Java 1

Sayed Ahmad Sahim

Lecture 05-Strings

Objects and classes

- **object:** An entity that contains:
  - *data* (variables), and
  - *behavior* (methods).

- **class**: A program, or a type of objects.

- Examples:
  - The class `String` represents objects that store text.
  - The class `DrawingPanel` represents graphical window objects.
  - The class `Scanner` represents objects that read information from the keyboard, files, and other sources.

Strings

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

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

  - Examples:

```
String name = "Marla Singer";

int x = 3;
int y = 5;
String point = "(" + x + ", " + y + ")";
```

Indexes

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

```
String name = "P. Diddy";
```

index	0	1	2	3	4	5	6	7
char	P	.		D	i	d	d	y

  - The first character's index is always 0
  - The last character's index is 1 less than the string's length

  - The individual characters are values of type `char` (seen later)

`String` methods

Method name	Description
`indexOf(`**`str`**`)`	index where the start of the given string appears in this string (-1 if it is not there)
`length()`	number of characters in this string

`substring(`**`index1`**`, `**`index2`**`)`
or
`substring(`**`index1`**`)`
	the characters in this string from *index1* (inclusive) to *index2* (exclusive);
	if *index2* omitted, grabs till end of string

`toLowerCase()`	a new string with all lowercase letters
`toUpperCase()`	a new string with all uppercase letters

- These methods are called using the dot notation:

```
String gangsta = "Dr. Dre";
System.out.println(gangsta.length());   // 7
```

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(2, 8);
System.out.println(s3.toLowerCase());     // "rty st"
```

- Given the following string:

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

  - How would you extract the word "Java" ?
  - How would you extract the first word from any string?

Modifying strings

- Methods like `substring`, `toLowerCase`, etc. create/return a new string, rather than modifying the current string.

```
String s = "lil bow wow";
s.toUpperCase();
System.out.println(s);   // lil bow wow
```

- To modify a variable, you must reassign it:

```
String s = "lil bow wow";
s = s.toUpperCase();
System.out.println(s);   // LIL BOW WOW
```

Strings as parameters

```
public class StringParameters {
    public static void main(String[] args) {
        sayHello("Marty");

        String teacher = "Helene";
        sayHello(teacher);
    }
    public static void sayHello(String name) {
        System.out.println("Welcome, " + name);
    }
}
```

Output:

```
Welcome, Marty
Welcome, Helene
```

Strings as user input

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

```
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? Ahmad
Ahmad has 5 letters and starts with A
```

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

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

Comparing strings

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

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name == "Ahmad") {
    System.out.println("I like you, you like 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.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name.equals("Ahmad")) {
    System.out.println("I like you, you like 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

Method	Description
`equals(`**`str`**`)`	whether two strings contain the same characters
`equalsIgnoreCase(`**`str`**`)`	whether two strings contain the same characters, ignoring upper vs. lower case
`startsWith(`**`str`**`)`	whether one contains other's characters at start
`endsWith(`**`str`**`)`	whether one contains other's characters at end
`contains(`**`str`**`)`	whether the given string is found within this one

```
String name = console.next();
if (name.startsWith("Dr.")) {
    System.out.println("Are you single?");
} else if (name.equalsIgnoreCase("Javid")) {
    System.out.println("I need your TPS reports.");
}
```

Strings question

- Write a program that reads a person's name and converts it into a "gangsta name."

Output (run 1):

```
Type your name, playa: Peter Griffin
(M)ale or (F)emale? m
Your gangsta name is "P. GRIFFIN Daddy Peter-izzle"
```

Output (run 2):

```
Type your name, playa: Marge Simpson
(M)ale or (F)emale? F
Your gangsta name is "M. SIMPSON Goddess Marge-izzle"
```

Strings answer

```
// This program prints your "gangsta" name.
import java.util.*;

public class GangstaName {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Type your name, playa: ");
        String name = console.nextLine();

        System.out.print("(M)ale or (F)emale: ");
        String gender = console.next();

        // split name into first/last name and initials
        String first = name.substring(0, name.indexOf(" "));
        String last = name.substring(name.indexOf(" ") + 1);
        last = last.toUpperCase();
        String fInitial = first.substring(0, 1);

        String title;
        if (gender.equalsIgnoreCase("m")) {
            title = "Daddy";
        } else {
            title = "Goddess";
        }

        System.out.println("Your gangsta name is \"" + fInitial + ". "
                + last + " " + title + " " + first + "-izzle\"");
    }
}
```

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
```

The charAt method

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

```
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.

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

Output:

```
C
S
E
```

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'

char vs. String

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

- String is an object; it contains methods

