# 44-542 Lab Activity

# Classes, Constructors, and the Scanner Class

***IMPORTANT****: The exercises on this worksheet do not require the use of an* ***if*** *statement or any kind of loop. Do not use loops or an* ***if*** *statement in your solutions.*

**Exercise 1:**

1. Create a class name **StudentFee**. This class provides methods for calculating tuition fee, scholarship, quarterly health insurance, total fee, and the minimum due that a student need to pay. It is described below.
2. Attributes are given in this table. Do not add any instance variables beyond those shown here.

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Identifier |
| Number of credit hours | **int** | **creditHours** |
| Fee per credit hour | **double** | **feePerCreditHour** |
| Scholarship for enrolling | **double** | **scholarshipAmount** |
| Health Insurance for one year | **double** | **healthInsurancePerAnnum** |

1. Constructor: There will be one constructor with parameters, with the following prototype. The parameters are used to set the values of the instance variables.

**public StudentFee(int creditHours, double feePerCreditHour, double scholarshipAmount, double healthInsurancePerAnnum)**

1. Constructor: There will be one no-arg constructor, with the following prototype, and with an empty body – that is, it will do nothing.

**public StudentFee()**

1. Methods: Methods are described below. The setters do not return a value. All other methods return type **double**, except for the **toString** method. Do not add any public methods beyond those shown here.
   1. There is a setter for each of the instance variables. Use the standard naming convention: **set**, followed by the name of the variable, with the first letter of the name capitalized – **setCreditHours** for example.
   2. **tuitionFee()**: Returns the tuition fee of the student. The tuition fee depends on the number of credit hours student enrolled in.

*Example: If* ***creditHours*** *is 6 and* ***feePerCreditHour*** *is 593.49, the tuition fee is $3590.94. (6 \* 593.49 = 3590.94)*

* 1. **getScholarshipAmount()**: Returns the scholarship amount.
  2. **quaterlyHealthInsurance()**:returns the quarterly insurance cost.

*Example: If* **healthInsurancePerAnnum** *is $1200.0, then the quarterly insurance is ( healthInsurancePerAnnum \* 4)/12.*

* 1. **totalFee()**:Returns the total feel amount of the student. This is equal to the tuition fee amount minus scholarship amount plus quarterly health insurance amount.

*Example: If* ***creditHours*** *is 6,* ***feePerCreditHour*** *is $593.49,* ***scholarshipAmount*** *is $1000, and* ***healthInsurancePerAnnum*** *is $1100, then the total fee for a student is (creditHours \* feePerCreditHour) –scholarshipAmount + ((healthInsurancePerAnnum \*4)/12).*

* 1. **minimumDue()**: Returns the minimum fee a student should pay.

*Example: Assume a* ***creditHours*** *of 6,* ***feePerCreditHour*** *is $593.49,* ***scholarshipAmount*** *is $1000, and* ***healthInsurancePerAnnum*** *is $1100.00, then the total fee for a student is totalFee = ((creditHours \* feePerCreditHour) -scholarshipAmount+(( healthInsurancePerAnnum \*4)/12)). Then the minimum due is totalFee/2.*

* 1. **toString()**: Returns a string representation of a student fee. For a student with a **creditHours** of 6, a ***feePerCreditHour***of *$593.49*, ***scholarshipAmount is* $**1000, and ***healthInsurancePerAnnum*** *is* $1100.00, the following string would be returned. (Hint: Use "\n" to include a line feed in your output.)

**Credit hours: 6**

**Fee per credit hour: $ 593.49**

**Scholarship amount: $ 1000.00**

**Health insurance amount per annum: $ 1100.00**

1. Include Javadoc comments, using the **@author**, **@param**, and **@return** annotations when appropriate. Remember that for each method, the first line of the Javadoc comment should be a brief description of the method.
2. Generate documentation for your project by clicking on **Run** from the NetBeans menu bar and then selecting **Generate Javadoc**. The documentation will be placed in a **javadoc** subfolder of the **dist** subfolder inside your project folder. You can view the documentation created by opening **index.html**.
3. Create a class named **StudentFeeDriver:** This class has a method **main** and uses a **Scanner** object to get the input from the console. An outline of the driver class is given here, followed by a sample run. Look at the sample run for details of the program output. When you run your program, it must produce the same output, including prompts for input and labels for output:
   1. Declare and initialize a **Scanner** object to read from the keyboard.
   2. Prompt for values for credit hours, fee per credit hour, scholarship amount, and the health insurance amount per annum. Read each value using the **Scanner** object you created in the previous step.
   3. Use the input values to create a new **StudentFee** object named **student1**.
   4. Print **student1**, using the **toString** method.
   5. Print, with appropriate labels, the values returned by **tuitionFee()**,

**getScholarshipAmount()**,**quaterlyHealthInsurance()**,

**totalFee(), and minimumDue()**.

* 1. Create a new **StudentFee** object named **student2** using the no-arg constructor.
  2. Print **student2**, using the **toString** method.
  3. Use the setter methods to set the values of **creditHours**, **feePerCreditHour**, **scholarshipAmount**, and **healthInsurancePerAnnum** to 9, 594.90, 1110.00, and 1202.10, respectively. Do not input these values. Hard-code them as parameters in the calls to the setter methods.
  4. Print **student2**, using the **toString** method.
  5. Print, with appropriate labels, the values returned by **tuitionFee()**,

