Building Java Programs

Chapter 4
Lecture 4-3: Strings; char

reading: 3.3, 4.3
Strings

reading: 3.3
Objects

- **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:
  ```java
  Type objectName = new Type(parameters);
  ```

- Calling an object's method:
  ```java
  objectName.methodName(parameters);
  ```
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 (with String value);
  ```

- Examples:
  ```java
  String names = "Alice and Bob");
  int x = 3;
  int y = 5;
  String point = "(" + x + ", " + y + ")";
  ```
Indexes

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

  ```java
  String name = "M. Mouse";
  ```

<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>M</td>
<td>.</td>
<td>M</td>
<td>o</td>
<td>u</td>
<td>s</td>
<td>e</td>
<td></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> or <code>substring(index1)</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 = "Prince vs. Michael";
System.out.println(starz.length()); // 18
```
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 = "Mumford & Sons";
  s.toUpperCase();
  System.out.println(s); // Mumford & Sons
  ```

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

  ```java
  String s = "Mumford & Sons";
  s = s.toUpperCase();
  System.out.println(s); // MUMFORD & SONS
  ```
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? **Bono**
BONO has 4 letters and starts with B

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

```java
System.out.print("What is your address? ");
String address = console.nextLine();
```
Name border

- Prompt the user for full name
- Draw out the pattern to the left
- This should be resizable. Size 1 is shown and size 2 would have the first name twice followed by last name twice
Strings question

- Write a program that outputs “The Name Game” with a person’s first and last name.

Example Output:

What is your name? James Joyce

James, James, bo-bames
Banana-fana fo-fames
Fee-fi-mo-mames
JAMES!

Joyce, Joyce, bo-boyce
Banana-fana fo-foyece
Fee-fi-mo-moyce
JOYCE!
// This program prints "The Name Game".
import java.util.*;

public class TheNameGame {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = console.nextLine();

        int spaceIndex = name.indexOf(" ");
        String firstName = name.substring(0, spaceIndex);
        String lastName = name.substring(spaceIndex + 1);

        singSong(firstName);
        singSong(lastName);
    }
}

Strings answer
public static void singSong(String name) {
    System.out.println();
    String allButLast = name.substring(1);
    System.out.println(name + ', ' + name + ', bo-b' + allButLast);
    System.out.println("Banana-fana fo-f" + allButLast);
    System.out.println("Fee-fi-mo-m" + allButLast);
    System.out.println(name.toUpperCase() + '!');
}
Comparing strings

- Relational operators such as `<` and `==` fail on objects.

```java
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name == "Barney") {
    System.out.println("I love you, you love me, ");
    System.out.println("We’re a happy family!");
}
```

- This code will compile, but it will not print the song.
- `==` compares objects by *references* (seen later), so it often gives false even when two Strings have the same letters.
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("Barney")) {
    System.out.println("I love you, you love me, ");
    System.out.println("We’re a happy family!");
}
```

- 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><code>equals(String str)</code></td>
<td>whether two strings contain the same characters</td>
</tr>
<tr>
<td><code>equalsIgnoreCase(String str)</code></td>
<td>whether two strings contain the same characters, ignoring upper vs. lower case</td>
</tr>
<tr>
<td><code>startsWith(String str)</code></td>
<td>whether one contains other's characters at start</td>
</tr>
<tr>
<td><code>endsWith(String str)</code></td>
<td>whether one contains other's characters at end</td>
</tr>
<tr>
<td><code>contains(String str)</code></td>
<td>whether the given string is found within this one</td>
</tr>
</tbody>
</table>

String name = console.nextLine();
if (name.startsWith("Dr.")) {
    System.out.println("Will you marry me?");
} else if (name.equalsIgnoreCase("buTteRs")) {
    System.out.println("You're grounded, young man!");
}

String documentation: [http://docs.oracle.com/javase/7/docs/api/java/lang/String.html](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html)
Strings question

- Write a program that reads two people's first names and suggests a name for their child.
  - The suggestion is the concatenation of the first halves of both names.

Example Output:
Parents 1 first name? Danielle
Parents 2 first name? John
Child gender? f
Suggested baby name: JODANI

Parents 1 first name? Danielle
Parents 2 first name? John
Child gender? Male
Suggested baby name: DANIJO
/ *Suggests a baby name based on parents' names.*

```java
import java.util.*;

public class BabyNamer {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("Parent 1 first name? ");
        String name1 = s.next();
        System.out.print("Parent 2 first name? ");
        String name2 = s.next();
        System.out.print("Child gender? ");
        String gender = s.next();

        System.out.println("Suggested name: 
                             " +
                            suggestChildName(gender, name1, name2).toUpperCase());
    }
}
```
/ Strings answer (cont.)

... 

// Return the first half of the given name.
public static String getHalfName(String name) {
    int halfIndex = name.length() / 2;
    return name.substring(0, halfIndex);
}

// Suggests a child's name (for a given gender) for parents with the given names.
public static String suggestChildName(String gender, String name1, String name2) {
    String halfName1 = getHalfName(name1);
    String halfName2 = getHalfName(name2);
    String name;
    if (gender.toLowerCase().startsWith("f")) {
        name = halfName1 + halfName2;
    } else {
        name = halfName2 + halfName1;
    }
    return name;
}

}
Another Strings question

- Prompt the user for two words and report whether they:
  - "rhyme" (end with the same last two letters)
  - alliterate (begin with the same letter)

- Example output: (run #1)
  Type two words: car STAR
  They rhyme!

