Building Java Programs

A Back to Basics Approach

CS 210

CHAPTER 3

INTRODUCTION TO
PARAMETERS AND OBJECTS

Please download the PPT, and use Slide Show for a better viewing experience

Winnie Li

Portions Copyright 2020 by Pearson Education

Topics will be covered

CS 210

- Parameters
- Return Values
- Objects and Classes
- String
- Interactive Programs with scanner

Portions Copyright 2020 by Pearson Education

Parameters

CS 210

PARAMETERS
COMMON ERROR
STRINGS AS PARAMETERS
STAR EXAMPLE
VALUE SEMANTICS

Portions Copyright 2020 by Pearson Education

Redundant figures

- Consider the task of printing the following lines/boxes:

```
*************

*******

*****************************

**********
*        *
**********

*****
*   *
*   *
*****
```

Stars solution 1 – redundant

CS 210

```
public class Stars1 {
    public static void main(String[] args) {
        lineOf13();
        lineOf7();
        lineOf35();
        box10x3();
        box5x4();
    }

    public static void lineOf13() {
        for (int i = 1; i <= 13; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    public static void lineOf7() {
        for (int i = 1; i <= 7; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    public static void lineOf35() {
        for (int i = 1; i <= 35; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
    ...
```

- This code is redundant.
- Would variables help? Would constants help?
- What is a better solution?

Portions Copyright 2020 by Pearson Education

A better solution

CS 210

- Generalized tasks:
  - `drawLine` - A method to draw a line of any number of stars
  - `drawBox` - A method to draw a box of any size

- We really want:

```
public static void drawLine() {
    for (int i = 1; i <= N; i++) {
        System.out.print("*");
    }
    System.out.println();
}
```

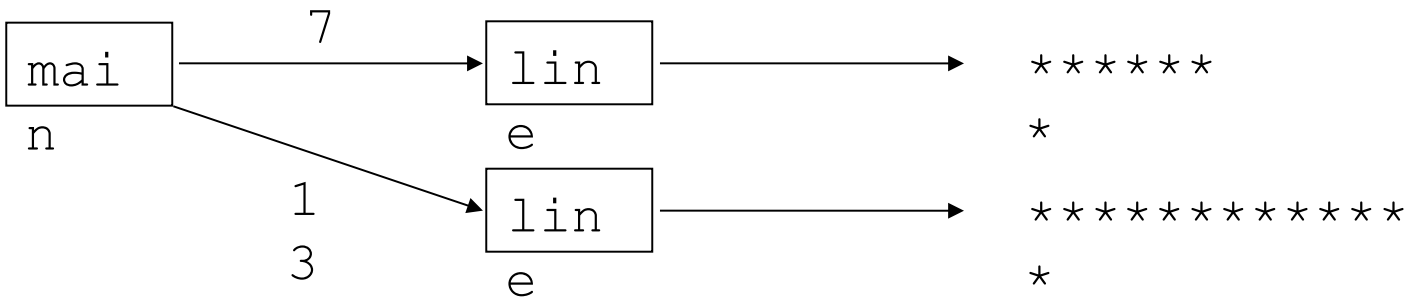
- Desired properties for `drawLine`:
  - Can call `drawLine` method multiple times, producing differently sized lines
  - Size of line printed is determined by the caller of the command (i.e. "Print me a line of size 7")

Portions Copyright 2020 by Pearson Education

Parameterization

CS 210

- **parameter**: A value passed to a method by its caller.
  - Instead of `lineOf7`, `lineOf13`, write `line` to draw any length.
    - When *declaring* the method, we will state that it requires a parameter for the number of stars.
    - When *calling* the method, we will specify how many stars to draw.



Portions Copyright 2020 by Pearson Education

Declaring a parameter

CS 210

Stating that a method requires a parameter in order to run

```
public static void name ( type name ) {
    statement(s);
}
```

- Example:

```
public static void sayPassword(int code) {
    System.out.println("The password is: " + code);
}
```

  - When `sayPassword` is called, the caller must specify the integer code to print.