```
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

```
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 ?

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:

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

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

```
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
```

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);
    }

    ...
```

Strings answer 2

```
// 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();
}
}
```

The Random class

- A Random object generates pseudo-random* numbers.
  - Class Random is found in the java.util package.

```
import java.util.*;
```

Method name	Description
nextInt()	returns a random integer
nextInt(**max**)	returns a random integer in the range [0, *max*) in other words, 0 to *max*-1 inclusive
nextDouble()	returns a random real number in the range [0.0, 1.0)

  - Example:

```
Random rand = new Random();
int randomNumber = rand.nextInt(10);   // 0-9
```

Generating random numbers

- Common usage: to get a random number from 1 to *N*

```
int n = rand.nextInt(20) + 1;   // 1-20 inclusive
```

- To get a number in arbitrary range [*min*, *max*] inclusive:

```
nextInt(size of range) + min
```

  - where (**size of range**) is (**max - min + 1**)

- Example: A random integer between 4 and 10 inclusive:

```
int n = rand.nextInt(7) + 4;
```

Random questions

- Given the following declaration, how would you get:

```
Random rand = new Random();
```

  - A random number between 1 and 100 inclusive?

```
int random1 = rand.nextInt(100) + 1;
```

  - A random number between 50 and 100 inclusive?

```
int random2 = rand.nextInt(51) + 50;
```

  - A random number between 4 and 17 inclusive?

```
int random3 = rand.nextInt(14) + 4;
```

Random and other types

- `nextDouble` method returns a `double` between 0.0 - 1.0
  - Example: Get a random GPA value between 1.5 and 4.0:

```
double randomGpa = rand.nextDouble() * 2.5 + 1.5;
```

- Any set of possible values can be mapped to integers
  - code to randomly play Rock-Paper-Scissors:

```
int r = rand.nextInt(3);
if (r == 0) {
    System.out.println("Rock");
} else if (r == 1) {
    System.out.println("Paper");
} else {
    System.out.println("Scissors");
}
```

Random question

- Write a program that simulates rolling of two 6-sided dice until their combined result comes up as 7.

```
2 + 4 = 6
3 + 5 = 8
5 + 6 = 11
1 + 1 = 2
4 + 3 = 7
You won after 5 tries!
```

  - Modify the program to play 3 dice games using a method.

Random answer

```
// Rolls two dice until a sum of 7 is reached.
import java.util.*;

public class Dice {
    public static void main(String[] args) {
        Random rand = new Random();
        int tries = 0;

        int sum = 0;
        while (sum != 7) {
            // roll the dice once
            int roll1 = rand.nextInt(6) + 1;
            int roll2 = rand.nextInt(6) + 1;
            sum = roll1 + roll2;
            System.out.println(roll1 + " + " + roll2 + " = " + sum);
            tries++;
        }

        System.out.println("You won after " + tries + " tries!");
    }
}
```

Random question

- Write a multiplication tutor program.
  - Ask user to solve problems with random numbers from 1-20.
  - The program stops after an incorrect answer.

```
14 * 8 = 112
Correct!
5 * 12 = 60
Correct!
8 * 3 = 24
Correct!
5 * 5 = 25
Correct!
20 * 14 = 280
Correct!
19 * 14 = 256
Incorrect; the answer was 266

You solved 5 correctly
Last correct answer was 280
```

  - The last line should not appear if the user solves 0 correctly.

Random answer

```
import java.util.*;

// Asks the user to do multiplication problems and scores them.
public class MultiplicationTutor {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        Random rand = new Random();

        // fencepost solution - pull first question outside of loop
        int correct = 0;
        int last = askQuestion(console, rand);
        int lastCorrect = 0;

        // loop until user gets one wrong
        while (last > 0) {
            lastCorrect = last;
            correct++;
            last = askQuestion(console, rand);
        }

        System.out.println("You solved " + correct + " correctly");
        if (correct > 0) {
            System.out.println("Last correct answer was " + lastCorrect);
        }
    }
    ...
```

Random answer 2

```
    ...

    // Asks the user one multiplication problem,
    // returning the answer if they get it right and 0 if not.
    public static int askQuestion(Scanner console, Random rand) {
        // pick two random numbers between 1 and 20 inclusive
        int num1 = rand.nextInt(20) + 1;
        int num2 = rand.nextInt(20) + 1;

        System.out.print(num1 + " * " + num2 + " = ");
        int guess = console.nextInt();
        if (guess == num1 * num2) {
            System.out.println("Correct!");
            return num1 * num2;
        } else {
            System.out.println("Incorrect; the correct answer was " +
                               (num1 * num2));
            return 0;
        }
    }
}
```