

CSE1710

Week 04, Lecture 08

Click to edit this text block.

Second level

Third level

Fifth level

Fall 2013 ♦ Thurs, Oct 3, 2013



Big Picture

The assigned reading was for today:

- The Client View; sec 2.2.2, pp. 60-64
- Post-Compilation Errors; sec 2.2.3, pp. 64-65
- Java Standard Library; sec 2.2.4, pp. 66-68
- Readymade I/O; sec 2.2.5, pp. 68-70



Big Picture

The assigned reading for next lecture is:

- Software Engineering sec 2.3
- Risk Mitigation Early Exposure; sec 2.3.1, pp. 71
- Handling Constants; sec 2.3.2, pp. 71-72
- **Contracts; sec 2.2.4, pp. 73-77**

3



Big Picture

What are we reinforcing with the exercises this lecture?

- Key Concepts 2.11-2.20
- Ability to complete review questions 2.19-2.29
- Ability to complete Exercises 2.1-2.12

4



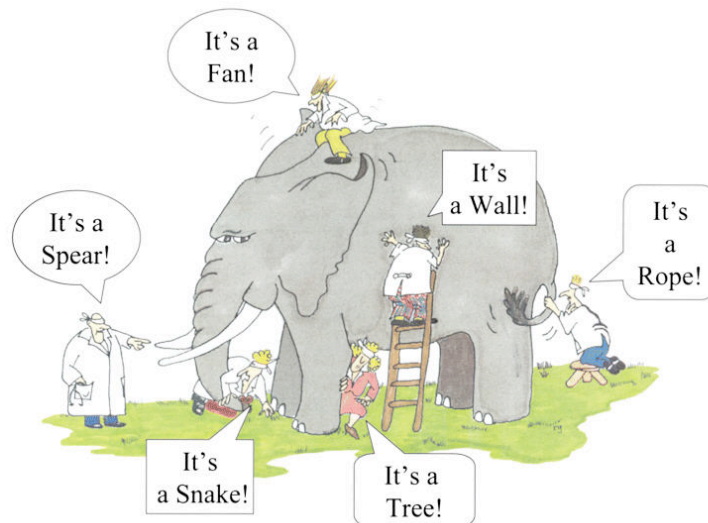
Checklist (for Lecture 09)

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

- review sections 2.1-2.3
- review Ch 2 KC's 1-25
- review answers to Ch 2 RQ's 1-35
- review answers to Ch 2 Ex's 2.1-2.22

5

What is this?



6

What is this?

```

1 package type.11b;
2 public class ToolBox {
3     // ...
4     private static double BMI_IMPERIAL_FACTOR = 703;
5     private ToolBox() {}
6 }
7
8 /**
9  * Compute the body mass index.
10  *
11  * @param weight
12  *      The weight in pounds. To be valid, weight must be positive.
13  * @param height
14  *      The height in feet/inches. To be valid, height must have a
15  *      feet component (a positive integer) optionally followed by an
16  *      inches component (a non-negative integer less than 12). One of
17  *      both components are present. They must be separated by a
18  *      single space.
19  * @return the body mass index (BMI) for the given weight and height.
20  * @throws RuntimeException
21  *      If either weight or height is not valid as defined above.
22  */
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
    
```

```

Compiled From "ToolBox.java"
public class type.11b.ToolBox extends java.lang.Object {
    public static double getBMI(double, java.lang.String):
    Code:
    0:   aload_0
    1:   aconst_double #211: //double 0.4545454545
    2:   dmul
    3:   dconst_double #213: //double 12.86
    4:   ddiv
    5:   dconst_double #215: //double 0.0254
    6:   dmul
    7:   dconst_double #217: //String :
    8:   invokevirtual #219: //Method java/lang/String.indexOf:()Ljava/lang/CharSequence;
    9:   lconst_int 56
    10:  lsub
    11:  lconst_int 1
    12:  lsub
    13:  lconst_int 1
    14:  lsub
    15:  lconst_int 1
    16:  lsub
    17:  invokevirtual #216: //Method java/lang/String.substring:(I)Ljava/lang/String;
    18:  invokevirtual #222: //Method java/lang/Integer.parseInt:(Ljava/lang/String;)I
    19:  lconst_int 9
    20:  ldiv
    21:  lconst_int 1
    22:  ldiv
    23:  lconst_int 1
    24:  ldiv
    25:  lconst_int 1
    26:  ldiv
    27:  lconst_int 1
    28:  ldiv
    29:  lconst_int 1
    30:  ldiv
    31:  lconst_int 1
    32:  ldiv
    33:  lconst_int 1
    34:  ldiv
    35:  lconst_int 1
    36:  ldiv
    37:  lconst_int 1
    38:  ldiv
    39:  lconst_int 1
    40:  ldiv
    41:  lconst_int 1
    42:  ldiv
    43:  lconst_int 1
    44:  ldiv
    45:  lconst_int 1
    46:  ldiv
    47:  lconst_int 1
    48:  ldiv
    49:  lconst_int 1
    50:  ldiv
    51:  lconst_int 1
    52:  ldiv
    53:  lconst_int 1
    54:  ldiv
    55:  lconst_int 1
    56:  ldiv
    57:  lconst_int 1
    58:  ldiv
    59:  lconst_int 1
    60:  ldiv
    61:  lconst_int 1
    62:  ldiv
    63:  lconst_int 1
    64:  ldiv
    65:  lconst_int 1
    66:  ldiv
    67:  lconst_int 1
    68:  ldiv
    69:  lconst_int 1
    70:  ldiv
    71:  lconst_int 1
    72:  ldiv
    73:  lconst_int 1
    74:  ldiv
    75:  lconst_int 1
    76:  ldiv
    77:  lconst_int 1
    78:  ldiv
    79:  lconst_int 1
    80:  ldiv
    81:  lconst_int 1
    82:  ldiv
    83:  lconst_int 1
    84:  ldiv
    85:  lconst_int 1
    86:  ldiv
    87:  lconst_int 1
    88:  ldiv
    89:  lconst_int 1
    90:  ldiv
    91:  lconst_int 1
    92:  ldiv
    93:  lconst_int 1
    94:  ldiv
    95:  lconst_int 1
    96:  ldiv
    97:  lconst_int 1
    98:  ldiv
    99:  lconst_int 1
    100: ldiv
    }
    
```