Portions Copyright 2020 by Pearson Education

Passing a parameter

CS 210

Calling a method and specifying values for its parameters

name (**expression**);

- Example:

```
public static void main(String[] args) {
    sayPassword(42);
    sayPassword(12345);
}
```

Output:

```
The password is 42
The password is 12345
```

Portions Copyright 2020 by Pearson Education

Parameters and loops

CS 210

- A parameter can guide the number of repetitions of a loop.

```
public static void main(String[] args) {
    chant(3);
}

public static void chant(int times) {
    for (int i = 1; i <= times; i++) {
        System.out.println("Study Hard!!!");
    }
}
```

Output:

Portions Copyright 2020 by Pearson Education

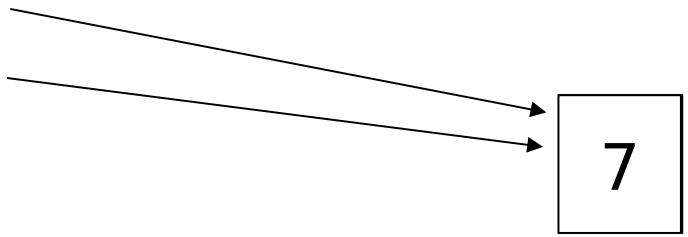
How parameters are passed

CS 210

- When the method is called:
  - The value is stored into the parameter variable.
  - The method's code executes using that value.

```
public static void main(String[] args) {
    chant(3);
    chant(7);
}

public static void chant(int times) {
    for (int i = 1; i <= times; i++) {
        System.out.println("Study Hard!!!");
    }
}
```



Common errors

CS 210

- If a method accepts a parameter, it is illegal to call it without passing any value for that parameter.

```
chant();        // ERROR: parameter value required
```

- The value passed to a method must be of the correct type.

```
chant(3.7);     // ERROR: must be of type int
```

Portions Copyright 2020 by Pearson Education

Stars solution 2 – using parameters


```
// Prints several lines of stars.
// Uses a parameterized method to remove redundancy.
public class Stars2 {
    public static void main(String[] args) {
        line(13);
        line(7);
        line(35);
    }

    // Prints the given number of stars plus a line break.
    public static void line(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

Portions Copyright 2020 by Pearson Education

Multiple parameters

CS 210

- A method can accept multiple parameters.
  - Parameters can be separated by “,”.
  - When calling it, you must pass values for each parameter.

- Declaration:

```
public static void name (type name, ..., type name) {
    statement(s);
}
```

- Call:

```
name (value, value, ..., value);
```

Multiple parameters example


```
public static void main(String[] args) {
    printNumber(4, 9);
    printNumber(17, 6);
    printNumber(8, 0);
    printNumber(0, 8);
}

public static void printNumber(int number, int count) {
    for (int i = 1; i <= count; i++) {
        System.out.print(number);
    }
    System.out.println();
}
```

- Output:

```
444444444
171717171717

00000000
```

Portions Copyright 2020 by Pearson Education

Stars solution 3 – using parameters

CS 210

```
// Prints several lines and boxes made of stars.
// Third version with multiple parameterized methods.

public class Stars3 {
    public static void main(String[] args) {
        line(13);
        line(7);
        line(35);
        System.out.println();
        box(10, 3);
        box(5, 4);
        box(20, 7);
    }

    // Prints the given number of stars plus a line break.
    public static void line(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
    ...
```

Portions Copyright 2020 by Pearson Education

Stars solution 3, cont'd.


```
    ...

