(Lab 8):
Containers: Objects & Classes II
Due: 15th November 2023, 11:59 PM

P3 Solutions limited in scope to:		
● P1 Concepts ● P2 Concepts ● P3 Concepts ● P4 Concepts	● Designing Objects ○ instance variables ○ instance methods ○ constructors ○ UML diagrams	● Using Objects ○ Instantiation ○ Storing objects into variables ○ invoking instance methods ○ using Java API

Submission Rules:

1. Submissions must be zipped into a **handin.zip** file. Each problem must be implemented in its own class file. Use the name of the problem as the class name.

2. You must use standard input and standard output for ALL your problems. It means that the input should be entered from the keyboard while the output will be displayed on the screen.

3. Your source code files should include a comment at the beginning including your name and that problem number/name.

4. The output of your solutions must be formatted exactly as the sample output to receive full credit for that submission.

5. Compile & test your solutions before submitting.

6. Each problem is worth up to 10 points total.

7. This lab is worth a max total of 30 points.

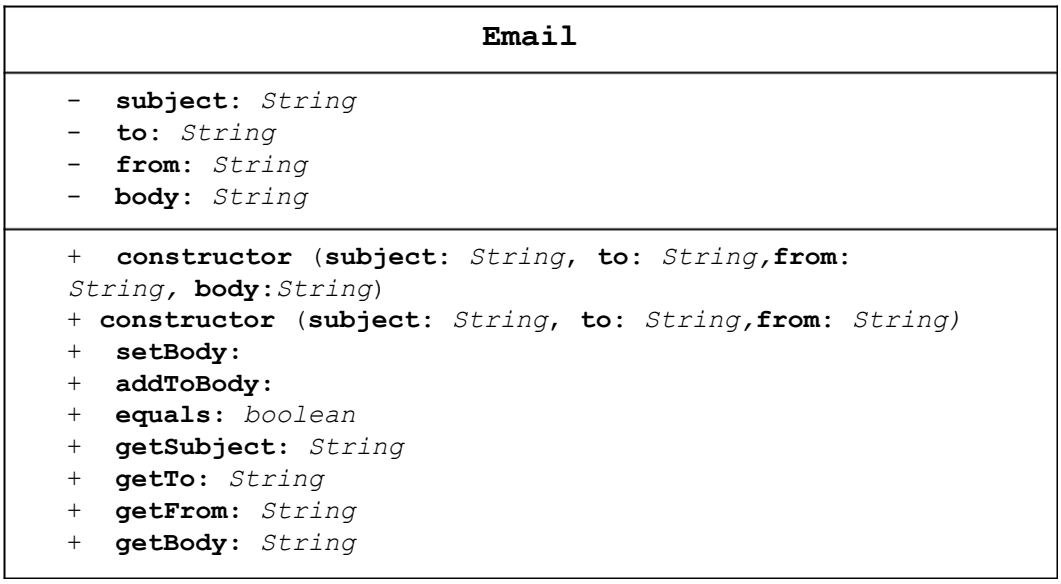
8. Submission:

- You have unlimited submission attempts until the deadline passes
- You'll receive your lab grade immediately after submitting
- You must submit a report (.pdf preferred) (GitLab) showing the result of your successful run for each of the solved problems.

Problem 1: Email (10 points)

(Software design) Emails.

UML Class Diagram:



Email Constructor Summary:

Constructor	Description
`Email(String subject, String to, String from, String body)`	Creates an email with a subject, a to, a from, and a body.
`Email(String subject, String to, String from)`	Creates an email with a subject, a to, and a from. Set the body String as the empty String “”.

Email Method API:

Modifier and Type	Method and Description
`void`	**`setBody`**`(String body)` Sets the email’s body to the given String.
`void`	**`addToBody`**`(String text)` Appends the text String to the end of this email’s body String.
`boolean`	**`equals`**`(Email other)` Returns true if the email has the same subject, to, from, and body Strings.
`String`	**`getSubject()`** `Returns this email’s subject.`
`String`	**`getTo()`** `Returns this email’s to.`
`String`	**`getFrom()`** `Returns this email’s subject.`
`String`	**`getBody()`** `Returns this email’s body.`

`String`	**`toString()`**
	`Returns this email as a String separated by semicolons.`
	`Example: “From: Me; To: Joe; Subject: Hw; Body: Hello”`

Tester Files:

Use the *EmailTester.java* file to test your implementation. Compare your results with the *email.txt* file.

Sample Method Calls	Sample Method Results

`Email x = new Email("Hw", "Joe", "Me", "Hello");`
`Email y = new Email("Hw2", "Tony", "You", "Hi");`
`x.equals(y);`
`x.toString()`
	`"false"`
	`"From: Me; To: Joe; Subject: Hw; Body: Hello"`

Problem 2: RGBColor (10 points)

(Software design) RGBColor. Red Green Blue color representations.