Class
Class ToolBox

java.lang.Object
↳ java.lang.ToolBox

public class ToolBox {
 ...
}

This class contains various utilities. Refer to the Java By Abstraction textbook.

Version: 1.0 - Summer 2010
Author: H. Rowland, rrowland@yorku.ca

Method	Signature	Description
getBMI	(double, java.lang.String) double	Compute the body mass index.
repeat	(int, char) String	Repeat the character the specified number of times.
launch	(String, String) String	Launch a new process with the specified command and arguments.
factorial	(int) double	Compute the factorial of the specified integer.
mortgagePayment	(double, double, double, int) double	Compute the monthly payment on a mortgage assuming constant payments and constant interest rate.
crash	(boolean, String) void	Crash the JVM with the specified message.

```

<<Java Class>>
classDiagram
    class ToolBox {
        +ToolBox()
        +crash(boolean,String):void
        +repeat(int,char):String
        +launch(String,String):String
        +mortgagePayment(double,double,int):double
        +getBMI(double,String):double
        +factorial(int):double
    }
    
```



7

What is this?

```

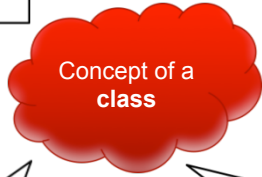
1 package type.11b;
2 public class ToolBox {
3     // ...
4     private static double BMI_IMPERIAL_FACTOR = 703;
5     private ToolBox() {}
6 }
7
8 /**
9  * Compute the body mass index.
10  *
11  * @param weight
12  *      The weight in pounds. To be valid, weight must be positive.
13  * @param height
14  *      The height in feet/inches. To be valid, height must have a
15  *      feet component (a positive integer) optionally followed by an
16  *      inches component (a non-negative integer less than 12). One of
17  *      both components are present. They must be separated by a
18  *      single space.
19  * @return the body mass index (BMI) for the given weight and height.
20  * @throws RuntimeException
21  *      If either weight or height is not valid as defined above.
22  */
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
    
```

```

Compiled From "ToolBox.java"
public class type.11b.ToolBox extends java.lang.Object {
    public static double getBMI(double, java.lang.String):
    Code:
    0:   aload_0
    1:   aconst_double #211: //double 0.4545454545
    2:   dmul
    3:   dconst_double #213: //double 12.86
    4:   ddiv
    5:   dconst_double #215: //double 0.0254
    6:   dmul
    7:   dconst_double #217: //String :
    8:   invokevirtual #219: //Method java/lang/String.indexOf:()Ljava/lang/CharSequence;
    9:   lconst_int 56
    10:  lsub
    11:  lconst_int 1
    12:  lsub
    13:  lconst_int 1
    14:  lsub
    15:  lconst_int 1
    16:  lsub
    17:  invokevirtual #216: //Method java/lang/String.substring:(I)Ljava/lang/String;
    18:  invokevirtual #222: //Method java/lang/Integer.parseInt:(Ljava/lang/String;)I
    19:  lconst_int 9
    20:  ldiv
    21:  lconst_int 1
    22:  ldiv
    23:  lconst_int 1
    24:  ldiv
    25:  lconst_int 1
    26:  ldiv
    27:  lconst_int 1
    28:  ldiv
    29:  lconst_int 1
    30:  ldiv
    31:  lconst_int 1
    32:  ldiv
    33:  lconst_int 1
    34:  ldiv
    35:  lconst_int 1
    36:  ldiv
    37:  lconst_int 1
    38:  ldiv
    39:  lconst_int 1
    40:  ldiv
    41:  lconst_int 1
    42:  ldiv
    43:  lconst_int 1
    44:  ldiv
    45:  lconst_int 1
    46:  ldiv
    47:  lconst_int 1
    48:  ldiv
    49:  lconst_int 1
    50:  ldiv
    51:  lconst_int 1
    52:  ldiv