    // Prints a box of stars of the given size.
    public static void box(int width, int height) {
        line(width);

        for (int line = 1; line <= height - 2; line++) {
            System.out.print("*");
            for (int space = 1; space <= width - 2; space++)
  {
                System.out.print(" ");
            }
            System.out.println("*");
        }

        line(width);
    }
}
```

Portions Copyright 2020 by Pearson Education

Strings

CS 210

- **string**: A sequence of text characters.

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

  - Examples:

```
String name = "Marla Singer";

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

Strings as parameters

CS 210

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

        String teacher = "Winnie";
        sayHello(teacher);
    }

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

Output:

```
Welcome, Bill
Welcome, Winnie
```

Portions Copyright 2020 by Pearson Education

Stars solution 4 – using string parameters

CS 210

```
// Prints several lines and boxes made of stars.
// Fourth version with String parameters.

public class Stars4 {
    public static void main(String[] args) {
        line(13);
        line(7);
        line(35);
        System.out.println();
        box(10, 3);
        box(5, 4);
        box(20, 7);
    }

    // Prints the given number of stars plus a line break.
    public static void line(int count) {
        repeat("*", count);
        System.out.println();
    }

    ...
```

Portions Copyright 2020 by Pearson Education

Stars solution, cont'd.


```
    ...

    // Prints a box of stars of the given size.
    public static void box(int width, int height) {
        line(width);

        for (int line = 1; line <= height - 2; line++) {
            System.out.print("*");
            repeat(" ", width - 2);
            System.out.println("*");
        }

        line(width);
    }

    // Prints the given String the given number of times.
    public static void repeat(String s, int times) {
        for (int i = 1; i <= times; i++) {
            System.out.print(s);
        }
    }
}
```

Portions Copyright 2020 by Pearson Education

Value semantics

CS 210

- **value semantics**: When primitive variables (`int`, `double`) are passed as parameters, their values are copied.
  - Modifying the parameter will not affect the variable passed in.

```
public static void main(String[] args) {
    int x = 23;
    strange(x);
    System.out.println("2. x = " + x);
    ...
}
public static void strange(int x) {
    x = x + 1;
    System.out.println("1. x = " + x);
}
```

Portions Copyright 2020 by Pearson Education

"Parameter Mystery" problem

CS 210

```
public class ParameterMystery {
    public static void main(String[] args) {
        int x = 9;
        int y = 2;
        int z = 5;

        mystery(z, y, x);

        mystery(y, x, z);
    }

    public static void mystery(int x, int z, int y) {
        System.out.println(z + " and " + (y - x));
    }
}
```

Values Passed in:

5	2	9

Output:

2 and 4

9 and 3

Value Semantics Exercise

CS 210

What output is produced by the following program?

```
public class MysterySoda {
    public static void main(String[] args) {
        String soda = "coke";
        String pop = "pepsi";
        String coke = "pop";
        String pepsi = "soda";
        String say = pop;

        carbonated(coke, soda, pop);
        carbonated(pop, pepsi, pepsi);
        carbonated("pop", pop, "koolaid");
        carbonated(say, "say", pop);
    }

  public static void carbonated(String coke, String soda, String pop){

     System.out.println("say " + soda + " not " + pop + " or " + coke);
  }
}
```

Portions Copyright 2020 by Pearson Education

Return values

CS 210

MATH CLASS
RETURN
TYPE CASTING
COMMON ERROR

Portions Copyright 2020 by Pearson Education

Java's `Math` class

CS 210

Method name	Description
`Math.abs(`*value*`)`	absolute value
`Math.ceil(`*value*`)`	rounds up
`Math.floor(`*value*`)`	rounds down
`Math.log10(`*value*`)`	logarithm, base 10
`Math.max(`*value1*`, `*value2*`)`	larger of two values
`Math.min(`*value1*`, `*value2*`)`	smaller of two values
`Math.pow(`*base*`, `*exp*`)`	*base* to the *exp* power
`Math.random()`	random `double` between 0 and 1
`Math.round(`*value*`)`	nearest whole number
`Math.sqrt(`*value*`)`	square root

`Math.sin(`*value*`)`
`Math.cos(`*value*`)`
`Math.tan(`*value*`)`
	sine/cosine/tangent of an angle in radians

`Math.toDegrees(`*value*`)` `Math.toRadians(`*value*`)`	convert degrees to radians and back

	Input Type	Output Type

`abs`
`min`
`max`
	`int` or `double`	`int`
		`double`

`round`	`int` or `double`	`int` ONLY!

