CSE1710

Week 03, Lecture 06

Fall 2013 ◆ Thursday, Sept 26, 2013

YORK UNIVERSITÉ UNIVERSITY

Checklist (for Today)

What we are reinforcing with the exercises this class…

- ❑ recognizing the various different arithmetic operators (in situ)
- ❑ being able to perform (manual) type casting; recognizing when automatic promotion is going to happen
- ❑ being able to evaluate arithmetic expressions (including those with all the *different* arithmetic operators and with *automatic* promotion)

From Lecture #05
- ❑read section 2.1
- ❑review Ch 2 KC's 1-10
- ❑do Ch 2 RQ's 1-18
- ❑do Ch 2 Ex's 2.1-2.10

YORK UNIVERSITÉ UNIVERSITY

2

Checklist (for Lecture 07)

What you should be doing to prepare for what comes next…

- ❑ read section 2.2
- ❑ review Ch 2 KC's 11-20
- ❑ do Ch 2 RQ's 19-29
- ❑ do Ch 2 Ex's 2.11-2.12

YORK UNIVERSITÉ UNIVERSITY

Review: Tasks you should be able to perform…

- Given an expression, determine its outcome (not only the value but the value's type)
- Characterize what is meant by closure
- Given a symbol, state which operators it represents
- Given an operator, describe its behaviour (closure, undefined operations)

YORK UNIVERSITÉ UNIVERSITY

Review: Usability vs Correctness

Usability
how easy is the app to use? how learnable is it? how steep is the learning curve?

Correctness
does the app do what it says it will do?

YORK UNIVERSITÉ UNIVERSITY

Example 1: Body Mass Index

Body Mass Index (BMI) is a heuristic for *estimating* an individual's **body fat** based on that individual's height and weight.

It is inexact. For instance, BMI overestimates body fat for athletes and underestimates body fat for those with low lean body mass.

YORK UNIVERSITÉ UNIVERSITY

Exercise

Suppose we want to compute the Body Mass Index (BMI) for an particular individual

weight: 170 pounds
height: 5'9"

Write the code that derives BMI for this individual.

1. declare and assign variable for `weight`
2. declare and assign variable `height`
3. declare a variable `bmi`
4. construct an assignment statement for `bmi`

$$BMI = \left(\frac{\text{Weight in Pounds}}{(\text{Height in inches}) \times (\text{Height in inches})} \right) \times 703$$

or

$$BMI = \frac{\text{Weight in Kilograms}}{(\text{Height in Meters}) \times (\text{Height in Meters})}$$

7

YORK UNIVERSITÉ UNIVERSITY

Solution *goes here*

8

YORK UNIVERSITÉ UNIVERSITY

Analysis of the Solution

- identify which statements handle storage (of data)
- identify which statements handle computation (of BMI)
- identify implications of choice of data type for variables and literal
- is there any **delegation** in the solution?

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.1

Today's applications must **delegate** some or all of its work to other **classes**.

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.2

In the **procedural** paradigm, the program invokes methods **on** the class.

YORK UNIVERSITÉ UNIVERSITY

Example of Procedural Delegation

Let's look at the class called `ToolBox`

`http://www.eecs.yorku.ca/teaching/docs/type-api/`

```
public static double getBMI(double weight,
                            java.lang.String height)
```

Compute the body mass index.

Parameters:

`weight` - the weight in pounds. To be valid, weight must be positive.

`height` - the height in feet'inches. To be valid, height must have a feet component (a positive integer) optionally followed by an inches component (a non-negative integer less than 12). And if both components are present then they must be separated by a single quote.

Returns:

the body mass index (BMI) for the given weight and height

Throws:

`java.lang.RuntimeException` - if either weight or height is not valid as defined above.

How nice!
There is a service that will compute BMI for us.
We can delegate this task to the class `ToolBox`

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.3

A **method** has a **signature** (name and parameter types) and a **return** type (type of the returned value). If a method is **void**, then it has no return. Data is **passed** to a method through **parameters**.

YORK UNIVERSITÉ UNIVERSITY

Exercise

Identify the signature, return type, and parameters of `getBMI`

```
public static double getBMI(double weight,
                            java.lang.String height)
```

Compute the body mass index.

Parameters:

`weight` - the weight in pounds. To be valid, weight must be positive.

`height` - the height in feet'inches. To be valid, height must have a feet component (a positive integer) optionally followed by an inches component (a non-negative integer less than 12). And if both components are present then they must be separated by a single quote.

Returns:

the body mass index (BMI) for the given weight and height

Throws:

`java.lang.RuntimeException` - if either weight or height is not valid as defined above.

YORK UNIVERSITÉ UNIVERSITY

Exercise

Suppose we want to compute the Body Mass Index (BMI) for an particular individual

weight: 170 pounds
height: 5'9"

Write the code that derives BMI for this individual.

