Kingdom of Saudi Arabia
Ministry of Education
Taibah University
College of Computer Science & Engineering
Department of Computer Science

CS 112 Student Lab Manual

المملكة العربية السعودية
وزارة التعليم
جامعة طيبة
كلية علوم وهندسة حاسب الآلي
قسم علوم الحاسب الآلي

Date:

Lab Manual

Programming 2

CS 112

STUDENTS

Student ID:

Student Name:

January 2019

Acknowledgment

I would like to thank Dr. Abdullah Alsaeedi for his idea of preparing a lab manual that could help improving the programming skills of CS students.

Also, I would like to thank the following members of the lab manual committee for the high level of professionalism and dedication they had shown in preparing and writing the lab manual:

- Dr. Mutaz Abusara
- Dr. Hamza Ghandorh
- Dr. Reyadh Alluhaibi
- Mrs. Modhi AlSobeihy
- Mr. Raja Al-hejaili

Chair of lab manual committee

Dr. Barzaiq

January 2019

STUDENTS

Declaration

The resources used to prepare and write the lab manual are:

- The academic experience of the members of the lab manual committee.

- The textbook of Y. Daniel Liang, titled “Introduction to Java programming”, tenth edition.

STUDENTS

Lab Instructions

- You are not allowed to attend the lab without your lab manual.
- Do not mess with the lab computers or accessories.
- Do not mess with software installed on computers.
- Food and beverages are not allowed in the lab.
- No obnoxious or belligerent behavior will be tolerated.
- Turn off your mobile phone or put it on silent mode.

STUDENTS

ASSIGNMENT DECLARATION FORM

(Group Assignment)

❖ **By submitting this work and signing this document, we declare that:**

1. This assignment is our own original work.
2. No part of this work has been copied from any other source or person except where explicitly stated otherwise by reference or acknowledgment.
3. No part of this work has been previously submitted for assessment at this or any other institution.
4. We may be subject to student discipline processes in the event of an act of academic misconduct by us including an act of plagiarism or cheating.

Course Code	Course Title

Student ID	**Student Name**	**Signature of student**
Student ID	**Student Name**	**Signature of student**
Student ID	**Student Name**	**Signature of student**
Date of signature		

STUDENTS

ASSIGNMENT DECLARATION FORM

(Individual Assignment)

❖ **By submitting this work and signing this document, I declare that:**

1. This assignment is my own original work.
2. No part of this work has been copied from any other source or person except where explicitly stated otherwise by reference or acknowledgment.
3. No part of this work has been previously submitted for assessment at this or any other institution.
4. I may be subject to student discipline processes in the event of an act of academic misconduct by me including an act of plagiarism or cheating.

DENTS

Course Code	
Course Title	
Student ID	
Student Name:	
Signature of student	
Date of signature	

Contents

STUDENTS

STUDENTS

STUDENTS

STUDENTS

Lab 001: Review of Programming 1

Objective:

- This lab is reviewing what you have studied in the first course of programming 1 (CS 111)
- Reviewing how to analyze a code and how to write the code by using different components: Print statement, Input from an user, Primitive types, Variables, Selection instruction, Loops, Methods, Single dimensional array and multi-dimensional array.

Examples:

Example 1- Analyzing a program or project:

1- **Pseudocode** is an artificial and informal language that helps programmers and develops algorithms. <u>**Pseudocode** is very similar to everyday English</u>. Example:

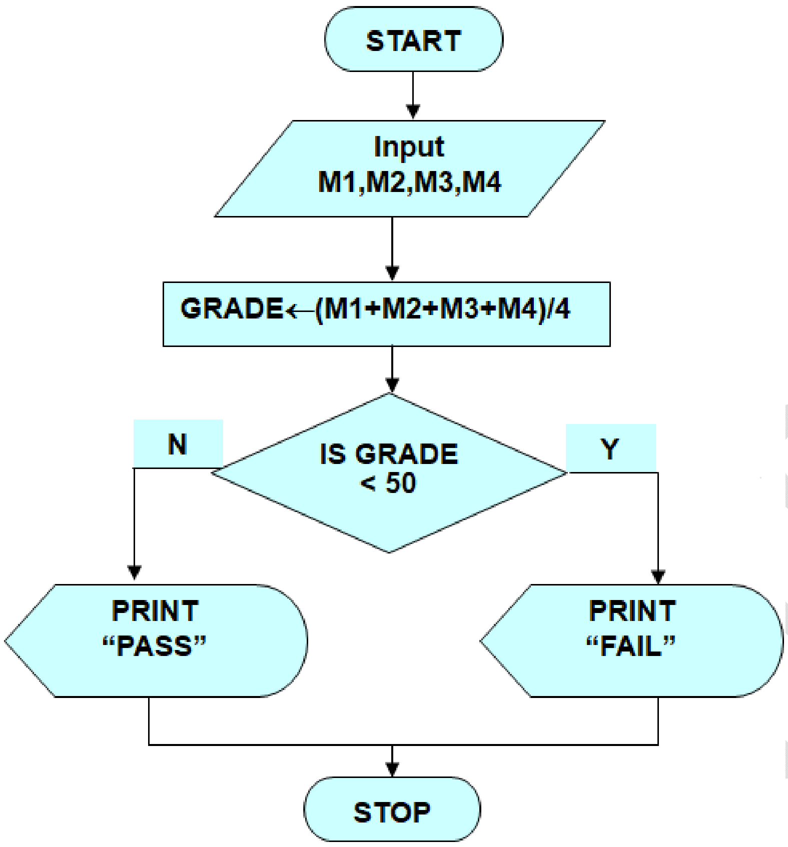
```
■ Input a set of 4 marks
■ Calculate their average by summing and dividing by 4
■ if average is below 50
    Print "FAIL"
  else
    Print "PASS"
```

2- An **algorithm** produces an ordered sequence of steps that describe a solution of the problem. Example:

```
Step 1:  Input M1,M2,M3,M4
Step 2:  GRADE ← (M1+M2+M3+M4)/4
Step 3:  if (GRADE < 50) then
                 Print "FAIL"
         else
                 Print "PASS"
         endif
```

3- A **Flow chart** is graphical representation of the sequence of operations in an information system or program.

STUDENTS


START
Input
M1,M2,M3,M4
GRADE←(M1+M2+M3+M4)/4
IS GRADE
< 50
N
Y
PRINT
"PASS"
PRINT
"FAIL"
STOP

Example 2- Strictly identical arrays for one-dimensional:

Write a test program which asks a user to enter the size of two arrays as well as to fill their elements. After that, the program checks their corresponding elements of two arrays. If these arrays are equal with element by element, the program displays message which is "The two arrays are strictly identical". If not equal, displaying "The two arrays are not strictly identical".

Write an **equals** method that reads two arrays and displays these messages above.

Here are the sample runs.

```
How many elements (the size) in the first array: 6
Enter the elements of the first array: 6 1 6 5 2 5
How many elements (the size) in the second array: 6
Enter the elements of the second array: 6 1 6 5 2 5
The two arrays are strictly identical arrays
```

```
How many elements (the size) in the first array: 6
Enter the elements of the first array: 6 1 6 5 2 5
How many elements (the size) in the second array: 6
Enter the elements of the second array: 6 6 1 5 2 5
The two arrays are not strictly identical arrays
```

```
How many elements (the size) in the first array: 6
Enter the elements of the first array: 6 1 6 5 2 5
How many elements (the size) in the second array: 5
Enter the elements of the second array: 6 1 6 5 2
The size of the two arrays are not equals
```

Here is the code of the example.

```java
import java.util.Scanner;

public class Test {
	/** Main method */
	public static void main(String[] args) {
		// Create Scanner object for input
		Scanner input = new Scanner(System.in);

		// Enter the size of the first array
		System.out.print("How many elements (the size) in the first 
array: ");
		int size = input.nextInt();

		// Prompt the user to enter the first array
				System.out.print("Enter the elements of the first 
array: ");
				int[] first = new int[size];
				for (int i = 0; i < first.length; i++)
					first[i] = input.nextInt();

		// Enter the size of the second array
		System.out.print("How many elements (the size) in the second 
array: ");
		size = input.nextInt();
```

STUDENTS

```
            // Prompt the user to enter the second array
            System.out.print("Enter the elements of the second array: ");
            int[] second = new int[size];
            for (int i = 0; i < second.length; i++)
                  second[i] = input.nextInt();

            // Invoke method
            equals(first, second);
      }

      // Define method
      public static void equals(int[] first, int[] second) {
            if (first.length != second.length)
                  System.out.println("The size of the two arrays are not
equals");

            else
            {
            for (int i = 0; i < first.length; i++) {
                  if (first[i] != second[i])
                  {
                        System.out.println("The two arrays are not
strictly identical arrays");
                        break;
                  }
                  if(i == first.length-1)
                        System.out.println("The two arrays are strictly
identical arrays");
            }
            }
      }
}
```

Exercise - Strictly identical arrays for two-dimensional:

Write a test program as same as the Example 2 but in two-dimensional.

<u>Here are some runs:</u>

```
How many elements (the size) in the first array: 3 3
Enter the elements of the first array:
51 22 25
6 1 4
24 54 6
How many elements (the size) in the second array: 3 3
Enter the elements of the second array:
51 22 25
6 1 4
24 54 6
The two arrays are strictly identical arrays
```

```
How many elements (the size) in the first array: 3 3
Enter the elements of the first array:
51 22 25
6 1 4
```

STUDENTS

```
24 54 6
How many elements (the size) in the second array: 3 3
Enter the elements of the second array:
51 22 25
6 2 4
24 54 6
The two arrays are not strictly identical arrays
```

```
How many elements (the size) in the first array: 3 3
Enter the elements of the first array:
51 22 25
6 1 4
24 54 6
How many elements (the size) in the second array: 2 3
Enter the elements of the second array:
51 22 25
6 1 4
The size of the two arrays are not equals
```

Homework – identical arrays for one-dimensional:

Write a test program which prompts a user to enter the size of two arrays as well as to fill their elements. After that, the program checks the same contents of the elements of the two arrays. If these arrays are equal, the program displays message which is "The two arrays are identical". If not equal, displaying "The two arrays are not identical".
Write an **equals** method that reads two arrays and displays these messages above.

Here are the sample runs.

```
How many elements (the size) in the first array: 6
Enter the elements of the first array: 6 1 6 5 2 5
How many elements (the size) in the second array: 6
Enter the elements of the second array: 6 6 1 5 2 5
The two arrays are identical arrays
```

```
How many elements (the size) in the first array: 6
Enter the elements of the first array: 6 6 6 5 2 5
How many elements (the size) in the second array: 6
Enter the elements of the second array: 6 6 1 5 2 5
The two arrays are not identical arrays
```

```
How many elements (the size) in the first array: 6
Enter the elements of the first array: 6 1 6 5 2 5
How many elements (the size) in the second array: 5
Enter the elements of the second array: 6 1 6 5 2
The size of the two arrays are not equals
```

STUDENTS

Challenge Question - identical arrays for two-dimensional:

Write a test program as same as the homework question but in two-dimensional. Before you start, you should analyze this question by using one of them (Pseudocode, Algorithm, or Flowchart).

Hint: The solution of this challenge is quite similar to the solution of homework question.