`ceil`
`floor`
`log10`
`pow`
`random`
`sqrt`
	`int` or `double`	`double` ONLY!

Constant	Description
`Math.E`	2.7182818...
`Math.PI`	3.1415926...

Calling `Math` methods

CS 210

`Math.`**methodName**`(`**parameters**`)`

- Examples:

```
double squareRoot = Math.sqrt(121.0);
System.out.println(squareRoot);          // 11.0

int absoluteValue = Math.abs(-50);
System.out.println(absoluteValue);       // 50

System.out.println(Math.min(3, 7) + 2);  // 5
```

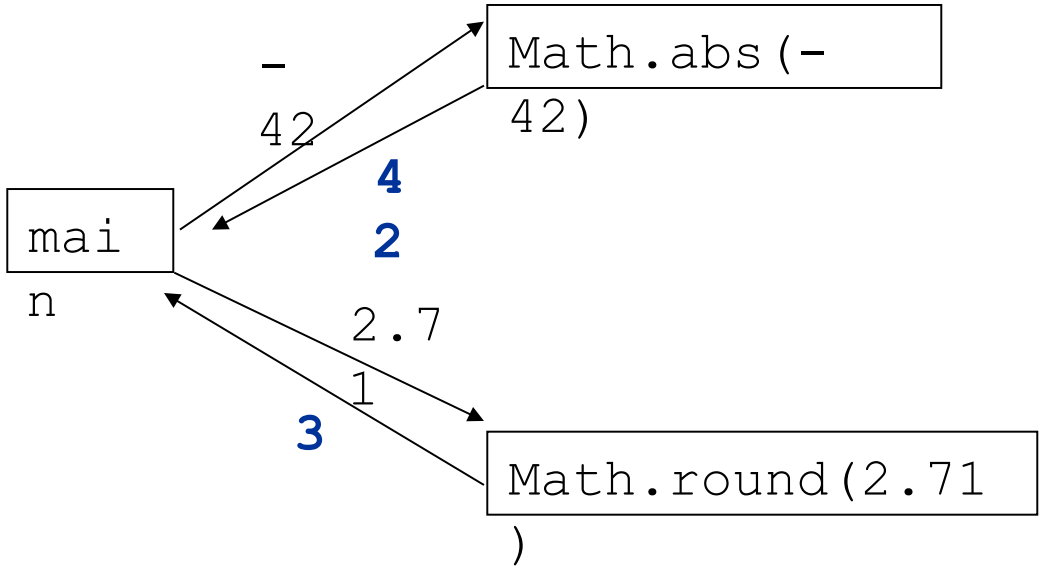
- The `Math` methods do not print to the console.
  - Each method produces ("returns") a numeric result.
  - The results are used as expressions (printed, stored, etc.).

Portions Copyright 2020 by Pearson Education

Return

CS 210

- **return**: To send out a value as the result of a method.
  - The opposite of a parameter:
    - Parameters send information *in* from the caller to the method.
    - Return values send information *out* from a method to its caller.
      - A call to the method can be used as part of an expression.



`Math` exercise

CS 210

- #1 - #8: Evaluate the following expressions:

```
1.  Math.abs(-1.23)
2.  Math.pow(3, 2)
3.  Math.pow(10, -2)
4.  Math.sqrt(121.0) - Math.sqrt(256.0)
5.  Math.round(Math.PI) + Math.round(Math.E)
6.  Math.ceil(6.022) + Math.floor(15.9994)
7.  Math.abs(Math.min(-3, -5))
8.  Math.log10(100)
```

- #9 - #10 `Math.max` and `Math.min` can be used to bound numbers.

Consider an `int` variable named `age`.

