## **CSE 1710**

Lecture 5 *Understanding the Client Role* 

### **UML** (Unified Modeling Language)

Consider the following UML class diagrams:

#### <<utility>> type::lib::ToolBox

computeBMI(int, String): double
factorial(int): double

## <<utility>> type::lang::Math

PI: double

sqrt(double): double

#### type::lib::Rectangle

-width: int -height int

getArea(): int
getCircumference (): int
getDiagonal(): int
getWidth(): int
getHeight(): int
setWidth(): void
setHeight(): void

#### L5App

main(): long
toString(): String

### **UML** (Unified Modeling Language)

Consider the following UML class diagrams:

## <<utility>> type::lib::ToolBox

computeBMI(int, String): double
factorial(int): double

#### type::lib::ToolBox

computeBMI(int, String): double
factorial(int): double

underlined method name indicates the method is static

recall: a utility class is a class that cannot be instantiated

# You are the Client (mostly)

#### Why?

- because as a *client* you are using the services provided by others
  - E.g., PrintStream, Rectangle, ToolBox, etc

Why do you say "mostly"?

- because you are also the *implementer* of a class
  - you are also providing a service
  - $\bullet$  Go to the example class L5App

4

# Your duties as an implementer...

- inform potential *clients* of your services
  - Use the app javadoc
- in principle, you could provide other services
  - other methods (demostrate)
  - fields (demonstrate)

5



A class diagram from the TYPE library:

A class diagram from the Java standard library

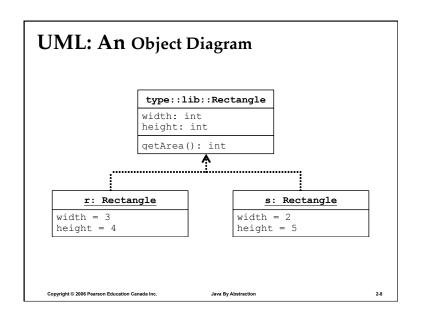
getTime(): long

java::util::Date

Copyright © 2006 Pearson Education Canada Inc.

Java By Abstraction 2-7

**UML** (Unified Modeling Language) Consider the following UML class diagrams: <<utility>> type::lib::Rectangle type::lib::ToolBox -width: int computeBMI(int, String): double -height int factorial(int): double getArea(): int getCircumference (): int getDiagonal(): int getWidth(): int getHeight(): int <<utility>> setWidth(): void type::lang::Math setHeight(): void PI: double sqrt(double): double java::util::Date PI: double getTime(): long toString(): String NOT java::sql::Date



## Take Home Points

- Do you know
  - the difference between <u>an object reference</u> and an <u>object</u>?
  - how to recognize the use of a static method?
  - how to recognize the use of a non-static method?
  - how to declare an object reference?
  - how to assign the object reference to refer to a particular object?
  - how to use a static method?
  - how to use a non-static method?