Here are some runs:

```
How many elements (the size) in the first array: 3 3
Enter the elements of the first array:
51 22 25
6 1 4
24 54 6
How many elements (the size) in the second array: 3 3
Enter the elements of the second array:
25 22 51
6 1 4
24 54 6
The two arrays are identical arrays
```

```
How many elements (the size) in the first array: 3 3
Enter the elements of the first array:
51 22 25
6 1 4
24 54 6
How many elements (the size) in the second array: 3 3
Enter the elements of the second array:
51 22 25
6 2 4
24 54 6
The two arrays are not identical arrays
```

```
How many elements (the size) in the first array: 3 3
Enter the elements of the first array:
51 22 25
6 1 4
24 54 6
How many elements (the size) in the second array: 2 3
Enter the elements of the second array:
51 22 25
6 1 4
The size of the two arrays are not equals
```

STUDENTS

Lab 002: Chapter 9: OBJECTS AND CLASSES (part 1)

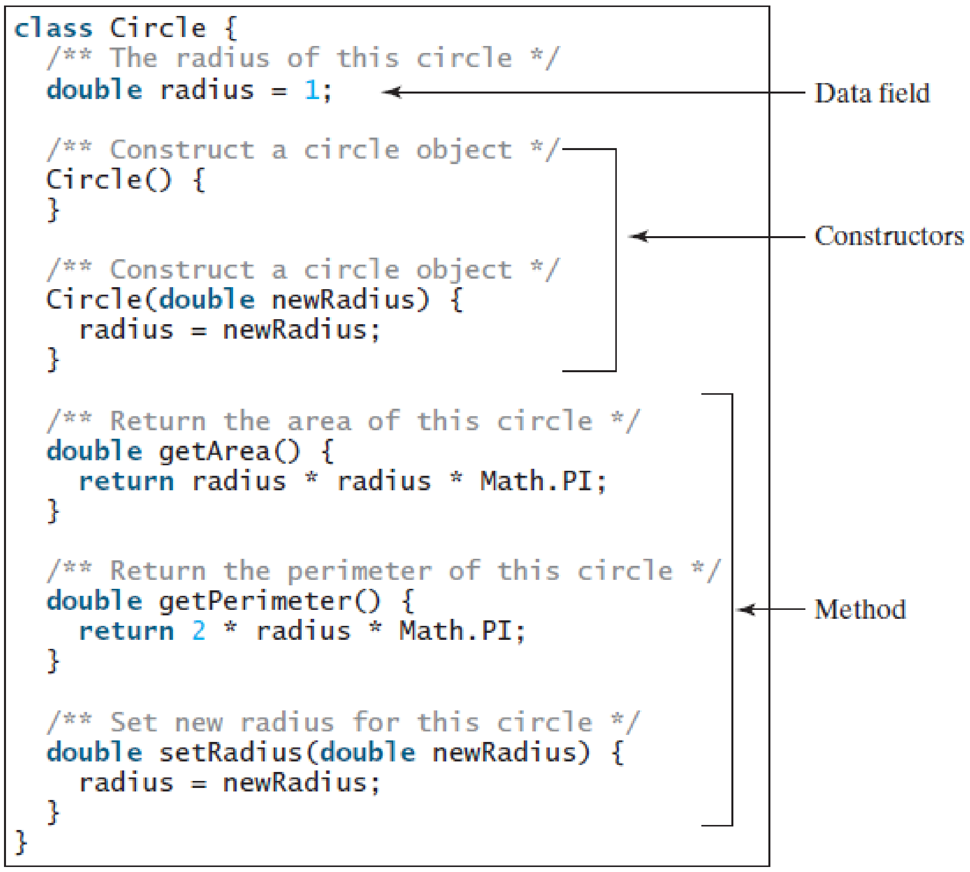
Objective:

■ Design classes and draw UML class diagrams.

■ Implement classes from the UML.

■ Use classes to develop applications.

Example – Circle class:



```
class Circle {
  /** The radius of this circle */
  double radius = 1;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * Math.PI;
  }

  /** Return the perimeter of this circle */
  double getPerimeter() {
    return 2 * radius * Math.PI;
  }

  /** Set new radius for this circle */
  double setRadius(double newRadius) {
    radius = newRadius;
  }
}
```

A class is a construct that defines objects of the same type.

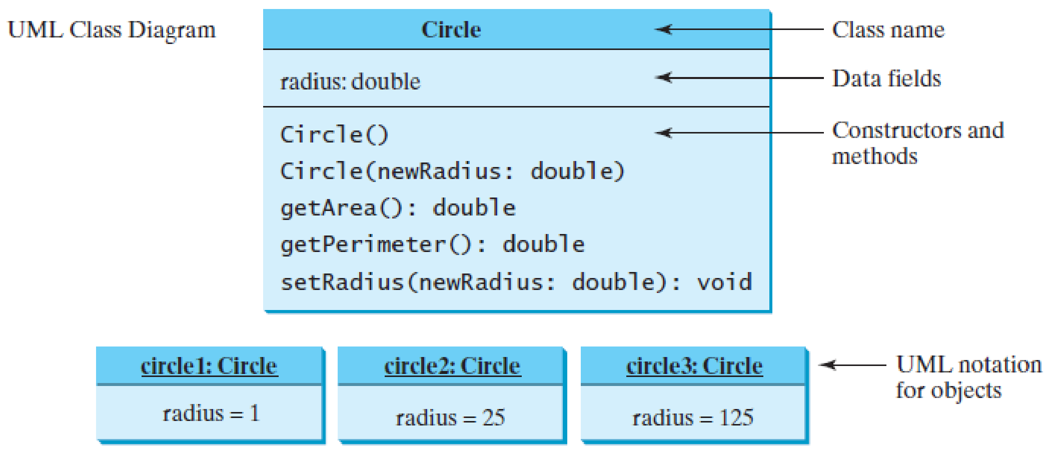


FIGURE 9.4 Classes and objects can be represented using UML notation.

STUDENTS

Exercise - Rectangle class:

Design a class named Rectangle to represent a rectangle. The class contains:

■ Two double data fields named width and height that specify the width and height of the rectangle. The default values are 1 for both width and height.

■ A no-arg constructor that creates a default rectangle.

■ A constructor that creates a rectangle with the specified width and height.

■ A method named getArea() that returns the area of this rectangle.

■ A method named getPerimeter() that returns the perimeter.

Draw the UML diagram for the class and then implement the class. Write a test program that creates two Rectangle objects—one with width 4 and height 40 and the other with width 3.5 and height 35.9. Display the width, height, area, and perimeter of each rectangle in this order.

Homework – Stock and Random classes:

Design a class named Stock that contains:

■ A string data field named symbol for the stock's symbol.

■ A string data field named name for the stock's name.

■ A double data field named previousClosingPrice that stores the stock price for the previous day.

■ A double data field named currentPrice that stores the stock price for the current time.

■ An integer data field named ID that stores the identifier number of stock.

■ A constructor that creates a stock with the specified symbol, name, and ID.

■ A method named getChangePercent() that returns the percentage changed from previousClosingPrice to currentPrice.

Draw the UML diagram for the class and then implement the class. Write a test program that
■ creates a Random object from the Random class (java.util.Random). –We will use this object for the ID number of Stock between (0 to 100)-

■ asks a user for the name and symbol of Stock. So, you should create two string variables in the test program.

■ creates a Stock object with the stock symbol (from the user), the name of symbol (from the user), and ID (from Random object by using nextInt(100) method).
■ asks the user for the previous closing price and the current price. **Note**: Don't create two variables for them. You can use the stock object to assign the values of these two variables from the user.
■ finally, displays the stock name, the stock symbol, the stock ID, and the price-change percentage.

Here is the simple run:

```
Enter the name of Stock: Oracle Corporation
Enter the symbol of Stock: ORCL
Enter the previous closing price: 34.5
Enter the current price: 34.35

Stock name: Oracle Corporation
Stock symbol: ORCL
Stock ID: 59
Price-change percentage: -0.434782608695648
```

Challenge Question – Using GregorianCalendar and Date classes:

(Use the GregorianCalendar class) Java API has the Date class and GregorianCalendar class in the java.util package, which you can use to obtain the current date with specific details like current year, current month, current day of a date and so on.

Write a program to perform these tasks:

1- Display the current year, month, and day.
2- Display the current date by using the toString() method from the Date class.
3- Display the current time –which is used to display the number of milliseconds- by using getTime() method from the Date class. We will use it in the step 17.
4- Display the current year by using get(GregorianCalendar.YEAR) method from the GregorianCalendar class.
5- Display the current year by using get(GregorianCalendar.MONTH) method from the GregorianCalendar class. **Note**: this method -which is used for month- starts from 0 to 11 (0 indicates January). You must take care about this issue for adding 1 in order to display 1 for January.
6- Display the current Day by using get(GregorianCalendar.DAY_OF_MONTH) method from the GregorianCalendar class.

STUDENTS

7- Display the current Week by using get(GregorianCalendar.DAY_OF_WEEK) method from the GregorianCalendar class.
8- Display the current DAY OF WEEK IN MONTH by using get(GregorianCalendar.DAY_OF_WEEK_IN_MONTH) method from the GregorianCalendar class.
9- Display the current Day of Year by using get(GregorianCalendar.DAY_OF_YEAR)
10- Display the current Hour by using get(GregorianCalendar.HOUR)
11- Display the current Hour of day by using get(GregorianCalendar.HOUR_OF_DAY)
12- Display the current MINUTE by using get(GregorianCalendar.MINUTE)
13- Display the current Second by using get(GregorianCalendar.SECOND)
14- Display the current Millisecond by using get(GregorianCalendar.MILLISECOND)
15- Display the current Week of month by using get(GregorianCalendar.WEEK_OF_MONTH)
16- Display the current Week of year by using get(GregorianCalendar.WEEK_OF_YEAR)
17- Finally, use setTime() method which sets your current time (Step 3) and display your date and time using the toString() method after adding these milliseconds (In for loop): 1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000, 1000000000, 10000000000, and 100000000000.

<u>Here is the sample run:</u>

```
Current Date: Mon Jan 07 11:05:17 AST 2019
Current time (the number of milliseconds): 1546848317397
Current Year: 2019
Current Month: 1
Current Day: 7
Current Week: 2
Current DAY OF WEEK IN MONTH: 1
Current Day of Year: 7
Current Hour: 11
Current Hour of day: 11
Current MINUTE: 5
Current Second: 17
Current Millisecond: 470
Current Week of month: 2
Current Week of year: 2


The time after adding 1 millisecond(s) : Mon Jan 07 11:05:17 AST 2019
The time after adding 10 millisecond(s) : Mon Jan 07 11:05:17 AST 2019
The time after adding 100 millisecond(s) : Mon Jan 07 11:05:17 AST 2019
The time after adding 1000 millisecond(s) : Mon Jan 07 11:05:18 AST 2019
The time after adding 10000 millisecond(s) : Mon Jan 07 11:05:27 AST 2019
The time after adding 100000 millisecond(s) : Mon Jan 07 11:06:57 AST 2019
The time after adding 1000000 millisecond(s) : Mon Jan 07 11:21:57 AST 2019
The time after adding 10000000 millisecond(s) : Mon Jan 07 13:51:57 AST 2019
The time after adding 100000000 millisecond(s) : Tue Jan 08 14:51:57 AST 2019
The time after adding 1000000000 millisecond(s) : Sat Jan 19 00:51:57 AST 2019
The time after adding 10000000000 millisecond(s) : Fri May 03 04:51:57 AST 2019
The time after adding 100000000000 millisecond(s) : Wed Mar 09 20:51:57 AST 2022
```

STUDENTS

Lab 003: Chapter 9: OBJECTS AND CLASSES (part 2)

Example – StopWatch class:

Design a class named **StopWatch**. The class contains:

■ Private data fields **startTime** and **endTime** with getter methods.

■ A no-arg constructor that initializes **startTime** with the current time.

■ A method named **start()** that resets the **startTime** to the current time.

■ A method named **stop()** that sets the **endTime** to the current time.

■ A method named **getElapsedTime()** that returns the elapsed time for the stopwatch in milliseconds.

Draw the UML diagram for the class and then implement the class. Write a test program that measures the execution time of sorting 100,000 numbers using selection sort.