9. What statement would replace negative ages with 0?
10. What statement would cap the maximum age to 40?

Portions Copyright 2020 by Pearson Education

Write down your answers…

Portions Copyright 2020 by Pearson Education

Quirks of real numbers

CS 210

- Some `Math` methods return `double` or other non-`int` types.

```
int x = Math.pow(10, 3);   // ERROR: incompat. types
```

- Some `double` values print poorly (too many digits).

```
double result = 1.0 / 3.0;
System.out.println(result);    // 0.3333333333333
```

- The computer represents `double`s in an imprecise way.

```
System.out.println(0.1 + 0.2);
```

  - Instead of 0.3, the output is `0.30000000000000004`

Portions Copyright 2020 by Pearson Education

Type casting

CS 210

- **type cast**: A conversion from one type to another.
  - To promote an `int` into a `double` to get exact division from `/`
  - To truncate a `double` from a real number to an integer

- Syntax:

  (**type**) **expression**

  Examples:

```
double result = (double) 19 / 5;      // 3.8
int result2 = (int) result;           // 3
int x = (int) Math.pow(10, 3);        // 1000
```

Portions Copyright 2020 by Pearson Education

More about type casting

CS 210

- Type casting has high precedence and only casts the item immediately next to it.
  - `double x = (double) 1 + 1 / 2;  // 1.0`
  - `double y = 1 + (double) 1 / 2;  // 1.5`
- You can use parentheses to force evaluation order.
  - `double average = (double) (a + b + c) / 3;`
- A conversion to `double` can be achieved in other ways.
  - `double average = 1.0 * (a + b + c) / 3;`

Portions Copyright 2020 by Pearson Education

Returning a value

CS 210

```
public static type name(parameters) {
    statements;
    ...
    return expression;
}
```

- Example:

```
// Returns the slope of the line between the given points.
public static double slope(int x1, int y1, int x2, int y2)
  {
    double dy = y2 - y1;
    double dx = x2 - x1;
    return dy / dx;
}
```

  - `slope(1, 3, 5, 11)` returns `2.0`

Portions Copyright 2020 by Pearson Education

Return examples

CS 210

```
// Converts degrees Fahrenheit to Celsius.
public static double fToC(double degreesF) {
    double degreesC = 5.0 / 9.0 * (degreesF - 32);
    return degreesC;
}

// Computes triangle hypotenuse length given its side
  lengths.
public static double hypotenuse(int a, int b) {
    double c = Math.sqrt(a * a + b * b);
    return c;
}
```

- You can shorten the examples by returning an expression:

```
public static double fToC(double degreesF) {
    return 5.0 / 9.0 * (degreesF - 32);
}
```

Portions Copyright 2020 by Pearson Education

Common error: Not storing

CS 210

- Many students incorrectly think that a `return` statement sends a variable's name back to the calling method.

```
public static void main(String[] args) {
    slope(0, 0, 6, 3);
    System.out.println("The slope is " + result);  //
  ERROR:
}                                                  // result not defined

public static double slope(int x1, int x2, int y1, int y2)
  {
    double dy = y2 - y1;
    double dx = x2 - x1;
    double result = dy / dx;
    return result;
}
```

Portions Copyright 2020 by Pearson Education

Fixing the common error

- Instead, returning sends the variable's *value* back.
  - The returned value must be stored into a variable or used in an expression to be useful to the caller.

```
public static void main(String[] args) {
    double s = slope(0, 0, 6, 3);
    System.out.println("The slope is " + s);
}

public static double slope(int x1, int x2, int y1, int y2)
  {
    double dy = y2 - y1;
    double dx = x2 - x1;
    double result = dy / dx;
    return result;
}
```