1. declare and assign variable for `weight`
2. declare and assign variable `height`
3. declare a variable `bmi`
4. construct an assignment statement for `bmi` using the services of the class `ToolBox`

YORK UNIVERSITÉ UNIVERSITY

Solution *goes here*

YORK UNIVERSITÉ UNIVERSITY

Analysis of the Solution

- identify which statements handle storage (of data)
- identify which statements handle computation (of BMI)
- is there any **delegation** in the solution?

YORK UNIVERSITÉ UNIVERSITY

Question

- What is the **contract** for the method `getBMI`?
- If the user gives a value for weight that is zero, does this **violate** the contract?
- If the user gives a value of "`89'99`" for height, does this violate the contract?
- If the method returns a value of `-12` for the parameters `170` and "`5'6`", is this a violation of the contract?
- If the method returns a value of `25.101869355177485` for the parameters `170` and "`4'18`", is this a violation of the contract?

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.4

In the **modular** paradigm, the program accesses attributes and invokes methods **on** the class.

In the **procedural** paradigm, the program invokes methods **on** the class.

An **attribute** has a name and a type and allows data to **persist**.

Attributes and methods are known collectively as **features**.

YORK UNIVERSITÉ UNIVERSITY

http://docs.oracle.com/javase/6/docs/api/

Another useful service

java.lang
Class Math

java.lang.Object
└ **java.lang.Math**

Field Summary	
static double	**E** The double value that is closer than any other to *e*, the base of the natural logarithms.
static double	**PI** The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

static double	**pow**(double a, double b) Returns the value of the first argument raised to the power of the second argument.

How nice!
There is a service that represents the value of *pi* for us. It also provides various operations, including exponent.

YORK UNIVERSITÉ UNIVERSITY

Exercise

Suppose we want to compute the area of a circle

radius: 15 cm

Write the code that derives the area of this circle.

1. declare and assign variable for `radius`
2. declare and assign variable `PI`
3. declare a variable `area`
4. construct an assignment statement for `area` using an arithmetic expression

YORK UNIVERSITÉ UNIVERSITY

Solution *goes here*

YORK UNIVERSITÉ UNIVERSITY

Analysis of the Solution

- identify which statements handle storage (of data)
- identify which statements handle computation (of area)
- identify implications of choice of data type for variables and literal, if any
- is there any **delegation** in the solution?
- what type of method has been employed? static or non-static?

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.5

In the **object-oriented** paradigm, the program instantiates the class and uses the created object (rather than the class).

This paradigm is also known as **OOP** (for object oriented paradigm).

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.5

A **Class** has:	An **Object** has:
• attributes • methods	• attributes • methods • object reference • state

state is defined as the value of all of the object's attributes

YORK UNIVERSITÉ UNIVERSITY

Exercise

Suppose we want to compute the area of a rectangle

height: 22 cm
width: 28 cm

Write the code that derives the area of this rectangle.

1. declare and assign variable for `height`
2. declare and assign variable `width`
3. declare a variable `area`
4. construct an assignment statement for `area` using an arithmetic expression

YORK UNIVERSITÉ UNIVERSITY

Solution *goes here*

YORK UNIVERSITÉ UNIVERSITY

Analysis of the Solution

- identify which statements handle storage (of data)
- identify which statements handle computation (of area)
- identify implications of choice of data type for variables and literal, if any
- is there any **delegation** in the solution?

YORK UNIVERSITÉ UNIVERSITY

http://www.eecs.yorku.ca/teaching/docs/type-api/

Another useful service

type.lib
Class Rectangle

java.lang.Object
  └ **type.lib.Rectangle**

All Implemented Interfaces:
java.io.Serializable

public class **Rectangle**
extends java.lang.Object
implements java.io.Serializable

Constructor Summary

`Rectangle()`	Construct a rectangle with zero width and zero height.
`Rectangle(int width, int height)`	Construct a rectangle with the passed width and height.
`Rectangle(Rectangle rectangle)`	Construct a copy of the passed rectangle.

Method Summary

boolean	`equals(java.lang.Object other)` Determine if this rectangle is the same as the passed one.
int	`getArea()` Determine the area of this rectangle.
int	`getCircumference()` Determine the circumference of this rectangle.
double	`getDiagonal()` Determine the length of the diagonal of this rectangle.
int	`getHeight()` Determine the height of this rectangle.
	the width of this rectangle.
	a hash code for this Rectangle.
	height) height this rectangle.
	width) width this rectangle.

How nice!
There is a service that represents rectangles for us.
It provides various operations, including area!
We can delegate to the class `Rectangle`

29

Exercise

Suppose we want to compute the area of a rectangle

height: 22 cm
width: 28 cm

Write the code that derives the area of this circle.