Here is the code.

```
public class Test {
      /** Main method */
       public static void main(String[] args) {
              StopWatch stopWatch = new StopWatch();
             java.util.Random r = new java.util.Random();
              int[] numbers = new int[100000];

              for(int i = 0; i < numbers.length; i++) {
                  numbers[i] = r.nextInt(100000);
              }


              selectionSort(numbers);
              stopWatch.stop();

              System.out.println("The execution time of sorting 100,000
numbers took " + stopWatch.getElapsedTime() + " milliseconds");
          }

      /** selectionSort performs a selection sort on an array */
          public static void selectionSort(int[] list) {
              for (int i = 0; i < list.length; i++) {
                  int currentMin = list[i];
                  int currentMinIndex = i;

                  for (int j = i + 1; j < list.length; j++) {
                      if (currentMin > list[j]) {
                          currentMin = list[j];
                          currentMinIndex = j;
                      }
                  }
```

STUDENTS

```java
                  if (currentMinIndex != i) {
                      list[currentMinIndex] = list[i];
                      list[i] = currentMin;
                  }
              }
          }
}


// Implement StopWatch class
class StopWatch {
    private long startTime;   // Start time
    private long endTime;     // End time


    public StopWatch() {
          startTime = System.currentTimeMillis();
    }

    public long getStartTime() {
        return this.startTime;
    }

    public long getEndTime() {
        return this.endTime;
    }

    public void start() {
        this.startTime = System.currentTimeMillis();
    }

    public void stop() {
        this.endTime = System.currentTimeMillis();
    }

    public long getElapsedTime() {
        return getEndTime() - getStartTime();
    }
}
```

STUDENTS

Here is the sample run.

```
The execution time of sorting 100,000 numbers took 2524 milliseconds
```

Here is the UML of StopWatch class.

StopWatch
-startTime: long -endTime: long

+StopWatch()
+getStratTime(): long
+getEndTime(): long
+start()
+stop()

+getElapsedTime(): long

Exercise - Account class:

Design a class named `Account` that contains:

a. A `private int` data field named `id` for the account (default `0`)
b. A `private double` data field named `balance` for the account (default `0`)
c. A `private Date` data field named `dateCreated` that stores the date when the account was created
d. A no-arg constructor that creates a default account
e. A constructor that creates an account with the specified `id` and initial `balance`
f. The accessor and mutator methods for `id` and `balance`.
g. The accessor method for `dateCreated`
h. A method named `withdraw()` that withdraws a specified amount from the account
i. A method named `deposit()` that deposits a specified amount to the account

1. Draw the UML diagram for the `Account` class
2. Implement the `Account` class
3. Use the test program given bellow that creates an `Account` object with an account ID of `1122`, and a `balance` of $20,000. Use the `withdraw()` method to withdraw $2,500, use the `deposit()` method to deposit $3,000, and print the `balance`, and the date when this account was created.

```
public class Test {
  public static void main (String[] args) {
    Account account = new Account(1122, 20000);

    account.withdraw(2500);
    account.deposit(3000);
    System.out.println("Balance is " + account.getBalance());
    System.out.println("This account was created at " +
      account.getDateCreated());
  }
}

Class Account {

  // Implement the class here

}
```

STUDENTS

Homework - Fan class:

Design a class named Fan to represent a fan. The class contains:

■ Three constants named SLOW, MEDIUM, and FAST with the values 0, 1, and 2 to denote the fan speed.

■ A private int data field named speed that specifies the speed of the fan (the default is SLOW).

■ A private boolean data field named on that specifies whether the fan is on (the default is false).

■ A private double data field named radius that specifies the radius of the fan (the default is 4).

■ A string data field named color that specifies the color of the fan (the default is red).

■ The accessor and mutator methods for all four data fields.

■ A no-arg constructor that creates a default fan.

■ A method named toString() that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string “fan is off” in one combined string.

Draw the UML diagram for the class and then implement the class. Write a test program that creates two Fan objects. Assign maximum speed, radius 10, color yellow, and turn it on to the first object. Assign medium speed, radius 4, color red, and turn it off to the second object. Display the objects by invoking their toString method.

STUDENTS

Lab 004: Chapter 10 - OBJECT-ORIENTED THINKING

Objective:

■ To apply class abstraction to develop software (§10.2).

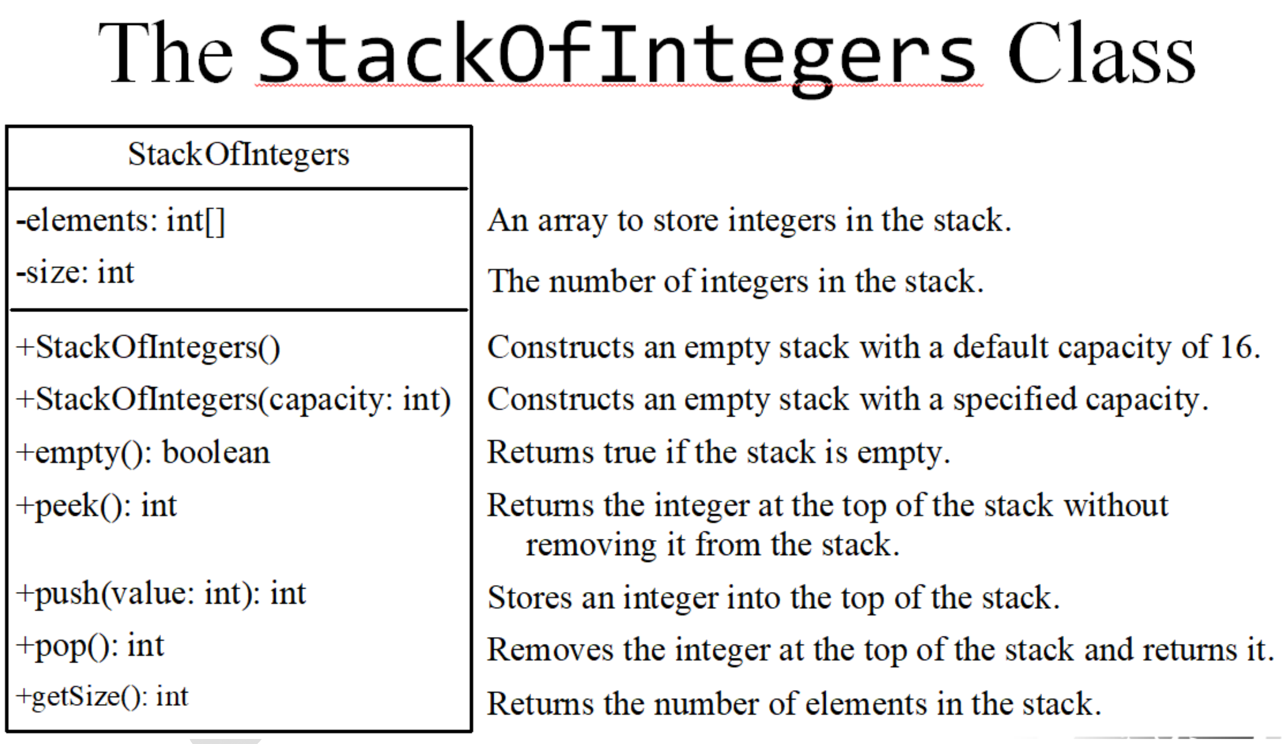
■ To explore the differences between the procedural paradigm and

object-oriented paradigm (§10.3).

■ To discover the relationships between classes (§10.4).

■ To design programs using the object-oriented paradigm

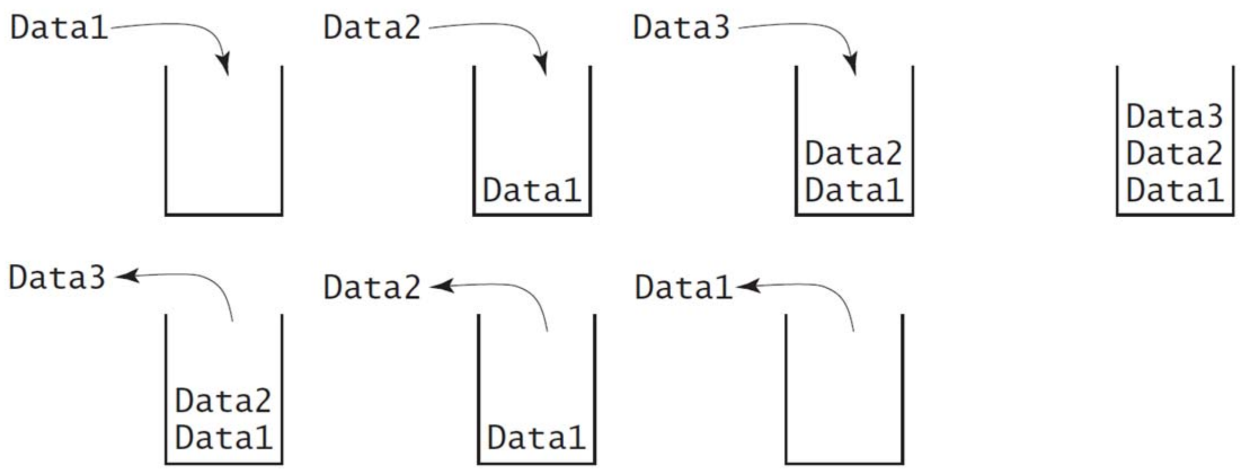
Example – StackOfIntegers class (Section 10.6):



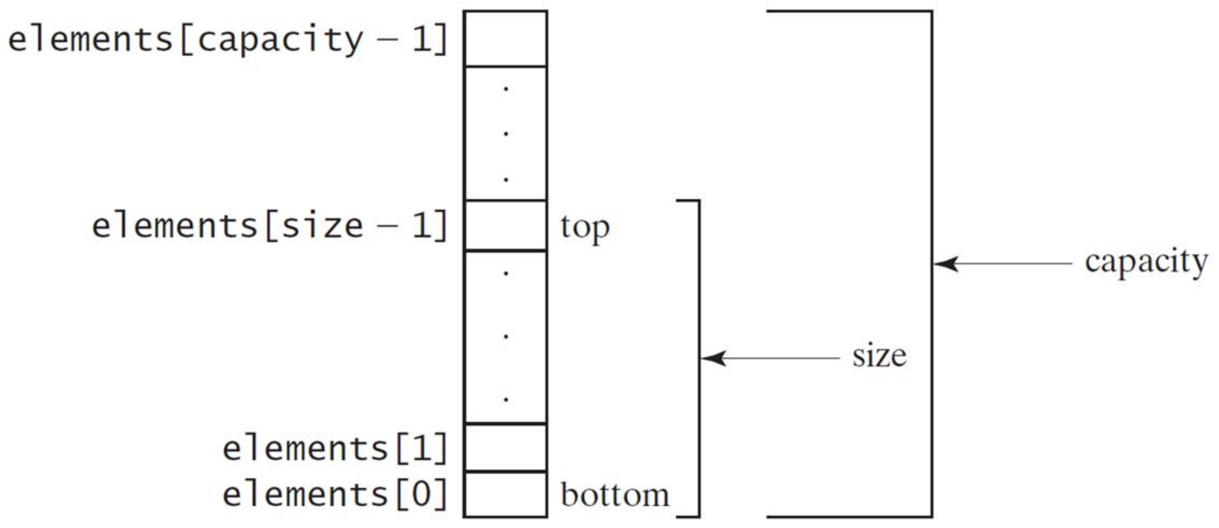
The StackOfIntegers Class

StackOfIntegers	
-elements: int[]	An array to store integers in the stack.
-size: int	The number of integers in the stack.
+StackOfIntegers()	Constructs an empty stack with a default capacity of 16.
+StackOfIntegers(capacity: int)	Constructs an empty stack with a specified capacity.
+empty(): boolean	Returns true if the stack is empty.
+peek(): int	Returns the integer at the top of the stack without removing it from the stack.
+push(value: int): int	Stores an integer into the top of the stack.
+pop(): int	Removes the integer at the top of the stack and returns it.
+getSize(): int	Returns the number of elements in the stack.