Portions Copyright 2020 by Pearson Education

Objects, Classes and Strings

CS 210

CLASSES
OBJECTS
STRING METHODS

Portions Copyright 2020 by Pearson Education

Classes and objects

CS 210

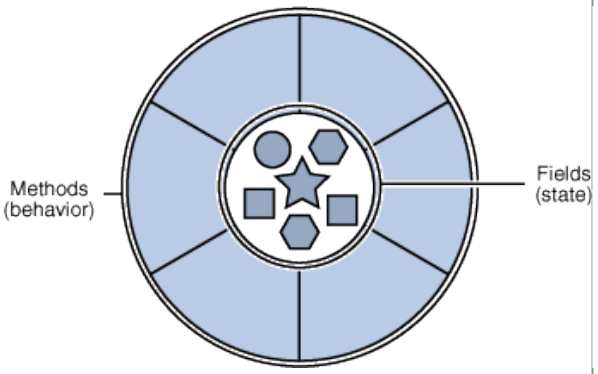
- **class**: A program entity that represents either:
  1. A program / module, or
  2. A type of objects.
  - A class is a blueprint or template for constructing objects.
  - Example: The `DrawingPanel` class (type) is a template for creating many `DrawingPanel` objects (windows).
    - Java has 1000s of classes. Later (Ch.8) we will write our own.
- **object**: An entity that combines data and behavior.
  - **object-oriented programming (OOP)**: Programs that perform their behavior as interactions between objects.

Portions Copyright 2020 by Pearson Education

Objects

CS 210

- **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.



- Constructing (creating) an object:

```
Type objectName = new Type(parameters);
```

- Calling an object's method:

```
objectName.methodName(parameters);
```

Portions Copyright 2020 by Pearson Education

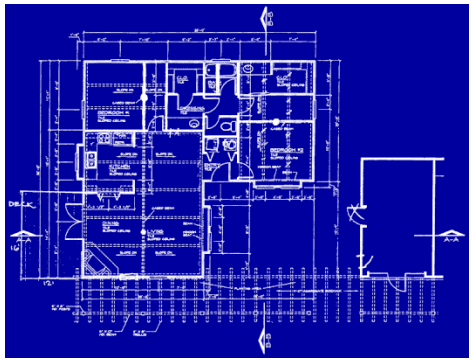
Blueprint analogy

CS 210

iPod blueprint/factory

state:
current song
volume
battery life

behavior:
power on/off
change station/song
change volume
choose random song



creates

iPod #1

state:
song = "1,000,000 Miles"
volume = 17
battery life = 2.5 hrs

behavior:
power on/off
change station/son
change volume
choose random song

iPod #2

state:
song = "Letting You"
volume = 9
battery life = 3.41 hrs

behavior:
power on/off
change station/song
change volume
choose random song

iPod #3

state:
song = "Discipline"
volume = 24
battery life = 1.8 hrs

behavior:
power on/off
change station/song
change volume
choose random song

Portions Copyright 2020 by Pearson Education

Strings

CS 210

- **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 + ")";
```

Portions Copyright 2020 by Pearson Education

Indexes

CS 210

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

```
String name = "R. Kelly";
```

index	0	1	2	3	4	5	6	7
character	R	.		K	e	l	l	y

  - 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

CS 210

Method name	Description
`indexOf(`**str**`)`	index where the start of the given string appears in this string (-1 if not found)
`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* is 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 firstname = "Winnie";
```

Portions Copyright 2020 by Pearson Education

String method examples

CS 210

```
// 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        01234567890123456789 01
String book = "Building Java Programs";
```

  - How would you extract the word "Java" ?

```
System.out.println(book.substring(9, 13));
```

Portions Copyright 2020 by Pearson Education

`string` methods exercises

CS 210

Assume that the following variables have been declared:

```
String str2 = "Arcturan Megadonkey";
String str3 = "Sirius Cybernetics Corporation";
```

Evaluate the following expressions:

```
#1. str2.substring(10, 14)
#2. str3.indexOf("C")
#3. str2 + str3.charAt(17)
#4. str3.substring(9, str3.indexOf("e"))
#5. str2.toLowerCase().substring(9, 13)
#6. str3.substring(18, str3.length() - 7)
```