1. use the services of `Rectangle` to store the values of `height` and `width`
2. use the services of `Rectangle` to derive the `area` that corresponds to these values

30

Solution *goes here*

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.6

A **class diagram** in UML (Unified Modelling Language) represents a class definition.

In such diagrams, a class is depicted as a rectangle with one or more compartments.

The top compartment is mandatory and contains the class name (possibly quantified) and an optional **stereotype**.

The next two compartments are optional: one for attributes and one for methods.

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.7

A utility class cannot be instantiated.

For a utility class, all features are said to be static.

All features are accessed/invoked **on the class name**.

Such classes are denoted in UML (Unified Modelling Language) by the stereotype `<<utility>>`

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.8

An **object diagram** in UML represents an object.

In such diagrams, an object is depicted as a rectangle with two compartments: the object's identity is in the top compartment and its state in the bottom one.

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.9

In UML, a dashed line can be placed between:

- a class diagram and a class diagram
- a class diagram and an object diagram

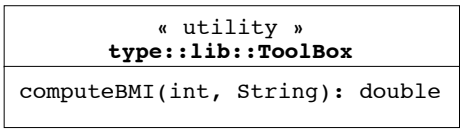
Between class diagram A and B means **class A delegates to class B**

Between class diagram A and object diagram X means the **object X is an instance of class A.**

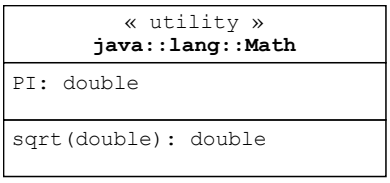
YORK UNIVERSITÉ UNIVERSITY

UML: class diagrams (utility classes)

The **class diagram** of a utility class in the TYPE library:



The **class diagram** of a utility class in the Java library:

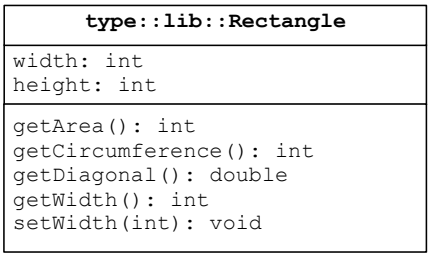


Copyright © 2006 Pearson Education Canada Inc.
Java By Abstraction
2-36

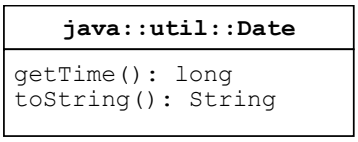
YORK UNIVERSITÉ UNIVERSITY

UML: class diagrams (non-utility classes)

The **class diagram** of a utility class in the TYPE library:



The **class diagram** of a utility class in the Java library:

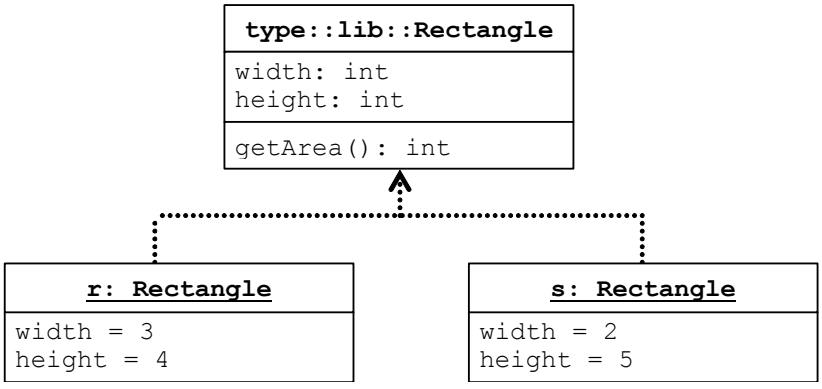


Copyright © 2006 Pearson Education Canada Inc.
Java By Abstraction
2-37

37

YORK UNIVERSITÉ UNIVERSITY

UML: object diagram



38

YORK UNIVERSITÉ UNIVERSITY

Key Concept 2.10

Abstraction is a strategy for replacing complexity with simplicity. **Layered abstractions** give rise to **abstraction** hierarchies in which **higher**-level abstractions have fewer details.

