Building Java Programs

Chapter 4
Lecture 4-3: Strings, char

reading: 3.3, 4.3
I was fascinated by locks as a kid. I loved how they turned information and patterns into physical strength. Why does my script keep dying?

And a lock invites you to try to open it. It's the hacker instinct. Only your ignorance stands in the way. Wait, it's passing bad strings.

I admired Harry Houdini, how he could open any lock and free himself from any restraint. Ah - Bash is parsing the spaces.

Sure, some of it was fakery and showmanship. But I still wonder how he so consistently escaped handcuffs. Backslashes? Huh? Never mind.
Strings

• **string**: An object storing a sequence of text characters.
  • Unlike most other objects, a String is not created with `new`.

    ```java
    String name = "text";
    String name = expression;
    ```

• Examples:

    ```java
    String name = "Marla Singer";
    int x = 3;
    int y = 5;
    String point = "(" + x + ", " + y + ")";
    ```
Objects (usage)

- **object**: An entity that contains data and behavior.
  - **data**: variables inside the object
  - **behavior**: methods inside the object
    - You interact with the methods; the data is hidden in the object.
    - A **class** is a type of objects.

- Constructing (creating) an object:
  
  ```
  Type objectName = new Type(parameters);
  ```

- Calling an object's method:
  ```
  objectName.methodName(parameters);
  ```
Indexes

- Characters of a string are numbered with 0-based *indexes*:

```java
String name = "Ultimate";
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>U</td>
<td>l</td>
<td>t</td>
<td>i</td>
<td>m</td>
<td>a</td>
<td>t</td>
<td>e</td>
</tr>
</tbody>
</table>

- First character's index : 0
- Last character's index : 1 less than the string's length
- The individual characters are values of type `char` (seen later)
# String methods

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>indexOf(str)</code></td>
<td>index where the start of the given string appears in this string (-1 if not found)</td>
</tr>
<tr>
<td><code>length()</code></td>
<td>number of characters in this string</td>
</tr>
<tr>
<td><code>substring(index1, index2)</code></td>
<td>the characters in this string from <code>index1</code> (inclusive) to <code>index2</code> (exclusive); if <code>index2</code> is omitted, grabs till end of string</td>
</tr>
<tr>
<td><code>toLowerCase()</code></td>
<td>a new string with all lowercase letters</td>
</tr>
<tr>
<td><code>toUpperCase()</code></td>
<td>a new string with all uppercase letters</td>
</tr>
</tbody>
</table>

- These methods are called using the dot notation:

```java
String starz = "Yeezy & Hova";
System.out.println(starz.length());  // 12
```
String method examples

// index  012345678901
String s1 = "Stuart Reges";
String s2 = "Marty Stepp";
System.out.println(s1.length());   // 12
System.out.println(s1.indexOf("e"));   // 8
System.out.println(s1.substring(7, 10));  // "Reg"

String s3 = s2.substring(1, 7);
System.out.println(s3.toLowerCase());  // "arty s"

• Given the following string:

// index 0123456789012345678901
String book = "Building Java Programs";

• How would you extract the word "Java"?
Modifying strings

- Methods like `substring` and `toLowerCase` build and return a new string, rather than modifying the current string.

```java
String s = "Aceyalone";
s.toUpperCase();
System.out.println(s);  // Aceyalone
```

- To modify a variable's value, you must reassign it:

```java
String s = "Aceyalone";
s = s.toUpperCase();
System.out.println(s);  // ACEYALONE
```
Name border

- Prompt the user for full name

- Draw out the pattern to the left

- This should be resizeable. Size 1 is shown and size 2 would have the first name twice followed by last name twice
Strings as user input

- **Scanner's next** method reads a word of input as a String.

  ```java
  Scanner console = new Scanner(System.in);
  System.out.print("What is your name? ");
  String name = console.next();
  name = name.toUpperCase();
  System.out.println(name + " has " + name.length() + " letters and starts with " + name.substring(0, 1));
  
  Output:
  What is your name? **Nas**
  NAS has 3 letters and starts with N
  ```

- **The nextLine** method reads a line of input as a String.

  ```java
  System.out.print("What is your address? ");
  String address = console.nextLine();
  ```
The `equals` method

- Objects are compared using a method named `equals`.