Example: Stack of Integers



Stack of Integers



Here is the code of StackOfIntegers class:

```
public class StackOfIntegers {
  private int[] elements;
  private int size;
  public static final int DEFAULT_CAPACITY = 16;

  /** Construct a stack with the default capacity 16
*/
  public StackOfIntegers() {
```

```java
    this(DEFAULT_CAPACITY);
  }

  /** Construct a stack with the specified maximum capacity */
  public StackOfIntegers(int capacity) {
    elements = new int[capacity];
  }

  /** Push a new integer into the top of the stack */
  public void push(int value) {
    if (size >= elements.length) {
      int[] temp = new int[elements.length * 2];
      System.arraycopy(elements, 0, temp, 0, elements.length);
      elements = temp;
    }

    elements[size++] = value;
  }

  /** Return and remove the top element from the stack */
  public int pop() {
    return elements[--size];
  }

  /** Return the top element from the stack */
  public int peek() {
    return elements[size - 1];
  }

  /** Test whether the stack is empty */
  public boolean empty() {
    return size == 0;
  }

  /** Return the number of elements in the stack */
  public int getSize() {
    return size;
  }
}
```

Here is the code of TestStackOfIntegers class (main method):

```java
public class TestStackOfIntegers {
  public static void main(String[] args) {
    StackOfIntegers stack = new StackOfIntegers();

    for (int i = 0; i < 10; i++)
```

STUDENTS

```
            stack.push(i);

        while (!stack.empty())
            System.out.print(stack.pop() + " ");
    }
}
```

Here is the output of StackOfIntegers program:

```
9 8 7 6 5 4 3 2 1 0
```

Exercise – StackOfStrings Class:

Design a class named StackOfStrings that contains:

a. A private array elements to store strings in the stack

b. A private data field size to store the number of strings in the stack

c. A constructor to construct an empty stack with a default capacity of 4

d. A constructor to construct an empty stack with a specified capacity

e. A method empty() that returns true if the stack is empty

f. A method push(String value) that stores value at the top of the stack. This method checks if the stack is full before pushing to it. In case the stack is full, it

i. copies the contents of the stack to a new temp array

ii. creates a new elements array with double the current capacity

iii. copies back temp to elements

iv. places value at the top of the new stack

g. A method pop() that removes the string at the top of the stack and returns it. In case the stack is empty, it returns "Stack is EMPTY"

h. A method peek() that returns the string at the top of the stack without removing it from the stack. In case the stack is empty, it returns "Stack is EMPTY"

i. A method printStack() as shown below:

```
public void printStack(){
  System.out.print("Stack (top to bottom) : ");
  for (int i=size-1; i>-1; i--)
    System.out.print(elements[i] + " ");
  System.out.println();
}
```

STUDENTS

1. Draw the UML diagram for the StackOfStrings class.

2. Implement the StackOfStrings class.

3. Use the test program shown on the following page which creates a StackOfStrings object, and then allows the user to push, pop or peek at the stack.

4. Draw the UML diagram for the StackOfStrings object created in the test program.

```
import java.util.Scanner;

public class Test {

   public static void main(String[] args) {

      StackOfStrings stack = new StackOfStrings();


      System.out.println("STACK OF STRINGS\nType PSH followed by a space "

        + "and a string and a carriage return or\nType POP followed "

        + "by a carriage return or\nType PEK followed by a carriage "

        + "return or \nType STP followed buy a carriage return");


      Scanner input = new Scanner(System.in);

      while (true) {

         System.out.print("Enter command: ");

         String s = input.nextLine();

         String stackCommand = s.substring(0, 3);


         switch (stackCommand) {

            case "PSH":

               stack.push(s.substring(s.indexOf(" ") + 1));

               stack.printStack();

               break;

            case "POP":

               System.out.println(stack.pop());

               stack.printStack();
```

STUDENTS

```
                break;
              case "PEK":
                System.out.println(stack.peek());
                stack.printStack();
                break;
              case "STP":
                System.out.println(stack.peek());
                System.exit(0)
              default:
                System.out.println("Illegal input. Try again.");
            }
          }
        }
      }
```

```
class StackOfStrings {
  // Implement the class here
}
```

Homework* – Prime factors and prime numbers:

Write a program that prompts the user to enter a positive integer which is greater than one and displays all its smallest factors in decreasing order. For example, if the integer is 140, the smallest factors are displayed as 7, 5, 2, 2. Use the StackOfIntegers class to store the factors (e.g., 2, 2, 5, 7) and retrieve and display them in reverse order. After that, write a program that displays all the prime numbers less than the number – which is used in the prime factors (e.g: 140)- in decreasing order. Use the StackOfIntegers class to store the prime numbers (e.g., 2, 3, 5, ...) and retrieve and display them in reverse order.

Here is the sample runs:

```
To use the prime factors you must enter a positive integer which is greater
than one: 140
7 5 2 2

Printing prime numbers under 140:
139 137 131 127 113 109 107 103 101 97 89 83 79 73 71 67 61 59 53 47 43 41
37 31 29 23 19 17 13 11 7 5 3 2
```

** This homework has a part of Challenge Question.*

STUDENTS

Lab 005: Inheritance & Polymorphism

Objective:

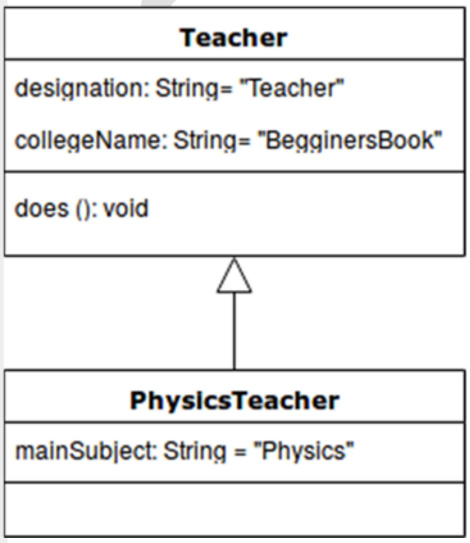
- To define a subclass from a superclass through inheritance.
- To use UML graphical notation to describe classes and objects.
- To override instance methods in the subclass.
- To explore the toString() method in the Object class.

Examples:

Example 1- Inheritance:

Write a test program to run the below classes where class Teacher is a base class, class PhysicsTeacher is a derived class which extends Teacher class as shown in UML digram.

```
class Teacher {
  String designation = "Teacher";
  String collegeName = "Beginnersbook";
  void does(){
          System.out.println("Teaching");
  }
}
class PhysicsTeacher extends Teacher{
  String mainSubject = "Physics";
}
```


Teacher
designation: String= "Teacher"
collegeName: String= "BegginersBook"
does (): void
PhysicsTeacher
mainSubject: String = "Physics"

Test.java

```
public class Test {
   public static void main(String args[]){
          PhysicsTeacher obj = new PhysicsTeacher();
          System.out.println(obj.collegeName);
          System.out.println(obj.designation);
          System.out.println(obj.mainSubject);
          obj.does();
  }
  }
```

Output:

```
Beginnersbook
```

Teacher

Physics

Teaching

Exercises:

Exercise 1- UML Digram:

Let us suppose we have five classes which are the Person, Student, Employee, Faculty, and Staff Classes. Moreover, let us make Student and Employee as subclasses of Person, and make Faculty and Staff as subclasses of Employee. Please create a UML digram for those classes based on the following instructions:

a) The Person class has the following private properties:
   - name: String and its accessor and mutator methods

b) The Student class has the following methods and private properties:
   - studentId: long
   - Constructor with two arguments: name, studentId
   - Override toString() in the Student class to return the name and studentId

c) The Employee class has the following private properties:
   - employeeId: int and its accessor and mutator methods

d) The Faculty class has the following methods and private properties:
   - rank: String
   - Constructor with three arguments: name, employeeId, rank
   - Override toString() in the Faculty class to return the name, employeeId, and rank

e) The Staff class has the following methods and private properties:
   - title: String
   - Constructor with three arguments: name, employeeId, title
   - Override toString() in the Staff class to return the name, employeeId, and title

STUDENTS

Homework:

Homework 1- Implement the previous five classes:

a) Create the previous five classes in Exercise 1 in Java.

   (Hins: use the UML digram)

b) Write a test program to creates objects from the Student, Faculty and Staff classes, and then invokes toString() methods for all three

Challenge Question:

Question 1- Remove duplicates:

- *Write a method that removes the duplicate elements from an array list of integers using the following header:*

public static void removeDuplicate(ArrayList<Integer> list)

- *Write a test program that prompts the user to enter 10 integers to a list and displays the distinct integers separated by exactly one space. Here is a sample run:*

```
Enter ten integers: 34 5 3 5 6 4 33 2 2 4

The distinct integers are 34 5 3 6 4 33 2
```

STUDENTS

Lab 006: Inheritance & Polymorphism (continuted)

Objective:

- To define a subclass from a superclass through inheritance.
- To discover polymorphism and dynamic binding
- To use UML graphical notation to describe classes and objects.
- To override instance methods in the subclass.
- To explore the toString() method in the Object class.

Examples:

Example 1- Runtime Polymorphism:

Lets say we have a class **`Animal`** that has a method **`sound().`** Since this is a generic class so we can't give it a implementation like: Roar, Meow, Oink etc. We had to give a generic message

```
public class Animal{
  public void sound(){
    System.out.println("Animal is making a sound");
  }
}
```

Now lets say we a subclass of Animal class: `Horse` that extends `Animal` class. We can provide the implementation to the same method like this:

```
class Horse extends Animal{
   @Override
   public void sound(){
      System.out.println("Neigh");
   }
   public static void main(String args[]){
      Animal obj = new Horse();
      obj.sound();
   }
}
```

Output:

```
Neigh
```

Exercises:

Exercise 1- Add a subclass:

- Create a subclass of Animal class: **Cat**.
- Create a a sound function with a massage “Meow’’ inside Cat Class that includes a sound function.
- Call Sound method in Cat class by the reference variable of Animal class.

Homework:

Homework 1- Account class:

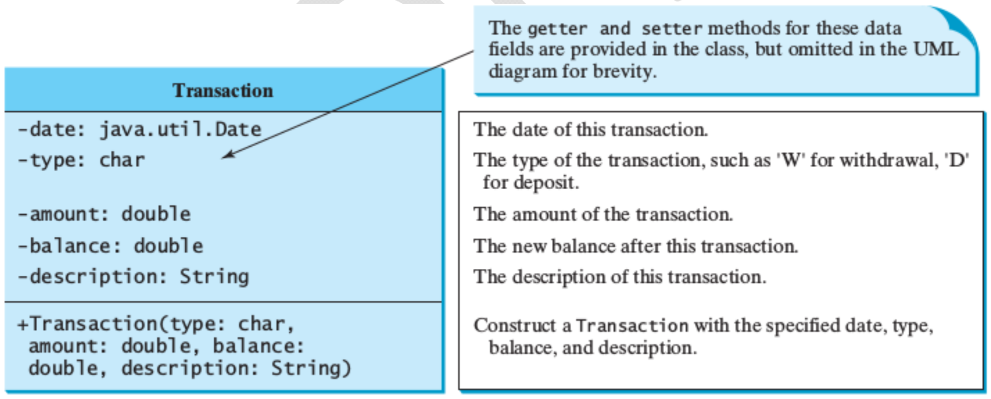
Update the below the Account class as follows:

a) Add a new data field name of the String type to store the name of the customer.

b) Add a new constructor that constructs an account with the specified name, id, and balance.

c) Add a new data field named transactions whose type is ArrayList that stores the transaction for the accounts.

d) Each transaction is an instance of the Transaction class. The Transaction class is defined as shown in the below figure:



The getter and setter methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

Transaction

```
-date: java.util.Date
-type: char

-amount: double
-balance: double
-description: String

+Transaction(type: char,
 amount: double, balance:
 double, description: String)
```

The date of this transaction.

The type of the transaction, such as 'W' for withdrawal, 'D' for deposit.

The amount of the transaction.

The new balance after this transaction.

The description of this transaction.

Construct a Transaction with the specified date, type, balance, and description.

e) Modify the withdraw and deposit methods to add a transaction to the transactions array list.

f) All other properties and methods are the same as in the Account class below.

g) Write a test program that creates an Account with annual interest rate 1.5%, balance 1000 , id 1122 , and name George . Deposit $30, $40, and $50 to the account and withdraw $5, $4, and $2 from the account.

h) Print an account summary that shows account holder name, interest rate, balance, and all transactions.

