Text Processing
Chris Piech
CS106A, Stanford University
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*. 

Piech, CS106A, Stanford University
Revisit our Programs

```java
// 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);
}
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);
}

// run
showLeftArrow
leftArrow

centerImage
img
x 327 y 104

heap

x = 327, y = 104
</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

Chris’ Favorite Program

Bright Simons

Piech, CS106A, Stanford University
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
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.

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

How it is actually stored

Length: 6

Hello!

text.charAt(index)
All characters in a string have an index.

You can access a character in the sequence via its index.
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:

```java
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.
numLs
How are characters represented?
The variable type `char`

- The primitive type `char` represents a single character or glyph.

- Some examples:
  ```java
  char letterA = 'A';
  char plus = '+'
  char zero = '0';
  char space = ' ';
  char newLine = '\n'; // special
  char first = text.charAt(0);
  ```
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!
The letter A, for example, has the ASCII value 65

* This is only the first half of the table
‘A’ -> ‘Z’ are sequential.
‘a’ -> ‘z’ are sequential.
‘0’ -> ‘9’ are sequential.
### Useful Character methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static boolean isDigit(char ch)</code></td>
<td>Determines if the specified character is a digit.</td>
</tr>
<tr>
<td><code>static boolean isLetter(char ch)</code></td>
<td>Determines if the specified character is a letter.</td>
</tr>
<tr>
<td><code>static boolean isLetterOrDigit(char ch)</code></td>
<td>Determines if the specified character is a letter or a digit.</td>
</tr>
<tr>
<td><code>static boolean isLowerCase(char ch)</code></td>
<td>Determines if the specified character is a lowercase letter.</td>
</tr>
<tr>
<td><code>static boolean isUpperCase(char ch)</code></td>
<td>Determines if the specified character is an uppercase letter.</td>
</tr>
<tr>
<td><code>static boolean isWhitespace(char ch)</code></td>
<td>Determines if the specified character is whitespace (spaces and tabs).</td>
</tr>
<tr>
<td><code>static char toLowerCase(char ch)</code></td>
<td>Converts <code>ch</code> to its lowercase equivalent, if any. If not, <code>ch</code> is returned unchanged.</td>
</tr>
<tr>
<td><code>static char toUpperCase(char ch)</code></td>
<td>Converts <code>ch</code> to its uppercase equivalent, if any. If not, <code>ch</code> is returned unchanged.</td>
</tr>
</tbody>
</table>
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
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

Hello!
Reversing a String

Hello!

!
Reversing a String

Hello!

!
Reversing a String

Hello!

! o
Reversing a String

Hello!

! o
Reversing a String

Hello!

! o l
Reversing a String

Hello!

! o l
Reversing a String

Hello!

! o l l
Reversing a String

Hello!

! o l l
Reversing a String

Hello!

!ollel
Reversing a String

Hello!

!ollelle
Reversing a String

Hello!

!olleH
Reversing a String

Hello!

Hello!
Reversing a String

Hello!
Reversing a String

Hello!

H
Reversing a String

Hello!

eH
Reversing a String

Hello!

decode

Hello!

decode

Hello!
Reversing a String

Hello!

olleH
Reversing a String

Hello!

olleH
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;
    }

    String str = readLine("Enter a string: ");
    String rev = reverseString(str);
    println(str + " spelled backwards is " + rev);
}

This program reverses a string.
Enter a string: STRESSED
STRESSED spelled backwards is DESSERTS
### Useful String methods

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

*Using portions of slides by Eric Roberts*
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 `==`
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 radial, 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?

Bright Simons