  ```java
  Scanner console = new Scanner(System.in);
  System.out.print("What is your name? ");
  String name = console.next();
  if (name.equals("Lance")) {
    System.out.println("Pain is temporary.");
    System.out.println("Quitting lasts forever.");
  }
  ```

- Technically this is a method that returns a value of type `boolean`, the type used in logical tests.
String test methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals(\texttt{str})</td>
<td>whether two strings contain the same characters</td>
</tr>
<tr>
<td>equalsIgnoreCase(\texttt{str})</td>
<td>whether two strings contain the same characters, ignoring upper vs. lower case</td>
</tr>
<tr>
<td>startsWith(\texttt{str})</td>
<td>whether one contains other's characters at start</td>
</tr>
<tr>
<td>endsWith(\texttt{str})</td>
<td>whether one contains other's characters at end</td>
</tr>
<tr>
<td>contains(\texttt{str})</td>
<td>whether the given string is found within this one</td>
</tr>
</tbody>
</table>

```java
String name = console.next();
if(name.endsWith("Kweli")) {
    System.out.println("Pay attention, you gotta listen to hear.");
} else if(name.equalsIgnoreCase("NaS")) {
    System.out.println("I never sleep 'cause sleep is the cousin of death.");
}
```
Strings question

- Write a program that reads two people's first names and suggests a name for their child

Example Output:

Parent 1 first name? Danielle
Parent 2 first name? John
Child Gender? f
Suggested baby name: JODANI

Parent 1 first name? Danielle
Parent 2 first name? John
Child Gender? Male
Suggested baby name: DANIJO
The `charAt` method

- The **chars** in a **String** **can be accessed using the `charAt` method**.

```java
String food = "cookie";
char firstLetter = food.charAt(0);   // 'c'
System.out.println(firstLetter + " is for " + food);
System.out.println("That's good enough for me!");
```

- You can use a **for** loop to print or examine each character.

```java
String major = "CSE";
for (int i = 0; i < major.length(); i++) {
    char c = major.charAt(i);
    System.out.println(c);
}
```

Output:
```
C
S
E
```
Type char

- **char**: A primitive type representing single characters.
  - Each character inside a **String** is stored as a **char** value.
  - Literal **char** values are surrounded with apostrophe (single-quote) marks, such as 'a' or '4' or '\n' or '\'

- It is legal to have variables, parameters, returns of type **char**

```
char letter = 'S';
System.out.println(letter); // S
```

- **char** values can be concatenated with strings.

```
char initial = 'P';
System.out.println(initial + " Diddy"); // P Diddy
```
char vs. String

- "h" is a String
  'h' is a char (the two behave differently)

- String is an object; it contains methods

  ```java
  String s = "h";
  s = s.toUpperCase(); // 'H'
  int len = s.length(); // 1
  char first = s.charAt(0); // 'H'
  ```

- char is primitive; you can't call methods on it

  ```java
  char c = 'h';
  c = c.toUpperCase(); // ERROR: "cannot be dereferenced"
  ```

- What is `s + 1`?  What is `c + 1`?
- What is `s + s`?  What is `c + c`?
char vs. int

- All char values are assigned numbers internally by the computer, called ASCII values.

  - Examples:
    'A' is 65,  'B' is 66,  ' ' is 32
    'a' is 97,  'b' is 98,  '*' is 42

  - Mixing char and int causes automatic conversion to int.
    'a' + 10 is 107,  'A' + 'A' is 130

  - To convert an int into the equivalent char, type-cast it.
    (char) ('a' + 2) is 'c'
Comparing char values

- You can compare char values with relational operators:
  
  'a' < 'b' and 'X' == 'X' and 'Q' != 'q'

- An example that prints the alphabet:

  ```java
  for (char c = 'a'; c <= 'z'; c++) {
      System.out.print(c);
  }
  ```

- You can test the value of a string's character:

  ```java
  String word = console.next();
  if (word.charAt(word.length() - 1) == 's') {
      System.out.println(word + " is plural.");
  }
  ```
String/char question

• A Caesar cipher is a simple encryption where a message is encoded by shifting each letter by a given amount.
  • e.g. with a shift of 3, A → D, H → K, X → A, and Z → C

• Write a program that reads a message from the user and performs a Caesar cipher on its letters:

  Your secret message: Brad thinks Angelina is cute
  Your secret key: 3
  The encoded message: eudg wklqnv dqjholqd lv fxwh
import java.util.*;

public class SecretMessage {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Your secret message: ");
        String message = console.nextLine();
        message = message.toLowerCase();

        System.out.print("Your secret key: ");
        int key = console.nextInt();

        encode(message, key);
    }

    ...
// This method encodes the given text string using a Caesar cipher, shifting each letter by the given number of places.
public static void encode(String text, int shift) {
    System.out.print("The encoded message: ");
    for (int i = 0; i < text.length(); i++) {
        char letter = text.charAt(i);

        // shift only letters (leave other characters alone)
        if (letter >= 'a' && letter <= 'z') {
            letter = (char) (letter + shift);
        }

        // may need to wrap around
        if (letter > 'z') {
            letter = (char) (letter - 26);
        } else if (letter < 'a') {
            letter = (char) (letter + 26);
        }

        System.out.print(letter);
    }
    System.out.println();
}