    53:  lconst_int 1
    54:  ldiv
    55:  lconst_int 1
    56:  ldiv
    57:  lconst_int 1
    58:  ldiv
    59:  lconst_int 1
    60:  ldiv
    61:  lconst_int 1
    62:  ldiv
    63:  lconst_int 1
    64:  ldiv
    65:  lconst_int 1
    66:  ldiv
    67:  lconst_int 1
    68:  ldiv
    69:  lconst_int 1
    70:  ldiv
    71:  lconst_int 1
    72:  ldiv
    73:  lconst_int 1
    74:  ldiv
    75:  lconst_int 1
    76:  ldiv
    77:  lconst_int 1
    78:  ldiv
    79:  lconst_int 1
    80:  ldiv
    81:  lconst_int 1
    82:  ldiv
    83:  lconst_int 1
    84:  ldiv
    85:  lconst_int 1
    86:  ldiv
    87:  lconst_int 1
    88:  ldiv
    89:  lconst_int 1
    90:  ldiv
    91:  lconst_int 1
    92:  ldiv
    93:  lconst_int 1
    94:  ldiv
    95:  lconst_int 1
    96:  ldiv
    97:  lconst_int 1
    98:  ldiv
    99:  lconst_int 1
    100: ldiv
    }
    
```

source code (partial)

bytecode (partial)



API

Class
Class ToolBox

java.lang.Object
↳ java.lang.ToolBox

public class ToolBox {
 ...
}

This class contains various utilities. Refer to the Java By Abstraction textbook.

Version: 1.0 - Summer 2010
Author: H. Rowland, rrowland@yorku.ca

Method	Signature	Description
getBMI	(double, java.lang.String) double	Compute the body mass index.
repeat	(int, char) String	Repeat the character the specified number of times.
launch	(String, String) String	Launch a new process with the specified command and arguments.
factorial	(int) double	Compute the factorial of the specified integer.
mortgagePayment	(double, double, double, int) double	Compute the monthly payment on a mortgage assuming constant payments and constant interest rate.
crash	(boolean, String) void	Crash the JVM with the specified message.

```

<<Java Class>>
classDiagram
    class ToolBox {
        +ToolBox()
        +crash(boolean,String):void
        +repeat(int,char):String
        +launch(String,String):String
        +mortgagePayment(double,double,int):double
        +getBMI(double,String):double
        +factorial(int):double
    }
    
```

UML Class Diagram



8

What Services Does A Class Offer?

Name two possible formats that are used in software practice to convey the service that a particular class offers.

For instance, take the `ToolBox` class.

Hint: Both of the formats are 3-letter acronyms

1. _____

2. _____

9



API for `ToolBox`

```
type:lib
Class ToolBox
java.lang.Object
├── type.lib.ToolBox
└──
public class ToolBox
extends java.lang.Object
This class contains various utilities. Refer to the Java By Abstraction textbook.
Version:
7.0 - Summer 2010
Author:
H. Roumani, roumani@cse.yorku.ca
```

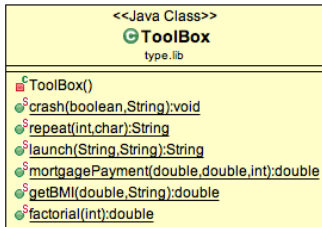
Method Summary	
static void	crash (boolean condition, java.lang.String msg) Test the passed condition and throw an exception if it is true.
static double	factorial (int n) Determine the factorial of the passed integer.
static double	getBMI (double weight, java.lang.String height) Compute the body mass index.
static java.lang.String	launch (java.lang.String app, java.lang.String input) Launch the indicated app, pass the indicated string to its standard input, and capture (and return) its standard output.
static double	mortgagePayment (double amount, double rate, int period) Compute the monthly payment on a mortgage assuming constant payments and constant interest rate.
static java.lang.String	repeat (int count, char c) Create a string containing the passed character repeated as many times as specified.