Portions Copyright 2020 by Pearson Education

Modifying strings

CS 210

- Methods like `substring` and `toLowerCase` build and 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's value, you must reassign it:

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

Portions Copyright 2020 by Pearson Education

Interactive Programs with `Scanner`

CS 210

SCANNER METHODS
SCANNER EXAMPLES
INPUT TOKENS
STRINGS AS USER INPUT

Portions Copyright 2020 by Pearson Education

Input and `System.in`

CS 210

- **interactive program**: Reads input from the console.
  - While the program runs, it asks the user to type input.
  - The input typed by the user is stored in variables in the code.

  - Can be tricky; users are unpredictable and misbehave.
  - But interactive programs have more interesting behavior.

- **`Scanner`**: An object that can read input from many sources.
  - Communicates with `System.in` (the opposite of `System.out`)
  - Can also read from files (Ch. 6), web sites, databases, ...

Portions Copyright 2020 by Pearson Education

`Scanner` syntax

- The `Scanner` class is found in the `java.util` package.

```
import java.util.*;    // so you can use Scanner
```

- Constructing a `Scanner` object to read console input:

```
Scanner name = new Scanner(System.in);
```

  - Example:

```
Scanner console = new Scanner(System.in);
```

Portions Copyright 2020 by Pearson Education

`Scanner` methods

CS 210

Method	Description
`nextInt()`	reads an `int` from the user and returns it
`nextDouble()`	reads a `double` from the user
`next()`	reads a one-word `String` from the user
`nextLine()`	reads a one-*line* `String` from the user

- Each method waits until the user presses Enter.
- The value typed by the user is returned.

```
System.out.print("How old are you? ");  // prompt
int age = console.nextInt();
System.out.println("You typed " + age);
```

Scanner example 1

CS 210

```
import java.util.*;   // so that I can use Scanner

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

        System.out.print("How old are you? ");
        int age = console.nextInt();


        int years = 65 - age;
        System.out.println(years + " years to
  retirement!");
    }
}
```

age 29

years 36

- Console (user input underlined):

```
How old are you? 29
36 years until retirement!
```

Portions Copyright 2020 by Pearson Education

Scanner example 2

CS 210

```
import java.util.*;   // so that I can use Scanner

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

        System.out.print("Please type two numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();

        int product = num1 * num2;
        System.out.println("The product is " + product);
    }
}
```

- Output (user input underlined):

```
Please type two numbers: 8 6
The product is 48
```

  - The `Scanner` can read multiple values from one line.

Portions Copyright 2020 by Pearson Education

Input tokens

CS 210

- **token**: A unit of user input, as read by the `Scanner`.
  - Tokens are separated by *whitespace* (spaces, tabs, new lines).
  - How many tokens appear on the following line of input?

```
23  John Smith   42.0  "Hello world"  $2.50  "  19"
```

- When a token is not the type you ask for, it crashes.

```
System.out.print("What is your age? ");
int age = console.nextInt();
```

Output:

```
What is your age? Timmy
java.util.InputMismatchException
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        ...
```

Portions Copyright 2020 by Pearson Education

Strings as user input

CS 210

- `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? Chamillionaire
CHAMILLIONAIRE has 14 letters and starts with C
```

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

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

Portions Copyright 2020 by Pearson Education

Strings question

CS 210

- Write a program that outputs a person's "gangsta name."
  - first initial
  - *Diddy*
  - last name (all caps)
  - first name
  - *-izzle*

Example Output:

```
Type your name, playa: Marge Simpson
Your gangsta name is "M. Diddy SIMPSON Marge-izzle"
```

Portions Copyright 2020 by Pearson Education

Strings answer

CS 210

```
// 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();

        // 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);

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

Portions Copyright 2020 by Pearson Education

The End

CS 210

CHAPTER 3

INTRODUCTION TO PARAMETERS AND OBJECTS

Winnie Li

Portions Copyright 2020 by Pearson Education