East West University
Department of Computer Science and Engineering

CSE 110: LAB 04 (Handout)
Course Instructor: Dr. Mohammad Rezwanul Huq

Introducing Classes and Objects

Lab Objective

Familiarize students with the implementation of classes and instantiation of objects accordingly.

Lab Outcomes

After completing this lab successfully, students will be able to:

1. **Understand** and **Apply** the concepts of classes and objects.
2. **Write** the definition of a class as per the specification and **create** objects accordingly using constructors.
3. **Write** various instance methods performing different actions on the objects of a class.

Psychomotor Learning Levels

This lab involves activities that encompass the following learning levels in psychomotor domain.

Level	**Category**	**Meaning**	**Keywords**
P1	Imitation	Copy action of another; observe and replicate.	Relate, Repeat, Choose, Copy, Follow, Show, Identify, Isolate.
P2	Manipulation	Reproduce activity from instruction or memory	Copy, response, trace, Show, Start, Perform, Execute, Recreate.

Lab Activities

A. Writing Class Definition

- Define a class 'Icecream' that has the following instance variables:
    a. icecreamType (String)
    b. icecreamCompany (String)
    c. icecreamPrice (double)
- At the beginning, the access specifier is default (no public or private while declaring them). Later, we will make them private and the difference will be discussed in the class.
- Define appropriate constructors for 'Icecream' class.
- Define the following instance methods:
    a. toString(): returns a string containing an icecream information.
    b. equals(Icecream I): returns true if price of caller object and callee object are same; false, otherwise.
    c. compareTo(Icecream I): returns 1 if price of caller object is higher, 0 if both prices are the same, -1 otherwise.

B. Demonstrating Class Functionalities

- Define the Main (Driver) class that has the main() method.
- Create three icecream objects (no array) using both constructors.
- Print their values.

- Now, call equals() and compareTo() methods using these objects. Show your result to the instructor.

C. Introducing private access specifier and Setters (Mutators) and Getters (Accessors)

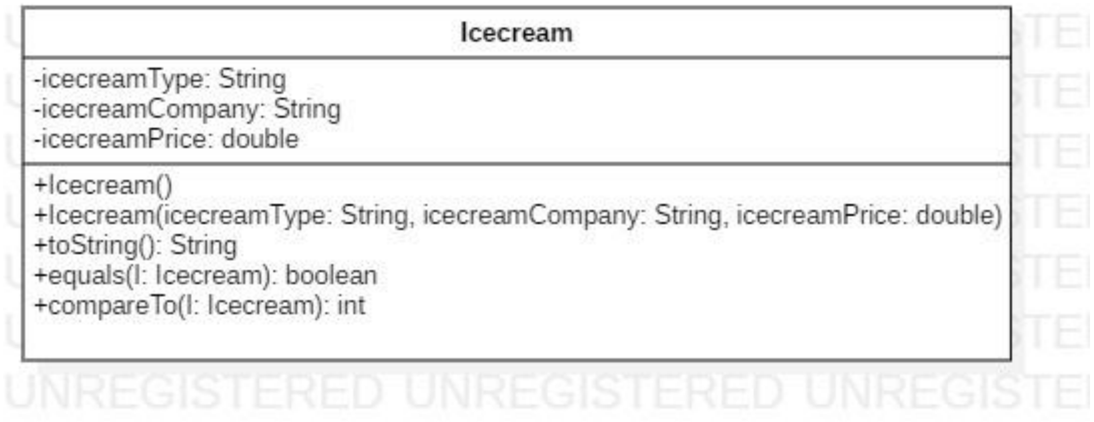
- While declaring instance variables, using private access modifier indicates that the instance variable cannot be accessed from outside the class definition. It is helping to achieve data hiding that states that the data of an object must not be accidentally modified or updated.
- Therefore, to access the private instance variables from outside the class, we need to include a few more instance methods known as setters and getters.
- **<u>Setters are used to set the value.</u>**
```
void setIcecreamType(String icecreamType){
    this.icecreamType = icecreamType;
}
```
- **<u>Getters are used to return the value.</u>**
```
String getIcecreamType(){
    Return this.icecreamType;
}
```

- **Modify your Icecream class definition accordingly by including setters and getters.**

D. Introducing Class Diagram

- Specifications of a class can be also expressed using a class diagram.
- Class diagram is a standard way to represent a class.
- The above-mentioned Icecream class can be represented using the following diagram.

.