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

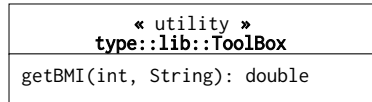
10



UML Class Diagram for Toolbox



Example of auto-generated UML (ObjectAid plugin)



Example of hand-crafted UML (partial – not all methods shown)

Differences in:

- notation, stereotype (all methods static vs short-hand “utility” stereotype)
- constructor (shown with red square vs not shown at all)
- completeness (all methods vs only one)



Other formats for Toolbox

```
Compiled from "ToolBox.java"
public class type.lib.ToolBox extends java.lang.Object {
    public static double getBMI(double, java.lang.String);
}
Code:
0: ldc2_w #211: //double 0.45359237d
3: dstore_3
4: ldc2_w #213: //double 12.0d
7: dstore_5
9: ldc2_w #215: //double 0.0254d
12: dstore_7
14: aload_2
15: ldc #217: //String '
17: invokevirtual #219: //Method java/lang/String.indexOf:(Ljava/lang/CharSequence)I
20: istore_11
22: lload_11
24: iconst_m1
25: ifltmpex 56
28: aload_2
29: iconst_0
30: lload_11
32: invokevirtual #76: //Method java/lang/String.substring:(II)Ljava/lang/String;
35: invokevirtual #222: //Method java/lang/Integer.parseInt:(Ljava/lang/String)I
38: istore_9
40: aload_2
41: lload_11
43: iconst_1
44: ladd
45: invokevirtual #88: //Method java/lang/String.substring:(II)Ljava/lang/String;
48: invokevirtual #222: //Method java/lang/Integer.parseInt:(Ljava/lang/String)I
51: istore_10
53: goto 65
56: aload_2
57: invokevirtual #222: //Method java/lang/Integer.parseInt:(Ljava/lang/String)I
60: istore_9
62: iconst_0
63: lstore_10
65: aload_0
66: ldc2_w #211: //double 0.45359237d
```

Bytecode (partial)

```
1 package type.lib;
2
3 public class Toolbox {
4
5     private static double BMI_IMPERIAL_FACTOR = 703;
6
7     private Toolbox() {
8     }
9
10    /**
11     * Compute the body mass index.
12     *
13     * @param weight
14     *     the weight in pounds. To be valid, weight must be positive.
15     * @param height
16     *     the height in feet'inches. To be valid, height must have a
17     *     feet component (a positive integer) optionally followed by an
18     *     inches component (a non-negative integer less than 12). And if
19     *     both components are present then they must be separated by a
20     *     single quote.
21     * @return the body mass index (BMI) for the given weight and height
22     * @throws RuntimeException
23     *     if either weight or height is not valid as defined above.
24     */
25    public static double getBMI(double weight, String height)
26        throws RuntimeException {
27        // validation methods below throw exception
28        validate(height);
29        validate(weight);
30        int heightInches = convertHeightFromString(height);
31        double bmi = weight / (heightInches * heightInches)
32            * BMI_IMPERIAL_FACTOR;
33        return bmi;
34    }
35
36    private static int convertHeightFromString(String height) {
```

source code (partial)



Exercise 2.4, 2.5

<code>« utility » jba::Orbit</code>
<code>payBack(double, int): double</code>

Are there any compile-time errors in the fragments below?
Assume the class `Orbit` has been properly imported.

```
//fragment#2.4  
double amount = 2500;  
int period = 4;  
double pay = Orbit.payBack(amount, period);
```

```
//fragment#2.5  
int amount = 2500;  
int period = 4;  
double pay = Orbit.payBack(amount, period);
```

13



Exercise 2.6, 2.7

<code>« utility » jba::Orbit</code>
<code>payBack(double, int): double</code>

Are there any compile-time errors in the fragments below?
Assume the class `Orbit` has been properly imported.

```
//fragment#2.6  
float amount = 2500;  
int period = 4;  
double pay = Orbit.payBack(amount, period);
```

```
//fragment#2.7  
double amount = 2500;  
long period = 4;  
double pay = Orbit.payBack(amount, period);
```

