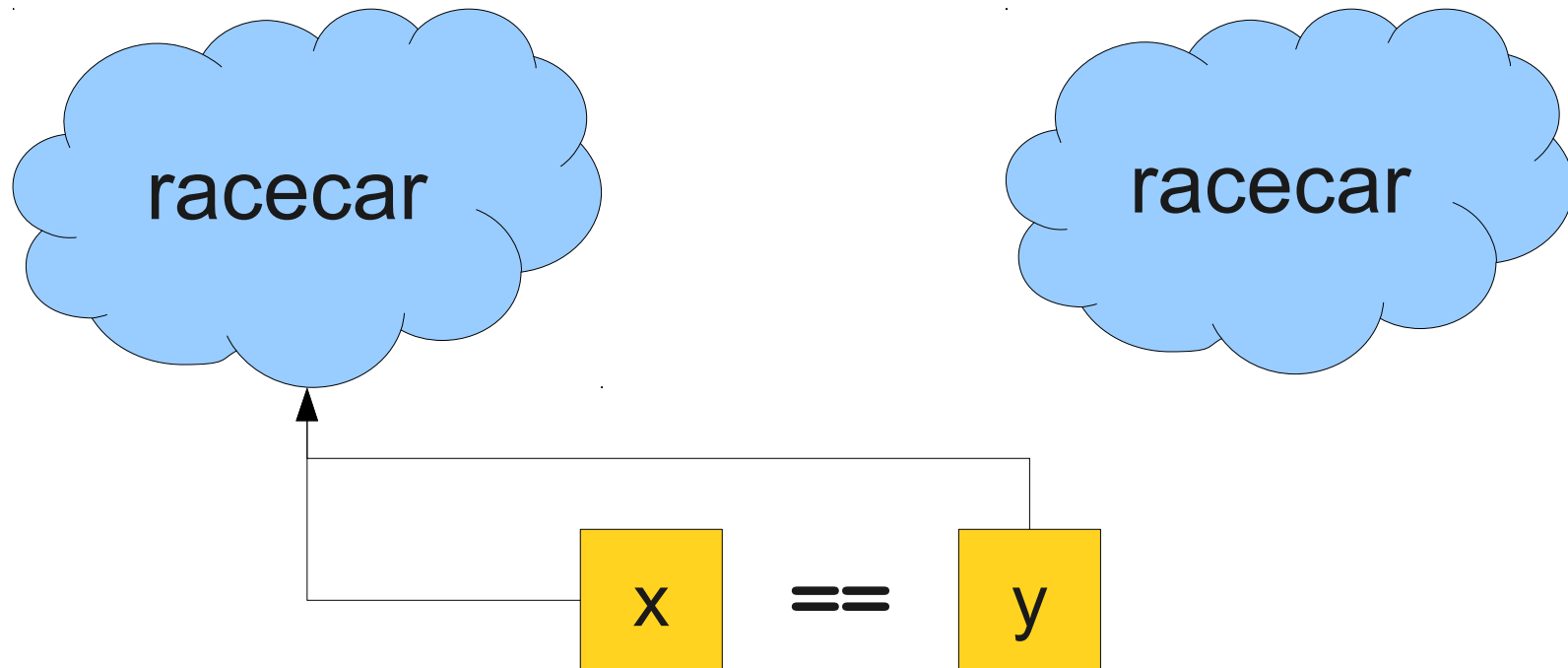
The == Operator

- When applied to objects, the == operator reports whether the two objects are the same object, not whether the *values* of those objects are equal.



The == Operator

- When applied to objects, the == operator reports whether the two objects are the same object, not whether the *values* of those objects are equal.



Comparing Strings for Equality

- To determine if two strings are equal, use the `.equals()` method:

```
String s1 = "racecar";  
String s2 = reverseString(s1);  
if (s1.equals(s2)) {  
    /* ... s1 and s2 are equal ... */  
}
```