YORK UNIVERSITÉ UNIVERSITY

Tasks you should be able to perform:

- Recognize the delegation of a task and the delegation of representation within an application
- Explain the difference between an object reference and an object

YORK UNIVERSITÉ UNIVERSITY

Tasks you should be able to perform:

- Recognize the use of a static method
- Recognize the use of a non-static method
- How to declare an variable to represents an object reference
- How to obtain an object reference and to store the reference for subsequent use
- How to use a static method
- How to use a non-static method

YORK UNIVERSITÉ UNIVERSITY

Little/No Delegation

decide on grain type …*rye*

buy the appropriate seed type; learn about growing techniques

grow grain

harvest grain

bring grain inside to grinding room

buy grinder
load hopper of grinder

grind grain (repeat until enough grain obtained)

secure yeast, water

prepare dough

bake bread

let bread cool and slice

eat bread

Little/No Delegation	Some Delegation
decide on grain type …*rye*	decide on grain type …*rye*
buy the appropriate seed type; learn about growing techniques	
grow grain	**place an order for a bag of grain** • need to find the farmer • need to figure out what size of bag to buy (see next step)
harvest grain	
bring grain inside to grinding room	transport grain to grinding place
buy grinder load hopper of grinder	**place an order to get grain milled** • need to find a mill • need to ensure that you give them the correctly-sized bag (e.g., the mill may stipulate input conditions, such as min size of packaging)
grind grain (repeat until enough grain obtained)	
secure yeast, water	secure yeast, water
prepare dough	prepare dough
bake bread	bake bread
let bread cool and slice	let bread cool and slice
eat bread	eat bread

Little/No Delegation	Some Delegation	Much Delegation
decide on grain type …*rye*	decide on grain type …*rye*	decide on grain type …*rye*
buy the appropriate seed type; learn about growing techniques		
grow grain	**place an order for a bag of grain** • need to find the farmer • need to figure out what size of bag to buy (see next step)	
harvest grain		
bring grain inside to grinding room	transport grain to grinding place	
buy grinder load hopper of grinder	**place an order to get grain milled** • need to find a mill • need to ensure that you give them the correctly-sized bag (e.g., the mill may stipulate input conditions, such as min size of packaging)	place an order: "I'd like a loaf of sliced *rye* bread"
grind grain (repeat until enough grain obtained)		
secure yeast, water	secure yeast, water	
prepare dough	prepare dough	
bake bread	bake bread	
let bread cool and slice	let bread cool and slice	
eat bread	eat bread	eat bread

SOLUTIONS

YORK UNIVERSITÉ UNIVERSITY

Is this a static method?

```
public static double getBMI(double weight,
                            java.lang.String height)
```

Compute the body mass index.

Parameters:

`weight` - the weight in pounds. To be valid, weight must be positive.
`height` - the height in feet'inches. To be valid, height must have a feet component (a positive integer) optionally followed by an inches component (a non-negative integer less than 12). And if both components are present then they must be separated by a single quote.

Returns:

the body mass index (BMI) for the given weight and height

Throws:

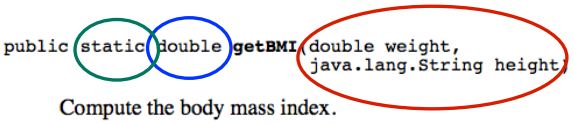
`java.lang.RuntimeException` - if either weight or height is not valid as defined above.

Is this a static method? yes, see the keyword

What value is returned? double

What parameters do we need? a double and a String

What is the contract?



Compute the body mass index.

Parameters:

`weight` - the weight in pounds. To be valid, weight must be positive.

`height` - the height in feet'inches. To be valid, height must have a feet component (a positive integer) optionally followed by an inches component (a non-negative integer less than 12). And if both components are present then they must be separated by a single quote.

Returns:

the body mass index (BMI) for the given weight and height

Throws:

`java.lang.RuntimeException` - if either weight or height is not valid as defined above.

```
double bmi;
double weight = 170.0;
String height = "5'9";
bmi = ???
```

47

So to use the method…

```
//double bmi;
double weight = 170.0;
String height = "5'9";
double bmi = ToolBox.getBMI(weight, height);
```

We have delegated computation,
but NOT storage.

48

Compare

```
double weightInLbs = 170.0;
int heightInInches = 5*12+9;
double bmi = weightInLbs
             / (heightInInches*heightInInches) * 703;
```

```
double bmi;
double weight = 170.0;
String height = "5'9″;
bmi = ToolBox.getBMI(weight, height);
```

49

We need to use the services of `Rectangle` to first represent the shape and then to perform computation….

```
public Rectangle(int width,
                 int height)
```

Construct a rectangle with the passed width and height.

Parameters:

`width` - the width of the rectangle to construct.
`height` - the height of the rectangle to construct.

```
int width = 22;
int heigh = 28;

Rectangle letterSizedPaper;
letterSizedPaper = new Rectangle(widthInCM, heightInCM);
```

50

Is this a static method? no – no keyword

What value is returned? int

What parameters do we need? none

```
public int getArea()
```

Determine the area of this rectangle.

Returns:

the area of this rectangle

```
int width = 22;
int heigh = 28;

Rectangle letterSizedPaper;
letterSizedPaper = new Rectangle(widthInCM, heightInCM);

double surfaceAreaLetterSizedPaper;
surfaceAreaLetterSizedPaper = letterSizedPaper.getArea()
```

We have delegated computation, AND storage.

51