14



Exercise 2.8

<code>« utility » jba::Orbit</code>
<code>payBack(double, int): double</code>

Are there any compile-time errors in the fragment below?
Assume the class `Orbit` has been properly imported.

```
//fragment#2.8  
double amount = 2500;  
int period = 4;  
int pay = Orbit.payBack(amount, period);
```

15



Exercise 2.9

<code>« utility » jba::Bond</code>
<code>rating: char rate: double</code>
<code>estimate(): double inflate(): void</code>

Are there any compile-time errors in the fragment below?
Assume the class `Bond` has been properly imported.

```
PrintStream output = System.out;  
  
Bond.rating = 'C';  
Bond.rate = 0.12;  
double x = Bond.estimate();  
output.println(Bond.inflate());
```

16



Exercise 2.10

« utility » jba::Bond
rating: char
rate: double
estimate(): double
inflate(): void

Are there any compile-time errors in the fragment below?
Assume the class Bond has been properly imported.

```
Bond.rating = 'C';
Bond.rate = 0.12;
double x = Bond.estimate();
Bond.inflate();
```

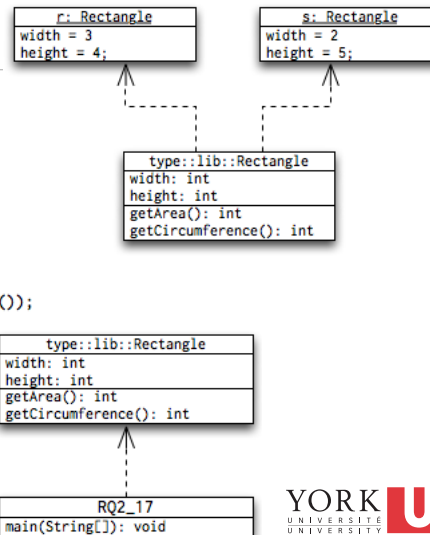
17



Q. Explain the difference between a class diagram and an object diagram.

Explain in terms of notation, the objects & relationships shown, and the contexts.

```
1 import type.lib.Rectangle;
2
3 public class RQ2_17 {
4
5     /**
6      * @param args
7      */
8     public static void main(String[] args) {
9         Rectangle r = new Rectangle(3, 4);
10        Rectangle s = new Rectangle(2, 5);
11        System.out.println(r.getArea());
12        System.out.println(s.getCircumference());
13    }
14
15 }
```



18



RQ2.17. Show that a UML object diagram does not duplicate information present in its class diagram

Discuss: How to approach answering this question

19



RQ2.18. How is an app different from an application?

RQ2.22. (riff) Driving a car requires knowing how to put the key in and turn on the ignition. Does this knowledge break the encapsulation?

20



Exercise2.12: Consider the following code. Identify the compile-time error, the run-time error, and a logic error.

```
// this code should compute the arithmetic
mean of the variables x, y, z
int x = 6 / 2;
int y = 12 / (x - 3);
int z = -3;
double mean = x + y + z) / 3;
```

21



RQ2.26. Explain the difference between correcting a compile-time error and debugging.

22



Q. What is the definition of a utility class?
Is it a class that contains only static methods?
Is it a class that cannot be instantiated?

Background Q's:

Can a class that contains only static methods be instantiated?

Can a class that cannot be instantiated contain non-static methods?

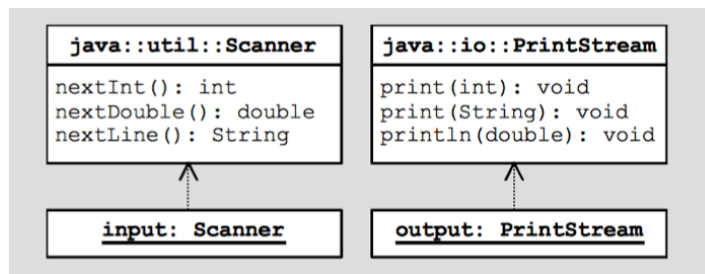
23



Ready-Made I/O (sec 2.2.5)

```
PrintStream output = System.out;  
output.println("hi");
```

```
Scanner input = new Scanner(System.in);  
int value = input.nextInt();
```



24



Q. Can a client make use of the services of a class even before the class is implemented?

Surprisingly, YES! (at partially)
Here is an example:

```
PrintStream output = System.out;
List peopleToPhone; // need to import java.util.List
peopleToPhone = null; // need to instantiate a list object!!!
peopleToPhone.add("Charles Oakley");
peopleToPhone.add("Joffrey Baratheon");
peopleToPhone.add("Boromir");
output.print("# of people who I need to call is:");
output.println(peopleToPhone.size());
```