Here is Account class :

```
class Account {
   private int id = 0;
   private double balance = 0.0;
   private static double annualInterestRate = 0.0;
   private java.util.Date dateCreated;
   public Account() {
      dateCreated = new java.util.Date();
      }
   public Account(int id, double balace) {
      this();
      this.id = id;
      this.balance = balance;
     }
   public int getId() {
      return this.id;
   }
   public double getBalance() {
      return this.balance;
   }
   public double getAnnualInterestRate() {
      return annualInterestRate;
   }
   public String getDateCreated() {
      return this.dateCreated.toString();
   }
   public void setId(int id) {
      this.id = id;
   }
   public void setBalance(double balance) {
      this.balance = balance;
```

```
    }
    public void setAnnualInterestRate(double annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }
    public double getMonthlyInterestRate() {
        return (annualInterestRate / 100) / 12 ;
    }
    public double getMonthlyInterest() {
        return balance * getMonthlyInterestRate();
    }
    public void withdraw(double amount) {
        this.balance -= amount;
    }
    public void deposit(double amount) {
        this.balance += amount;
    }
}
```

The Test Class for the above Account Class is:

```
public class Test {
    public static void main(String[] args) {
        Account account = new Account(1122, 20000);
        account.setAnnualInterestRate(4.5);
        account.withdraw(2500.0);
        account.deposit(3000.0);
        System.out.println("Balance: $" + account.getBalance());
        System.out.println("Monthly Interest: " + account.getMonthlyInterest());
        System.out.println("Date Created: " + account.getDateCreated());
    }
}
```

Challenge Question:

Question 1- Shows extra info:

The point (h) in HomeWork1 is a bonus question which requires to print an account summary that shows account holder name, interest rate, balance, and all transactions.

STUDENTS

Lab 007: Exception Handling and File I/O

Objective:

- To declare exceptions in a method header.
- To throw exceptions in a method.
- To write a try-catch block to handle exceptions.
- To discover file/directory properties, to delete and rename files/directories, and to create directories using the File class.
- To write data to a file using the PrintWriter class.
- To read data from a file using the Scanner class.
- To develop a program that replaces text in a file.
- To read data from the Web.

Examples:

Example 1- Exception Handling:

The following examples shows how Exception Handling works:

```
public class MethodCallStackDemo {
  public static void main(String[] args) {
    System.out.println("Enter main()");
    methodA();
    System.out.println("Exit main()");
  }
  public static void methodA() {
    System.out.println("Enter methodA()");
    try {
      System.out.println(1 / 0);
        // A divide-by-0 triggers an ArithmeticException - an unchecked exception
        // This method does not catch ArithmeticException
        // It runs the "finally" and popped off the call stack
    } finally {
      System.out.println("finally in methodA()");
    }
    System.out.println("Exit methodA()");
  }
}
```

Output

```
Enter main()
Enter methodA()
finally in methodA()
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at MethodCallStackDemo.methodA(MethodCallStackDemo.java:11)
    at MethodCallStackDemo.main(MethodCallStackDemo.java:4)
```

Exercises:

Exercise 1- Exception Handling (continued):

What will be the output of the previous example if it is correct?

Hint: fix the error to know the answer.

Homework:

Homework 1- MergeFiles:

1. Design a class named MergeFiles

    I. Three file names are passed as command-line arguments to MergeFiles as follows:
    `java MergeFiles file1 file2 mergedFile`
    II. MergeFiles reads names stored in files file1 and file2
    III. Merges the two list of names into a single list
    IV. Sorts that single list
    V. Ignores repetitions
    VI. Writes the sorted, free-of-repetitions list to a new file named mergedFile.txt
    VII. The names in all three files are stored as one name per line
    NOTE: The design of MergeFiles class is very similar to the ReplaceText class that was discussed during the lectures on Exception Handling and File I/O.

2. Create file1.txt and file2.txt in the MergeFiles NetBeans project directory. Put at least five names in each of these files

3. In NetBeans, go to *Run* > *Set Project Configuration* > *Customize…* > *Arguments*. In the field in front of *Arguments* type the following:
    `file1.txt file2.txt mergedFile.txt`

4. Now if you run MergeFiles from within NetBeans, it will run with those three command line arguments

5. Run the program, and examine the resultant mergedFile.txt to ensure that the program is working properly

6. Delete throws Exception from main() and try to run the program. You will receive an error. Is this error due an *unchecked* exception or *checked* exception?

7. There is a try block, but no catch or finally blocks in the code. The program, however, still works properly. Why is that?

STUDENTS

Program Logic

f) Create an ArrayList object, names
g) Open file1.txt for reading using File and Scanner
h) Read the contents of file1.txt into the names object
i) Open file2.txt for reading using File and Scanner
j) Read the contents of file2.txt and add them to names
k) Sort names by first converting it to an array and then applying the built-in Arrays.sort() method
l) Print the contents of the sorted array to mergedFile.txt using File and PrintWriter
m) Skip any duplicate names while printing
n) The MergedFiles class can be invoked as follows:
```
java MergeFiles file1.txt file2.txt mergedFile.txt
```

Challenge Question:

Question 1- Read and Save:

Create a Java program that reads from a file on the Web using the URL class and save the file content in a local file.

STUDENTS

Lab 008: Abstract Classes & Interfaces

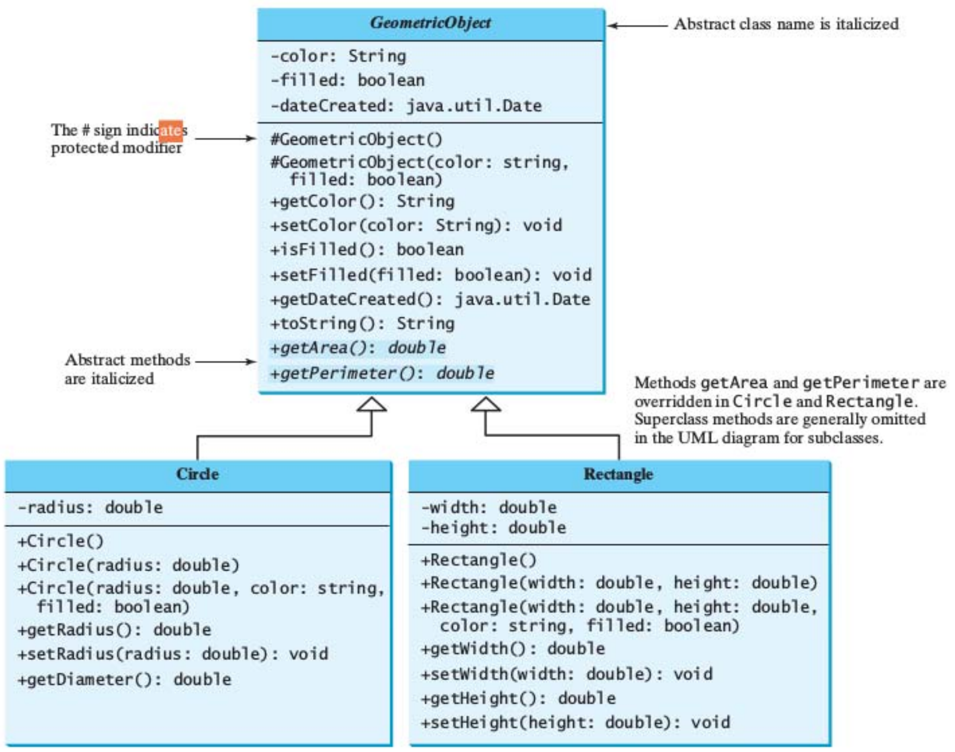
Objective:

- To design and use abstract classes.
- To design and use abstract methods.
- To design the Rational class for processing rational numbers.
- To design and use ArrayList.

Examples:

Example 1- Abstract Classes and Methods:

Here is an example to show how abstract Classes and Methods use:


GeometricObject
-color: String
-filled: boolean
-dateCreated: java.util.Date
#GeometricObject()
#GeometricObject(color: string, filled: boolean)
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String
+getArea(): double
+getPerimeter(): double
Abstract class name is italicized
The # sign indicates protected modifier
Abstract methods are italicized
Methods getArea and getPerimeter are overridden in Circle and Rectangle. Superclass methods are generally omitted in the UML diagram for subclasses.
Circle
-radius: double
+Circle()
+Circle(radius: double)
+Circle(radius: double, color: string, filled: boolean)
+getRadius(): double
+setRadius(radius: double): void
+getDiameter(): double
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+Rectangle(width: double, height: double, color: string, filled: boolean)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void

GeometricObject.java

```
public abstract class GeometricObject {
  private String color = "white";
  private boolean filled;
  private java.util.Date dateCreated;
  /** Construct a default geometric object */
  protected GeometricObject() {
    dateCreated = new java.util.Date();
  }
```

```
/** Construct a geometric object with color and filled value */
protected GeometricObject(String color, boolean filled) {
  dateCreated = new java.util.Date();
  this.color = color;
  this.filled = filled;
}
/** Return color */
public String getColor() {
  return color;
}
/** Set a new color */
public void setColor(String color) {
  this.color = color;
}
/** Return filled. Since filled is boolean,
 *  the get method is named isFilled */
public boolean isFilled() {
  return filled;
}
/** Set a new filled */
public void setFilled(boolean filled) {
  this.filled = filled;
}
/** Get dateCreated */
public java.util.Date getDateCreated() {
  return dateCreated;
}
@Override
public String toString() {
  return "created on " + dateCreated + "\ncolor: " + color +
    " and filled: " + filled;
 }
 /** Abstract method getArea */
 public abstract double getArea();
 /** Abstract method getPerimeter */
 public abstract double getPerimeter();
}
```

Rectangle.java

```
public class Rectangle extends GeometricObject {
 private double width;
 private double height;
 public Rectangle() {
 }
 public Rectangle(double width, double height) {
  this.width = width;
  this.height = height;
 }
 /** Return width */
 public double getWidth() {
  return width;
 }
 /** Set a new width */
 public void setWidth(double width) {
  this.width = width;
 }
 /** Return height */
 public double getHeight() {
  return height;
 }
 /** Set a new height */
 public void setHeight(double height) {
  this.height = height;
 }
```

```
  @Override /** Return area */
  public double getArea() {
    return width * height;
  }
  @Override /** Return perimeter */
  public double getPerimeter() {
    return 2 * (width + height);
  }
}
```

Circle.java

```
public class Circle extends GeometricObject {
  private double radius;
  public Circle() {
  }
  public Circle(double radius) {
    this.radius = radius;
  }
  /** Return radius */
  public double getRadius() {
    return radius;
  }
  /** Set a new radius */
  public void setRadius(double radius) {
    this.radius = radius;
  }
  @Override /** Return area */
  public double getArea() {
    return radius * radius * Math.PI;
  }
  /** Return diameter */
  public double getDiameter() {
    return 2 * radius;
```

```
 }
 @Override /** Return perimeter */
 public double getPerimeter() {
   return 2 * radius * Math.PI;
 }
 /* Print the circle info */
 public void printCircle() {
   System.out.println("The circle is created " + getDateCreated() +
     " and the radius is " + radius);
 }
}
```

Exercises:

Exercise 1- Retrieve the areas and perimeters of objects:

Write a test program to get areas and perimeters of both circle and rectangle objects in the previous example.

Homework:

Homework 1- Square class:

Design a new Square class that extends the abstract GeometricObject class (GeometricObject class is provided in Slides' Chapter 13).

- Draw the UML diagram for the classes Square and GeometricObject and then implement the Square class.
- Write a test program that prompts the user to one side of the square, a color, and a Boolean value to indicate whether the square is filled.