Icecream
-icecreamType: String
-icecreamCompany: String
-icecreamPrice: double
+Icecream()
+Icecream(icecreamType: String, icecreamCompany: String, icecreamPrice: double)
+toString(): String
+equals(I: Icecream): boolean
+compareTo(I: Icecream): int

Lab 04: Practice Problems

Lab04_Problem01:

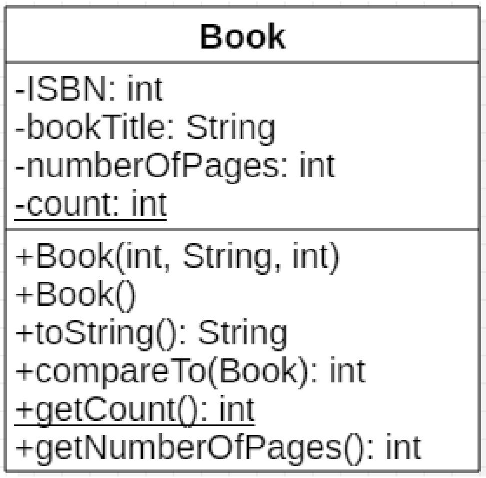
Based on the ‘Icecream’ class as defined before, now create an array of Icecream type objects and get the input from the user to initialize those objects. The size of array must be at least 5.

Now, write a static method searchByCompany in the Main class of the previous implementation, which takes a String parameter representing company name and then prints all icecream’s information manufactured by that company.

Lab04_Problem02:

Write a program in Java that follows the specifications as given below.

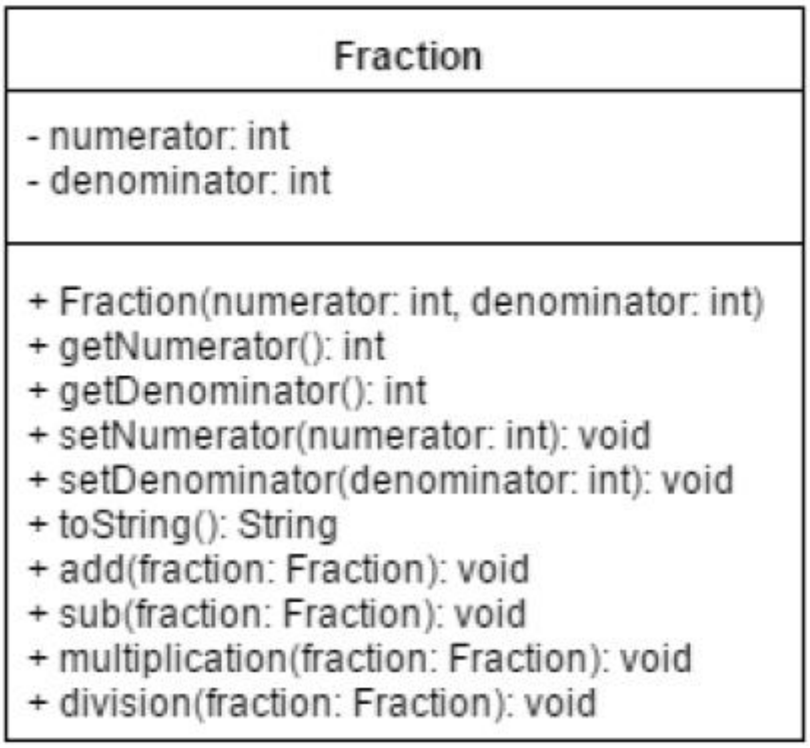
- Define a class **Book** as shown in the following figure.



- Create your *Main* class and the *main()* method. Define an array of Book type objects of size 5.
- Instantiate these Book objects by taking the inputs from the user.
- Print all Book objects’ data using a static method *displayAll()*.
- Invoke *compareTo()* method to compare any two Book objects based on their pages. If caller object’s number of pages is greater than callee object’s number of pages, compareTo() returns 1; if both pages are same, the method returns 0 and -1 otherwise. Make sure to print the returned value.
- Define a static method *isHeavier()* within the *Main* class that takes a Book object as input parameter and returns true if the Book’s number of pages is greater than 500; false, otherwise. Add appropriate code into your main() method to demonstrate its functionalities.

Lab04_Problem03:

Implement the following class ‘Fraction’ and test its methods.



```
Fraction
- numerator: int
- denominator: int
+ Fraction(numerator: int, denominator: int)
+ getNumerator(): int
+ getDenominator(): int
+ setNumerator(numerator: int): void
+ setDenominator(denominator: int): void
+ toString(): String
+ add(fraction: Fraction): void
+ sub(fraction: Fraction): void
+ multiplication(fraction: Fraction): void
+ division(fraction: Fraction): void
```

void add(Fraction fraction)
Adds two Fraction objects and **stores the result** into **calling object**. This is how addition is performed for fractions:
1 / 4 + 3 / 5 = 1 * 5 + 3 * 4 / 4 * 5 = 17 / 20

String toString()
Returns the value of the fraction in 1 / 2 format where 1 is numerator and 2 is denominator.

Now write a test program, take two Fraction objects. Print both of them. Test add, sub, multiplication and division methods. Print calling object after each method call.