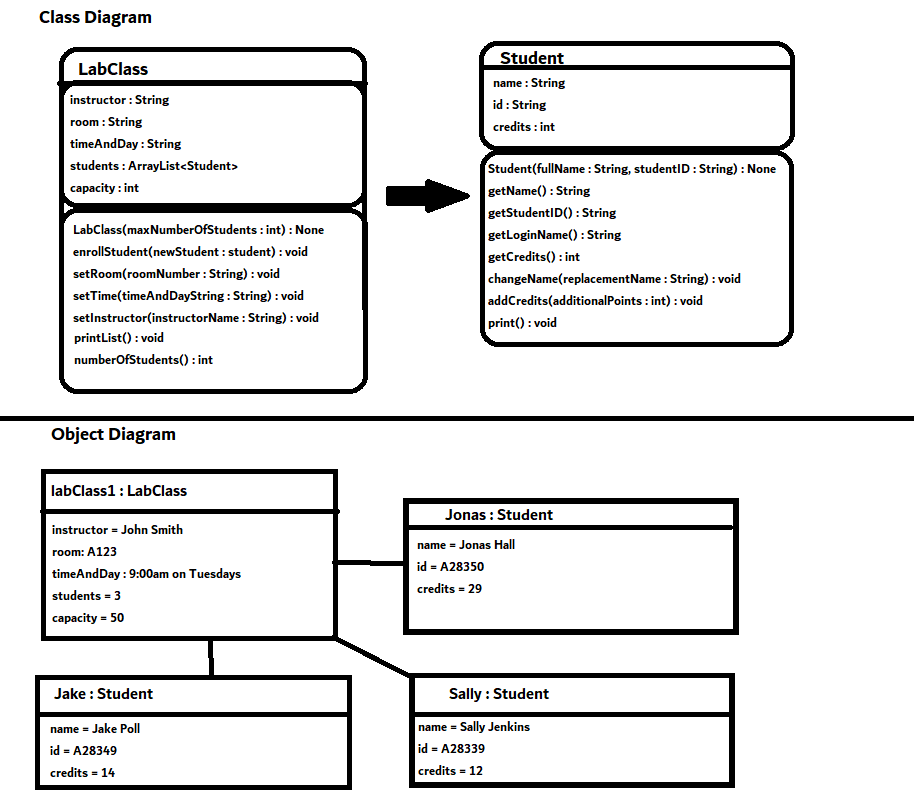
Lab04\_BlueJ – Michael Ng

**3.1 ------------------------------------------------------\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



The Class Diagram defines the fields, methods, parameters, and return types for each class, while Object Diagrams create instances and define their fields as described in the Class Diagram.

3.2

The Class Diagram can change if a method, field, parameter, or return type is altered for a class. It can only change through user editing.

3.3

An Object Diagram can change based on user input. Objects can change based on values the user inputs into the program.

3.4

private Instructor tutor;

3.7-

3.8

“Error: non-static method getValue() cannot be referenced from a static context”

3.9

“Error: '.class' expected”

Remove the “int ” in the parameter, so that it looks like “nd.setValue(5);”

3.10

Nothing happens, since it cannot go above the limit or be below 0. This solution works fine. A slightly better solution could be to set the value to be 0 or the limit, depending on which is closer to the user’s input. That way, it is closer to the user’s expectations.

3.11

The “>=” test would accept 0 as user input, as opposed to “>”.

3.12

replacementValue would only need to satisfy one condition in order to execute the code inside the if statement.

3.13

* ! false
* (2 > 2) || (4 == 4) && (1 < 0)
* (34 != 33) && ! false

3.14

((a && b) || (!a && !b))

3.15

!((a && b) || (!a && !b))

3.16

!(!a || !b) [De Morgan’s Law]

3.17

No, it assumes that the number is positive and is not greater than 99 (is not 3 digits). For a number display with limit 800, it will return the 3-digit number instead of 2 digits.

3.18

There is no difference.

3.19

“value” is a name, so it is replaced with its actual value, returning value’s value.

3.20

The Modulo Operator divides the value until it can no longer be divided, returning the remainder when completed.

3.21

2.

3.25

The increment method adds one to the value, and then preforms modular division on the new value. If the new value is equal to limit, then value is reset back to 0. Else, it retains its new value.

**3.26**

public void increment()

{

value = value + 1;

if(value >= limit)

{

value = 0;

}

}

The solution with the modulo operator is more efficient and easier to understand.

3.27

The constructor automatically sets the value to 0. The parameter in the method call defines the Rolloverlimit.

3.29

It would need to be called 60 times. Another method could be using the setTime() method, passing an int hour and int minute parameter in order to change the time.

3.30

Rectangle window = new Rectangle(int numOne, int numTwo);

3.31

The second constructor takes in 2 parameters, and sets the fields hours and minutes to the parameter.

3.32

One constructor requires parameters while the other doesn’t. The second constructor sets the time using the setTime() method before calling the updateDisplay() method.

3.33

p1.print(“Memes.jpg”, true);

p1.print(“Shrek.gif”, false);

p1.getStatus(5);

p1.getStatus(1);

3.34

2 squares, 1 triangle, and 1 circle objects are created in the constructor.

3.35

moveHorizontal();

moveVertical();

changeSize();

makeVisible();

3.36

There are no internal method calls in the Picture class.

**3.37**

public void draw()

{

if(!drawn) {

setColor();

wall.moveHorizontal(-140);

wall.moveVertical(20);

wall.changeSize(120);

wall.makeVisible();

window.moveHorizontal(-120);

window.moveVertical(40);

window.changeSize(40);

window.makeVisible();

roof.changeSize(60, 180);

roof.moveHorizontal(20);

roof.moveVertical(-60);

roof.makeVisible();

sun.moveHorizontal(100);

sun.moveVertical(-40);

sun.changeSize(80);

sun.makeVisible();

drawn = true;

}

}

public void setColor()

{

wall.changeColor("red");

sun.changeColor("yellow");

}