Note: The program should create a Square object with one side and set the color and filled properties using the input. Moreover, The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

Challenge Question:

Question 1- Shuffles an ArrayList:

Write the following method that shuffles an ArrayList of numbers:

public static void shuffle(ArrayList<Number> list)

STUDENTS

Lab 009: Abstract Classes & Interfaces (continued)

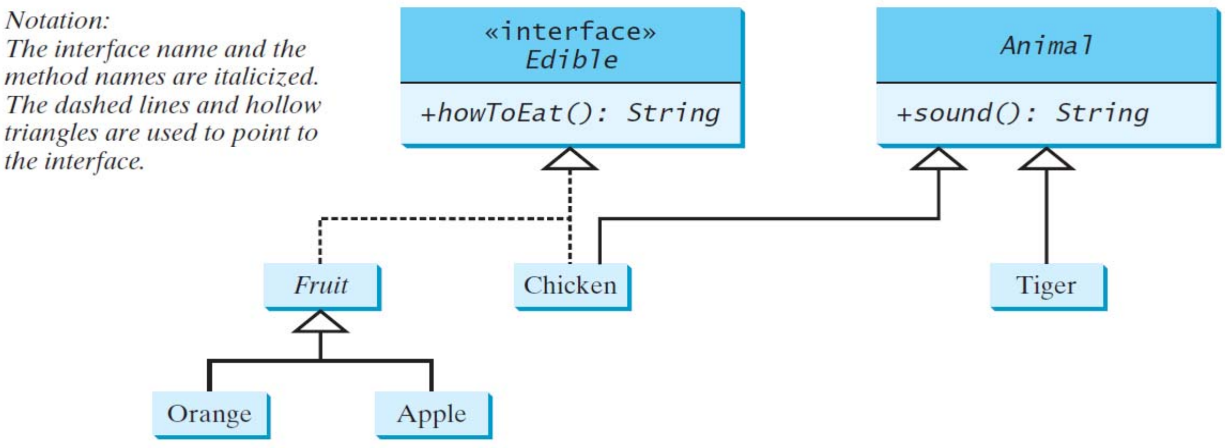
Objective:

- To generalize numeric wrapper classes, BigInteger, and BigDecimal using the abstract Number class.
- To specify common behavior for objects using interfaces.
- To define interfaces and define classes that implement interface.
- To define a natural order using the Comparable interface.
- To make objects cloneable using the Cloneable interface.
- To design the Rational class for processing rational numbers.

Examples:

Example 1- Edible interface:

We can use the Edible interface to specify whether an object is edible. This is accomplished by letting the class for the object implement this interface (or inherit the interface) using the implements keyword. For example, Chicken & Fruit implement the Edible interface in TestEdible while Tiger is not.


Notation:
The interface name and the method names are italicized. The dashed lines and hollow triangles are used to point to the interface.
«interface»
Edible
+howToEat(): String
Animal
+sound(): String
Fruit
Chicken
Tiger
Orange
Apple

Edible.java

```
public interface Edible {
  /** Describe how to eat */
  public abstract String howToEat();
}
```

TestEdible.java

```
public class TestEdible {
  public static void main(String[] args) {
    Object[] objects = {new Tiger(), new Chicken(), new Apple()};
    for (int i = 0; i < objects.length; i++) {
      if (objects[i] instanceof Edible)
        System.out.println(((Edible)objects[i]).howToEat());
      if (objects[i] instanceof Animal) {
        System.out.println(((Animal)objects[i]).sound());
      }
    }
  }
}
abstract class Animal {
  private double weight;
    public double getWeight() {
    return weight;
  }
   public void setWeight(double weight) {
    this.weight = weight;
  }
    /** Return animal sound */
  public abstract String sound();
}
class Chicken extends Animal implements Edible {
  @Override
  public String howToEat() {
    return "Chicken: Fry it";
  }
  @Override
  public String sound() {
    return "Chicken: cock-a-doodle-doo";
  }
}
class Tiger extends Animal {
  @Override
  public String sound() {
    return "Tiger: RROOAARR";
  }
}
abstract class Fruit implements Edible {
  // Data fields, constructors, and methods omitted here
}
class Apple extends Fruit {
  @Override
  public String howToEat() {
    return "Apple: Make apple cider";
  }
}
class Orange extends Fruit {
  @Override
  public String howToEat() {
    return "Orange: Make orange juice";
  }
}
```

Output

```
Tiger: RROOAARR
Chicken: Fry it
Chicken: cock-a-doodle-doo
Apple: Make apple cider
```

Exercises:

Exercise 1- Java.math.BigInteger.compareTo() Method:

Write an example to shows the usage of math.BigInteger.compareTo() method.

Homework:

Homework 1- The Colorable interface:

- Design an interface named Colorable with a void method named howToColor() . Every class of a colorable object must implement the Colorable interface.
- Design a class named Square that extends GeometricObject and implements Colorable. Implement howToColor to display the message Color all four sides .

  Hint: you may use **GeometricObject.java** in Example 1 in Lab 018 and update **Square.java** from Homework 1 in Lab 018.

Challenge Question:

Question 1- Enable the Course class cloneable:

Rewrite the Course class in Listing below to add a clone method to perform a deep copy on the students field.

Course.java

```
public class Course {
 private String courseName;
 private String[] students = new String[100];
 private int numberOfStudents;
 public Course(String courseName) {
   this.courseName = courseName;
 }
 public void addStudent(String student) {
  students[numberOfStudents] = student;
  numberOfStudents++;
}
public String[] getStudents() {
  return students;
}
public int getNumberOfStudents() {
 return numberOfStudents;
}
public String getCourseName() {
 return courseName;
}
}
```

STUDENTS

Lab 010: JAVAFX BASICS

Objective:

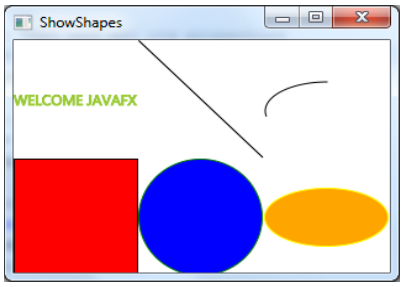
- To write a simple JavaFX program and understand the relationship among stages, scenes, and nodes.
- To create user interfaces using panes, UI controls, and shapes.

Examples:

Example 1- Shapes:

Write a program that shows different types of geometric shapes.

Here is a sample output:



```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Arc;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Ellipse;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.Line;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class ShowShapes extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane to hold the shapes
    GridPane pane = new GridPane();

    // Create a text and set its properties
    Text text = new Text("WELCOME JAVAFX ");
    text.setStroke(Color.RED);

    // Create a line and set its properties
    Line line = new Line(0, 0, 100, 100);
    text.setStroke(Color.YELLOWGREEN);

    // Create an arc and set its properties
    Arc arc = new Arc(50, 50, 50, 25, 90, 100);
    arc.setStroke(Color.BLACK);
    arc.setFill(Color.WHITE);
```

STUDENTS

```
    // Create a rectangle and set its properties
    Rectangle rectangle = new Rectangle(100, 100);
    rectangle.setStroke(Color.BLACK);
    rectangle.setFill(Color.RED);

    // Create a circle and set its properties
    Circle circle = new Circle(50);
    circle.setStroke(Color.GREEN);
    circle.setFill(Color.BLUE);

    // Create an ellipse and set its properties
    Ellipse ellipse = new Ellipse(50, 25);
    ellipse.setStroke(Color.YELLOW);
    ellipse.setFill(Color.ORANGE);

    // Add text to the pane
    pane.add(text, 0, 0);
    // Add line to the pane
    pane.add(line, 1, 0);
    // Add arc to the pane
    pane.add(arc, 2, 0);
    // Add circle to the pane
    pane.add(rectangle, 0, 1);
    // Add rectangle to the pane
    pane.add(circle, 1, 1);
    // Add ellipse to the pane
    pane.add(ellipse, 2, 1);
    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 305, 200);
    primaryStage.setTitle("ShowShapes"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage

  }

  /**
   * The main method is only needed for the IDE with limited
   * JavaFX support. Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}
```

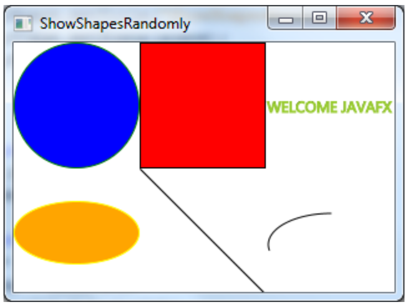
STUDENTS

Exercises:

Exercise 1- Random shapes:

Reprogram the application in the example 1, so it displays shapes randomly.

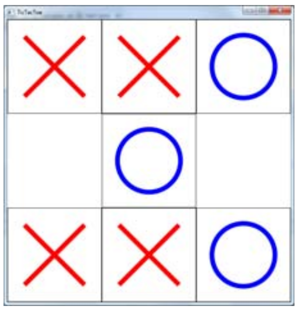
Here is a sample output:



Exercise 2- Tic-tac-toe board:

a) Write a program that displays a tic-tac-toe board. A cell may be X, O, or empty. The X and O are shapes.
b) (optional) What to display at each cell is randomly decided.

Here is a sample output:



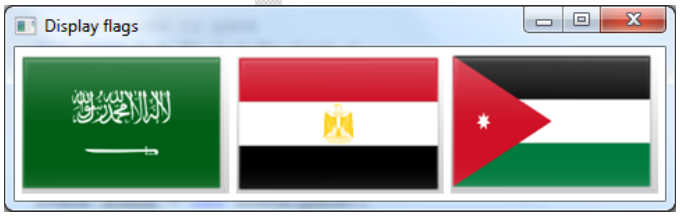
Homework:

Homework 1- Display flags:

Write a program that displays four Flags selected from a deck of 10, The flag image files are named 0.png, 1.png, …, 9.png and stored in the image directory.

- All four flags are distinct and selected randomly.
- Use different types of panes to display the flags regardless of the size of the screen:
  - In one row;
  - In one column;
  - In a grid.

Here is a sample output:



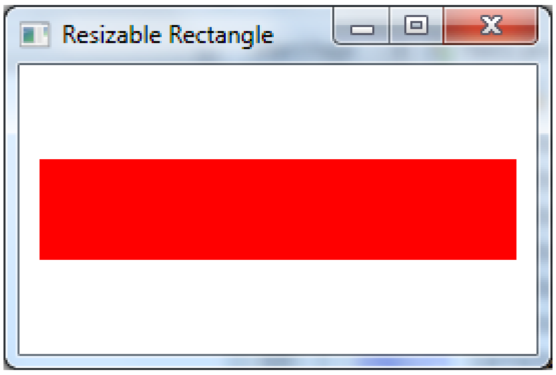
Challenge Question:

Question 1- Resizable rectangle:

Write a program that displays rectangle that changes its size as long as the size of the screen is changing

Here is a sample output:

STUDENTS



STUDENTS

Lab 011: Event-Driven Programming and Animations

Objective:

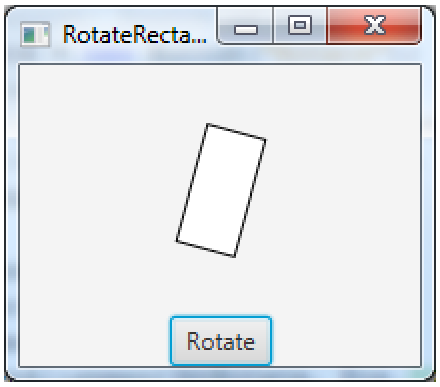
- To describe events, event sources, and event classes.
- To define handler classes, register handler objects with the source object, and write the code to handle events.

Examples:

Example 1- Rotate a rectangle:

Write a program that rotates a rectangle 15 degrees right when the Rotate button is clicked.