UML Class Diagram:

RGBColor
- **red**: *int* - **green**: *int* - **blue**: *int*
+ **constructor** (**red**: *int*, **green**: *int*, **blue**: *int*) + **equals**: *boolean* + **toString**: *String* + **toHex**: *String* + **getRed**: *int* + **getGreen**: *int* + **getBlue**: *int*

`RGBColor` Constructor Summary:

Constructor	Description
`RGBColor(int red, int green, int blue)`	Creates an RGBColor object with the given red, green, blue values.

`RGBColor` Method API:

Modifier and Type	Method and Description
`boolean`	**`equals`**`(RGBColor other)` Returns true if the RGBColor's colors all are the same values. Returns false otherwise.
`String`	**`toString`**`()` Returns the String representation of the RGBColor object as "rgb(red, green, blue)" where red, green, and blue are their respective integer values.
`String`	**`toHex`**`()` Returns the String representation of the RGBColor object as "#redgreenblue" where red, green, and blue are the hex values.

int	**getRed()** Returns this object's red value.
int	**getGreen()** Returns this object's green value.
int	**getBlue()** Returns this object's blue value.

Facts

- **String.format() converts an integer to its hexadecimal equivalent using "%x"**
- **String.format() allows us to pad the empty space with zeros by a set amount (here the amount is 2) by saying %02x**
  - **https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html**
  - **Example: String.format("%x", 15) would return "f"**
  - **Example: String.format("%02x", 15) would return "0f"**

Tester Files:

Use the *RGBColorTester.java* file to test your implementation. Compare your results with the *rgbcolor.txt* file.

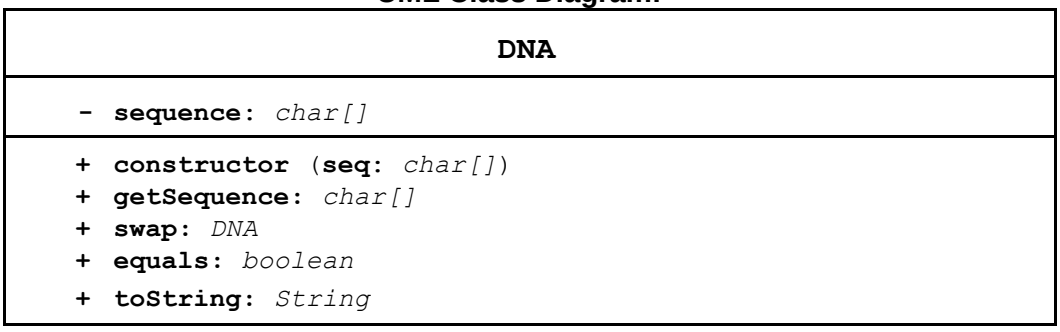
Sample Method Calls	Sample Method Results

`RGBColor x = new RGBColor(99, 16, 0);`
`RGBColor y = new RGBColor(0, 0, 0);`
`boolean t = x.equals(y);`
`String dec = x.toString();`
`String hex = x.toHex();`
	"false"
	"rgb(99,16,0)"
	"#631000"

Problem 3: DNA (10 points)

(Bioinformatics)

UML Class Diagram:



DNA Constructor Summary:

Constructor	Description
`DNA(char[] sequence)`	Creates a new DNA object with the supplied sequence array.

DNA Method API:

Modifier and Type	Method and Description

char[]	**getSequence()** Returns this DNA object's sequence.
DNA	**swap**(DNA other, int swapPoint) Returns a new DNA object with this DNA object's sequence before the swap point and the other DNA object's sequence from the swap point onward.
boolean	**equals**(DNA other) Returns true if this and the other DNA object's sequences match.
String	**toString()** Returns the sequence as a line of characters.

Facts

- **Our swap will take this DNA object and the other DNA object and create a new DNA object with part of the sequence from this (the indexes up until the swap point) and part of the sequence from the other (the indexes at and following from the swap point). We are assuming both DNA object's sequences are of the same length.**

Tester Files:

Use the *DNATester.java* file to test your implementation. Compare your results with the *dna.txt* file.

Sample Method Calls	Sample Method Results
char[] first = {'a', 't', 'c', 'g'}; DNA d1 = new DNA(first); char[] second = {'t', 'c', 'a', 'a'} DNA d2 = new DNA(second); String out = d1.toString(); DNA nextDNA = d1.swap(d2, 2); String newOut = nextDNA.toString();	"atcg" "ataa"