**getScholarshipAmount()**,**quaterlyHealthInsurance()**,**totalFee(),and minimumDue()**.

**Sample Run (user input is in red):**

**Enter the Number of Credit Hours Enrolled In:6**

**Enter the fee amount per credit hour:593.49**

**Enter the scholarship amount:1000.00**

**Enter the health insurance amount per annum:1100.00**

**Student1 Information**

**Credit Hours: 6**

**Fee per credit Hour: $593.49**

**Scholarship Amount: $1000.0**

**Health Insurance Amount Per Annum: $1100.0**

**Student1 tuition fee :$3560.94**

**Student1 fee after scholarship: $1000.0**

**Student1 health insurance amount for 4 months: $366.6666666666667**

**Student1 total fee: $2927.6066666666666**

**Student1 minimum due amount: $1463.8033333333333**

**Student2 Fee Information:**

**Credit Hours: 0**

**Fee per credit Hour: $0.0**

**Scholarship Amount: $0.0**

**Health Insurance Amount Per Annum: $0.0**

**New Student2 Information:**

**Credit Hours: 9**

**Fee per credit Hour: $594.9**

**Scholarship Amount: $1110.0**

**Health Insurance Amount Per Annum: $1202.1**

**Student2 tuition fee: $5354.099999999999**

**Student2 fee after scholarship: $1110.0**

**Student2 health insurance amount for 4 months: $400.7**

**Student2 total fee: $4644.799999999999**

**Student2 minimum due amount: $2322.3999999999996**

**Exercise 2 (Optional):** Want some more practice? Try this problem.

1. Create a class named **Square**. This class should NOT contain a main method. An object of type **Square** has one attribute **side** of type **double**.
2. Define a constructor with one parameter for the initializes the value of the attribute.
3. Define a no-arg constructor that initializes the value of **side** to 0.0.
4. Define getter and setter methods for the attribute **side**.
5. Define a method **getArea()** that returns the area of the square. This method has no parameters and returns a value of type **double**.
6. Define a method **calculateDiagonalLengthOfSquare()** that returns the diagonal length of the square. This method has no parameters and returns a value of type **double**.
7. Define a method **calculatePerimeterOfSquare**() that returns the perimeter of the square. This method has no parameters and returns a value of **double**.
8. To test your class, create a new class named **SquareTester**. Details for writing this class are specified within the provided JavaDoc for **SquareTester**, which are included at the end of this document. In the sample run below, note that the first object is created using the default **Square** constructor (and then has its side set to 5.00), while the second object is created by passing the value 10.00 (read from the console) to the **Square** constructor with one parameter.

**Sample Run:**

**Initial side of default square is 0.0**

**The side of the square is:5.0**

**The area of the square is:25.0**

**The diagonal length of the square is:7.0710678118654755**

**The perimeter of the square is:20.0**

**Please enter the side of a square:10.00**

**The side of the square is:10.0**

**The area of the square is:100.0**

**The diagonal length of the square is:14.142135623730951**

**The perimeter of the square is:40.0**

1. When you are certain your class is correct, add Javadoc comments for the class, each constructor, and each method.
2. Generate documentation for your project by clicking on **the Run** from the NetBeans menu bar and then selecting **Generate Javadoc**. The documentation will be placed in a **javadoc** subfolder of the **dist** subfolder inside your project folder. After generating the documentation, open the **index.html** file that is created. Documentation for both **Square** and **SquareTester** should be included. The documentation for **SquareTester** should look like the documentation included in this document. For both **Square** and **SquareTester**, study the javadoc comments included in the code and see how that information is displayed in the generated documentation.

## Class SquareTester

* java.lang.Object
  + square.SquareTester

public class SquareTester

extends java.lang.Object

This class is used to test the implementation of class Square.

* + ***Constructor Summary***

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [SquareTester](file:///C:\Users\S521805\Documents\NetBeansProjects\ObjectsLab02\dist\javadoc\square\SquareTester.html#SquareTester--)() |

* + ***Method Summary***

|  |  |
| --- | --- |
| **All Methods**[**Static Methods**](javascript:show(1);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| static void | [main](file:///C:\Users\S521805\Documents\NetBeansProjects\ObjectsLab02\dist\javadoc\square\SquareTester.html#main-java.lang.String:A-)(java.lang.String[] args)  This method instantiates and outputs two different Square objects. |
| static void | [outputSquare](file:///C:\Users\S521805\Documents\NetBeansProjects\ObjectsLab02\dist\javadoc\square\SquareTester.html#outputSquare-square.Square-)([**Square**](file:///C:\Users\S521805\Documents\NetBeansProjects\ObjectsLab02\dist\javadoc\square\Square.html) outputSquare)  This method prints the side, area, diagonal length and perimeter of the specified Square object using four output messages. |

* + - **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

* + ***Constructor Detail***
    - **SquareTester**

public SquareTester()

* + ***Method Detail***
    - **main**

public static void main(String[] args)

{

// instantiate a Square class using the default

constructor and name it as square

// Print the side of the square

// set the side of the square with given value

// Call the method outputSquare() using square object

// Read the side of the square from the console using Scanner

// Set the side of the square with new value using setter

// Call the method outputSquare() using square object

}

* + - **outputSquare**

public static void outputSquare([Square](file:///C:\Users\S521805\Documents\NetBeansProjects\ObjectsLab02\dist\javadoc\square\Square.html) outputSquare)

This method prints the side, area, diagonal length and perimeter of the specified Square object using four output messages.

Parameters:

outputSquare - The square object to output to the console.