Here is a sample output:



STUDENTS

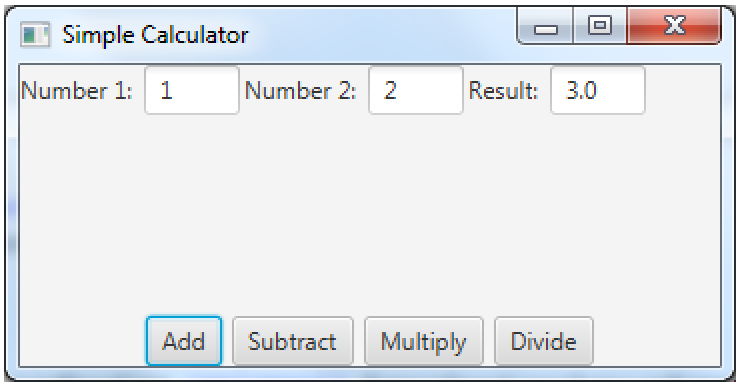
```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class RotateRectangle extends Application {

  private double angle = 0;

  @Override
  // Override the start method in the Application class
  public void start(Stage primaryStage) {
    StackPane stackPane = new StackPane();
    Rectangle rectangle = new Rectangle(30, 30, 30, 60);
    rectangle.setFill(Color.WHITE);
    rectangle.setStroke(Color.BLACK);
    stackPane.getChildren().add(rectangle);

    // Create a button- Source of event
    Button btRotate = new Button("Rotate");
    // Create Rotate handler
    RotateHandler rotateHandler = new RotateHandler(rectangle, angle);
    // Register the handler with the source
    btRotate.setOnAction(rotateHandler);

    BorderPane pane = new BorderPane();
    pane.setCenter(stackPane);
    pane.setBottom(btRotate);
    BorderPane.setAlignment(btRotate, Pos.TOP_CENTER);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 200, 150);
    primaryStage.setTitle("RotateRectangle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

  /**
   * The main method is only needed for the IDE with limited JavaFX support.
   * Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}

class RotateHandler implements EventHandler<ActionEvent> {

  Rectangle rectangle;
  double angle;

  public RotateHandler(Rectangle rectangle, double angle) {
    this.rectangle = rectangle;
    this.angle = angle;
  }

  @Override
  public void handle(ActionEvent event) {
    angle += 15;
    rectangle.setRotate(angle);
  }
}
```

STUDENTS

Example 2- Simple calculator:

Write a program to perform addition, subtraction, multiplication, and division.

Here is a sample output:



```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;

public class SimpleCalculator extends Application {
 @Override
 // Override the start method in the Application class
 public void start(Stage primaryStage) {
  FlowPane pane = new FlowPane();
  pane.setHgap(2);
  TextField tfNumber1 = new TextField();
  TextField tfNumber2 = new TextField();
  TextField tfResult = new TextField();
  tfNumber1.setPrefColumnCount(3);
  tfNumber2.setPrefColumnCount(3);
  tfResult.setPrefColumnCount(3);

  pane.getChildren().addAll(new Label("Number 1: "), tfNumber1,
   new Label("Number 2: "), tfNumber2, new Label("Result: "), tfResult);

  // Create four buttons
  HBox hBox = new HBox(5);
  Button btAdd = new Button("Add");
  Button btSubtract = new Button("Subtract");
  Button btMultiply = new Button("Multiply");
  Button btDivide = new Button("Divide");
  hBox.setAlignment(Pos.CENTER);
  hBox.getChildren().addAll(btAdd, btSubtract, btMultiply, btDivide);

  BorderPane borderPane = new BorderPane();
```
STUDENTS

```
    borderPane.setCenter(pane);
    borderPane.setBottom(hBox);
    BorderPane.setAlignment(hBox, Pos.TOP_CENTER);

    // Create a scene and place it in the stage
    Scene scene = new Scene(borderPane, 350, 150);
    primaryStage.setTitle("Simple Calculator"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage

    // Create general handler
    OperationHandler operationHandler = new OperationHandler(tfResult, tfNumber1,
tfNumber2);

    // Register the handler with all sources
    btAdd.setOnAction(operationHandler);

    btSubtract.setOnAction(operationHandler);

    btMultiply.setOnAction(operationHandler);

    btDivide.setOnAction(operationHandler);
  }

  /**
   * The main method is only needed for the IDE with limited
   * JavaFX support. Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}

class OperationHandler implements EventHandler<ActionEvent>{
    TextField tfResult;
    TextField tfNumber1;
    TextField tfNumber2;

    public OperationHandler(TextField tfResult, TextField tfNumber1, TextField tfNumber2) {
        this.tfResult = tfResult;
        this.tfNumber1 = tfNumber1;
        this.tfNumber2 = tfNumber2;
    }

    @Override
    public void handle(ActionEvent event) {
        Object o = event.getSource();
        if(o instanceof Button){
            Button b = (Button)o;
            String operation = b.getText();
            String result;
```

STUDENTS

```
         if(operation.equals("Add")){
            result = Double.parseDouble(tfNumber1.getText()) +
Double.parseDouble(tfNumber2.getText()) + "";
         }
         else if(operation.equals("Subtract")){
            result = Double.parseDouble(tfNumber1.getText()) -
Double.parseDouble(tfNumber2.getText()) + "";
         }
         else if(operation.equals("Multiply")){
            result = Double.parseDouble(tfNumber1.getText()) *
Double.parseDouble(tfNumber2.getText()) + "";
         }
         else {
            result = Double.parseDouble(tfNumber1.getText()) /
Double.parseDouble(tfNumber2.getText()) + "";
         }
         tfResult.setText(result);
      }
   }
}
```

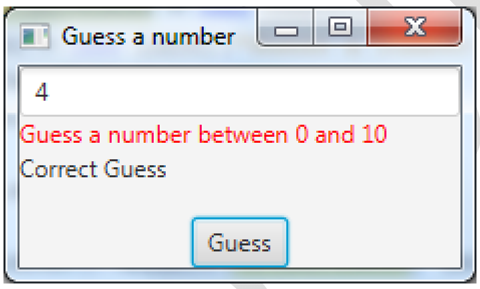
STUDENTS

Exercises:

Exercise 1- Guess a number:

Write a program that prompt a user to enter a guess of a number and checks wither the guess is correct or not.

Here is a sample output:



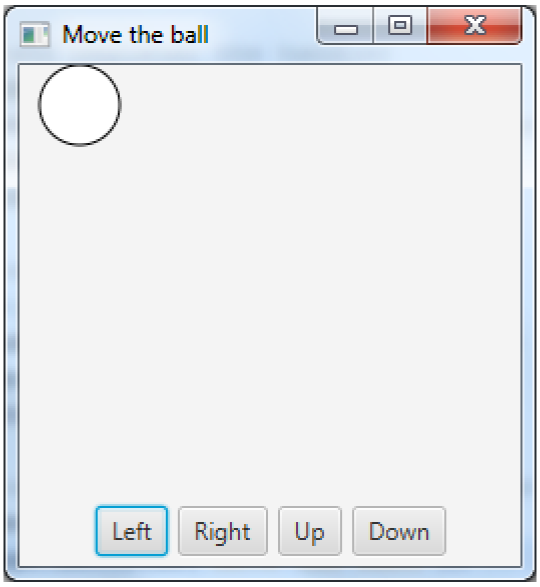
4
Guess a number between 0 and 10
Correct Guess

Homework:

Homework 1- Move the ball:

Write a program that moves the ball in a pane. You should define a pane class for displaying the ball and provide the methods for moving the ball left, right, up, and down.

Here is a sample output:



Challenge Question:

Question 1- Move the ball inside the boundary:

Reprogram the homework exercise, but this time check the boundary to prevent the ball from moving out of sight completely.

STUDENTS

Lab 012: Event-Driven Programming and Animations (continue)

Objective:

- To define handler classes using inner classes.
- To simplify event handling using lambda expressions.

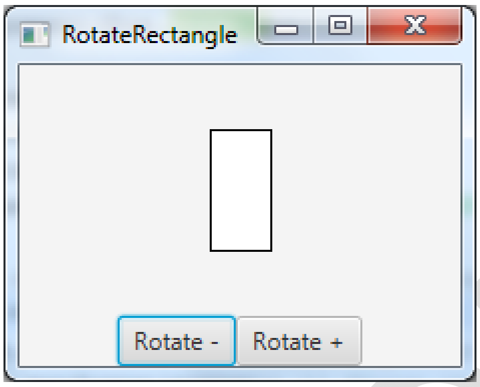
Examples:

Example 1- Rotate a rectangle left and right:

Write a program that rotates a rectangle 15 degrees right or left when the Rotate button is clicked.

Note: Use inner handler class.

Here is a sample output:



STUDENTS

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class RotateRectangleLeftRight extends Application {

   private double angle = 0;
   private Rectangle rectangle;

   @Override
   // Override the start method in the Application class
   public void start(Stage primaryStage) {
      StackPane stackPane = new StackPane();
      rectangle = new Rectangle(30, 30, 30, 60);
      rectangle.setFill(Color.WHITE);
```

```
    rectangle.setStroke(Color.BLACK);
    stackPane.getChildren().add(rectangle);

    // Create a button- Source of event
    Button btRightRotate = new Button("Rotate +");
    Button btLeftRotate = new Button("Rotate -");

    // Create Rotate handler
    RotateRightHandler rotateRightHandler = new RotateRightHandler();
    RotateLeftHandler rotateLeftHandler = new RotateLeftHandler();
    // Register the handler with the source
    btRightRotate.setOnAction(rotateRightHandler);
    btLeftRotate.setOnAction(rotateLeftHandler);

    HBox hbox = new HBox(btLeftRotate, btRightRotate);
    hbox.setAlignment(Pos.CENTER);

    BorderPane pane = new BorderPane();
    pane.setCenter(stackPane);
    pane.setBottom(hbox);
    BorderPane.setAlignment(hbox, Pos.TOP_CENTER);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 220, 150);
    primaryStage.setTitle("RotateRectangle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

  /**
   * The main method is only needed for the IDE with limited JavaFX support.
   * Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }

  private class RotateRightHandler implements EventHandler<ActionEvent> {

    @Override
    public void handle(ActionEvent event) {
      angle += 15;
      rectangle.setRotate(angle);
    }
  }

  private class RotateLeftHandler implements EventHandler<ActionEvent> {

    @Override
    public void handle(ActionEvent event) {
      angle -= 15;
```

STUDENTS

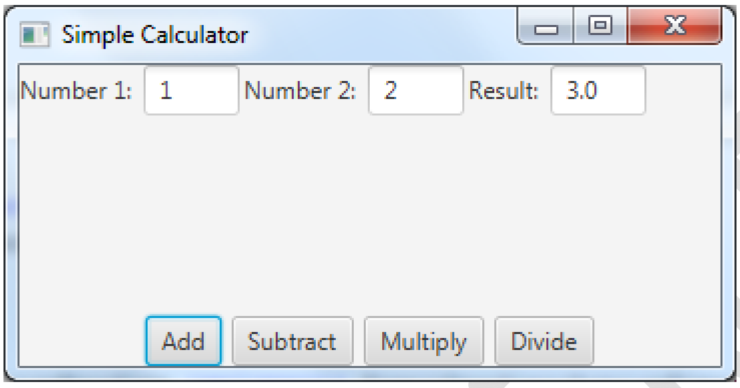
```
        rectangle.setRotate(angle);
      }
    }
}
```

Example 2- Simple calculator:

Write a program to perform addition, subtraction, multiplication, and division.

Note: Use lambda expression.

Here is a sample output:



```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;

public class Test1 extends Application {
  @Override
  // Override the start method in the Application class
  public void start(Stage primaryStage) {
    FlowPane pane = new FlowPane();
    pane.setHgap(2);
    TextField tfNumber1 = new TextField();
    TextField tfNumber2 = new TextField();
    TextField tfResult = new TextField();
    tfNumber1.setPrefColumnCount(3);
    tfNumber2.setPrefColumnCount(3);
    tfResult.setPrefColumnCount(3);

    pane.getChildren().addAll(new Label("Number 1: "), tfNumber1,
      new Label("Number 2: "), tfNumber2, new Label("Result: "), tfResult);
```

STUDENTS