  (run #2)
  Type two words: bare bear
  They alliterate!

  (run #3)
  Type two words: sell shell
  They alliterate!
  They rhyme!

  (run #4)
  Type two words: extra strawberry
// Determines whether two words rhyme and/or alliterate.
import java.util.*;

public class Rhyme {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Type two words: ");
        String word1 = console.next().toLowerCase();
        String word2 = console.next().toLowerCase();
        printIfRhyme(word1, word2);
        printIfAlliterate(word1, word2);
    }

    // print if two words "rhyme" (i.e., end with the same two letters)
    public static void printIfRhyme(String word1, String word2) {
        if (word2.length() >= 2 &&
            word1.endsWith(word2.substring(word2.length() - 2))) {
            System.out.println("They rhyme!");
        }
    }

    // print if two alliterate
    public static void printIfAlliterate(String word1, String word2) {
        if (word1.startsWith(word2.substring(0, 1))) {
            System.out.println("They alliterate!");
        }
    }
}
char

reading: 4.3
Type **char**

- **char**: A primitive type representing single characters.
  - **A String** is stored internally as an array of **char**

```
String s = "nachos";
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>'n'</td>
<td>'a'</td>
<td>'c'</td>
<td>'h'</td>
<td>'o'</td>
<td>'s'</td>
</tr>
</tbody>
</table>

- It is legal to have variables, parameters, returns of type **char**
  - surrounded with apostrophes: 'a' or '4' or '
  - or ''

```java
char initial = 'J';
System.out.println(initial); // J
System.out.println(initial + " Joyce"); // J Joyce
```
The `charAt` method

- The **chars** in a **String** can be accessed using the `charAt` method.
- It accepts an **int** index parameter and returns the **char** at that index.

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

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

```java
String major = "CSE";
for (int i = 0; i < major.length(); i++) {  // output:
    char c = major.charAt(i);  // C
    System.out.println(c);  // S
}  // E
```
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.");
  }
  ```
char vs. String

• "h" is a String, but 'h' is a char (they are different)

• A 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'
```

• A char is primitive; you can't call methods on it.

```java
char c = 'h';
c = c.toUpperCase(); // ERROR
s = s.charAt(0).toUpperCase(); // ERROR
```

• What is s + 1? What is c + 1?
• What is s + s? What is c + c?
char vs. int

- Each char is mapped to an integer value internally
  - Called an ASCII value

' A ' is 65  ' B ' is 66  ' ' is 32
' a ' is 97  ' b ' is 98  ' * ' is 42

- Doing "math" on a char 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 '
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: Computer science is awesome
Your secret key: 3
The encoded message: frpsxwhu vflhqfh lv dzhvrph
Strings answer 1

// This program reads a message and a secret key from the user and
// encrypts the message using a Caesar cipher, shifting each letter.

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);
    }

    public static String encode(String message, int key) {
        // Implement Caesar cipher encoding logic here
        return encodedMessage;
    }
}

...
// 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();
}
(Optional) printf

reading: 4.3
Formatting text with `printf`

```
System.out.printf("format string", parameters);
```

- A format string can contain *placeholders* to insert parameters:
  - `%d` integer
  - `%f` real number
  - `%s` string
    - these placeholders are used instead of + concatenation

- Example:
  ```java
  int x = 3;
  int y = -17;
  System.out.printf("x is %d and y is %d!\n", x, y);
  // x is 3 and y is -17!
  ```

- `printf` does not drop to the next line unless you write `\n`
printf width

- `${Wd}`: integer, `W` characters wide, right-aligned
- `${-Wd}`: integer, `W` characters wide, left-aligned
- `${Wf}`: real number, `W` characters wide, right-aligned
- ...

```java
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.printf("%4d", (i * j));
    }
    System.out.println();  // to end the line
}
```

Output:

```
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
```
printf precision

- \%.Df real number, rounded to D digits after decimal
- \%W.Df real number, W chars wide, D digits after decimal
- %W.Df real number, W wide (left-align), D after decimal

```java
double gpa = 3.253764;
System.out.printf("your GPA is %.1f\n", gpa);
System.out.printf("more precisely: %8.3f\n", gpa);
```

Output:

```
your GPA is 3.3
more precisely: 3.254
```

3

8
Modify our Receipt program to better format its output.

Display results in the format below, with 2 digits after .

Example log of execution:

How many people ate? 4
Person #1: How much did your dinner cost? 20.00
Person #2: How much did your dinner cost? 15
Person #3: How much did your dinner cost? 25.0
Person #4: How much did your dinner cost? 10.00

Subtotal: $70.00
Tax: $5.60
Tip: $10.50
Total: $86.10
...  

// Calculates total owed, assuming 8% tax and 15% tip 
public static void results(double subtotal) {
    double tax = subtotal * .08;
    double tip = subtotal * .15;
    double total = subtotal + tax + tip;

    // System.out.println("Subtotal: $" + subtotal);
    // System.out.println("Tax: $" + tax);
    // System.out.println("Tip: $" + tip);
    // System.out.println("Total: $" + total);

    System.out.printf("Subtotal: $%.2f\n", subtotal);
    System.out.printf("Tax: $%.2f\n", tax);
    System.out.printf("Tip: $%.2f\n", tip);
    System.out.printf("Total: $%.2f\n", total);
}
}