```
    // Create four buttons
    HBox hBox = new HBox(5);
    Button btAdd = new Button("Add");
    Button btSubtract = new Button("Subtract");
    Button btMultiply = new Button("Multiply");
    Button btDivide = new Button("Divide");
    hBox.setAlignment(Pos.CENTER);
    hBox.getChildren().addAll(btAdd, btSubtract, btMultiply, btDivide);

    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(pane);
    borderPane.setBottom(hBox);
    BorderPane.setAlignment(hBox, Pos.TOP_CENTER);

    // Create a scene and place it in the stage
    Scene scene = new Scene(borderPane, 350, 150);
    primaryStage.setTitle("Simple Calculator"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage

    btAdd.setOnAction(e -> {
      tfResult.setText(Double.parseDouble(tfNumber1.getText()) +
        Double.parseDouble(tfNumber2.getText()) + "");
    });

    btSubtract.setOnAction(e -> {
      tfResult.setText(Double.parseDouble(tfNumber1.getText()) -
        Double.parseDouble(tfNumber2.getText()) + "");
    });

    btMultiply.setOnAction(e -> {
      tfResult.setText(Double.parseDouble(tfNumber1.getText()) *
       Double.parseDouble(tfNumber2.getText()) + "");
    });

    btDivide.setOnAction(e -> {
      tfResult.setText(Double.parseDouble(tfNumber1.getText()) /
        Double.parseDouble(tfNumber2.getText()) + "");
    });
  }

  /**
   * The main method is only needed for the IDE with limited
   * JavaFX support. Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}
```

STUDENTS

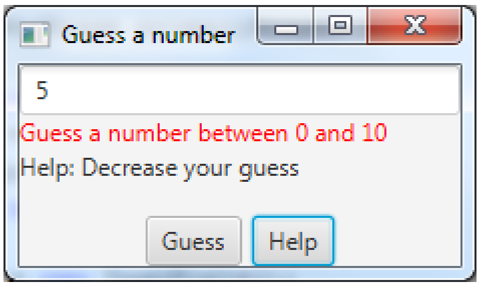
Exercises:

Exercise 1- Guess a number with help:

Write a program that prompt a user to enter a guess of a number and checks wither the guess is correct or not. Add help button that gives a user a hint to increase or decrease the guess.

Note: Use inner handler class.

Here is a sample output:


Guess a number

5

Guess a number between 0 and 10

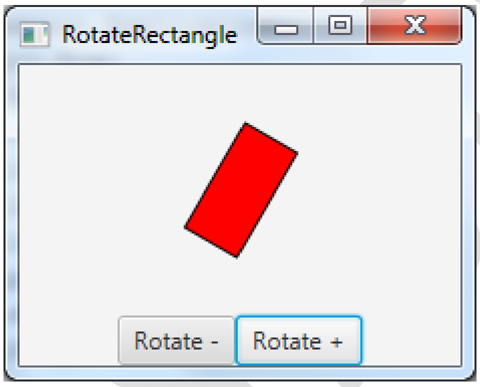
Help: Decrease your guess

Exercise 2- Rotate a rectangle with random color:

Reprogram example 1, so its color is changing randomly with each rotation. Use five different predefined colors for this purpose.

Note: Use lambda expression.

Here is a sample output:



STUDENTS

Lab 013: JAVAFX UI CONTROLS AND MULTIMEDIA

Objective:

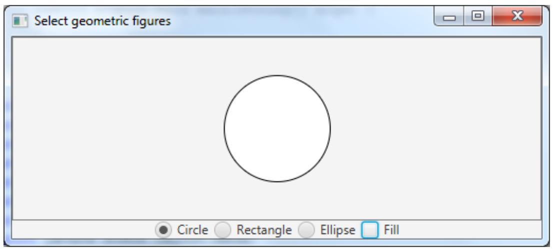
- To create graphical user interfaces with various user-interface controls

Examples:

Example 1- Select geometric figures:

Write a program that draws various figures. The user selects a figure from a radio button and uses a check box to specify whether it is filled.

Here is a sample output:



```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.CheckBox;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Ellipse;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class SelectGeometricFigures extends Application {
  private double paneWidth = 500;
  private double paneHeight = 150;

  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    StackPane pane = new StackPane();
    Circle circle = new Circle(20, 20, 50);
    circle.setStroke(Color.BLACK);
    circle.setFill(Color.WHITE);
    Rectangle rectangle = new Rectangle(20, 20, 80, 50);
    rectangle.setStroke(Color.BLACK);
    Ellipse ellipse = new Ellipse(20, 20, 50, 70);
    ellipse.setStroke(Color.BLACK);
    pane.setStyle("-fx-border-color: gray");
    pane.getChildren().add(circle);
```

STUDENTS

```
RadioButton rbCircle = new RadioButton("Circle");
RadioButton rbRectangle = new RadioButton("Rectangle");
RadioButton rbEllipse = new RadioButton("Ellipse");
CheckBox chkFill = new CheckBox("Fill");

ToggleGroup group = new ToggleGroup();
rbCircle.setToggleGroup(group);
rbCircle.setSelected(true);
rbRectangle.setToggleGroup(group);
rbEllipse.setToggleGroup(group);

HBox hBox = new HBox(5);
hBox.getChildren().addAll(rbCircle, rbRectangle, rbEllipse, chkFill);
hBox.setAlignment(Pos.CENTER);

BorderPane borderPane = new BorderPane();
borderPane.setCenter(pane);
borderPane.setBottom(hBox);

// Create a scene and place it in the stage
Scene scene = new Scene(borderPane, paneWidth, paneHeight + 40);
primaryStage.setTitle("Select geometric figures"); // Set the stage title
primaryStage.setScene(scene); // Place the scene in the stage
primaryStage.show(); // Display the stage

rbCircle.setOnAction(e -> {
  pane.getChildren().clear();
  pane.getChildren().add(circle);
  if (chkFill.isSelected()) {
    circle.setFill(Color.BLACK);
  }
  else {
    circle.setFill(Color.WHITE);
  }
});

rbRectangle.setOnAction(e -> {
  pane.getChildren().clear();
  pane.getChildren().add(rectangle);
  if (chkFill.isSelected()) {
    rectangle.setFill(Color.BLACK);
  }
  else {
    rectangle.setFill(Color.WHITE);
  }
});

rbEllipse.setOnAction(e -> {
  pane.getChildren().clear();
  pane.getChildren().add(ellipse);
```

STUDENTS

```
      if (chkFill.isSelected()) {
        ellipse.setFill(Color.BLACK);
      }
      else {
        ellipse.setFill(Color.WHITE);
      }
    });

    chkFill.setOnAction(e -> {
      if (chkFill.isSelected()) {
        circle.setFill(Color.BLACK);
        rectangle.setFill(Color.BLACK);
        ellipse.setFill(Color.BLACK);
      }
      else {
        circle.setFill(Color.WHITE);
        rectangle.setFill(Color.WHITE);
        ellipse.setFill(Color.WHITE);
      }
    });
  }

  /**
   * The main method is only needed for the IDE with limited
   * JavaFX support. Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}
```

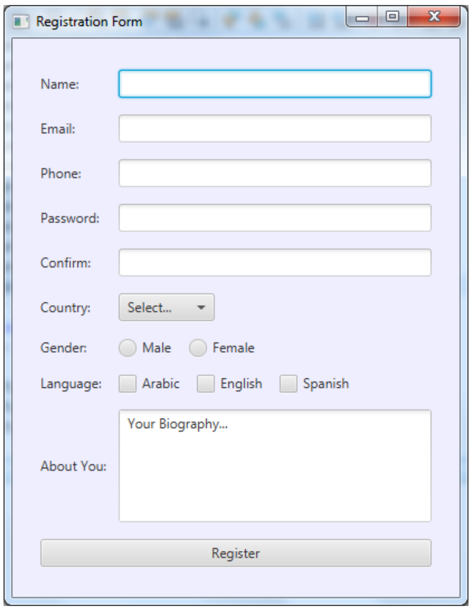
Exercises:

Exercise 1- Registration Form:

Write a program for account registration.

Here is a sample output:

STUDENTS

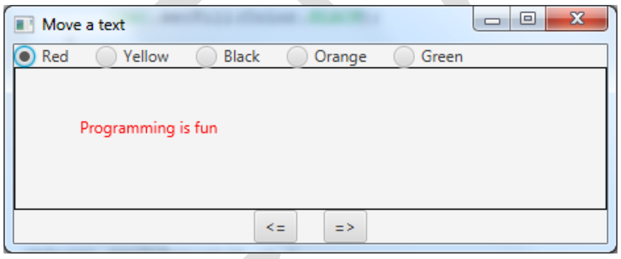


Homework:

Homework 1- Move a text:

Write a GUI program that uses buttons to move the message to the left and right and use the radio buttons to change the color for the message displayed.

Here is a sample output:



Programming is fun

Challenge Question:

Question 1- Active registration form:

Reprogram the exercise 1, so the form checks for validation when the button is clicked as follows:

Name- Letters only.

Email- Valid Email (it has @ and *dot* characters).

Phone- 8 digits.

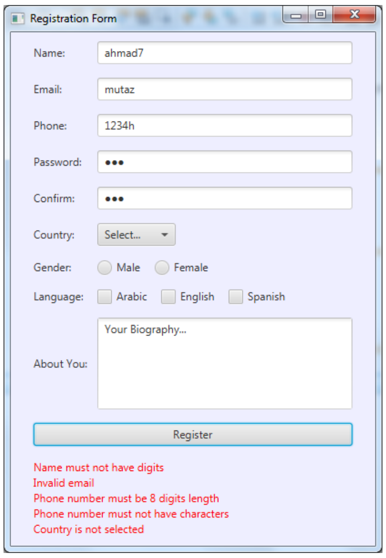
Password- Contains at least one capital letter, one small letter, one digit and minimum length is 8.

Confirm- Matches the entered password.

Country- must be selected.

About You- at least 50 characters.

Here is a sample output:



Registration Form

Name:	ahmad7
Email:	mutaz
Phone:	1234h
Password:	•••
Confirm:	•••
Country:	Select...
Gender:	Male Female
Language:	Arabic English Spanish
About You:	Your Biography...

Register

```
Name must not have digits
Invalid email
Phone number must be 8 digits length
Phone number must not have characters
Country is not selected
```

STUDENTS

Lab 014: JAVAFX Key and Mouse Events

Objective:

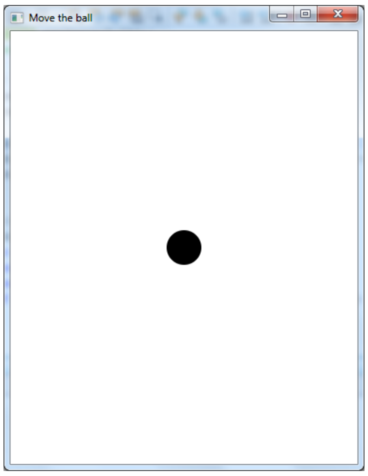
- To write programs to deal with MouseEvents
- To write programs to deal with KeyEvents

Examples:

Example 1- Move a circle using keys:

Write a program that moves a circle up, down, left, or right using the arrow keys.

Here is a sample output:



STUDENTS

```
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class MoveCircleKeys extends Application implements EventHandler<KeyEvent> {

  // Creat a ball
  Circle circle = new Circle(20);
  Pane pane = new Pane();

  @Override
  public void start(Stage primaryStage) throws Exception {
    circle.setCenterX(200);
    circle.setCenterY(250);
    pane.getChildren().add(circle);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 400, 500);
    scene.setOnKeyPressed(this);
```

```
        primaryStage.setTitle("Move a circle");
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage
    }

    // Method for moving a ball
    @Override
    public void handle(KeyEvent event) {
        double centerX = circle.getCenterX();
        double centerY = circle.getCenterY();
        if (null != event.getCode()) switch (event.getCode()) {
            case UP:
                circle.setCenterY(centerY - 5);
                break;
            case DOWN:
                circle.setCenterY(centerY + 5);
                break;
            case RIGHT:
                circle.setCenterX(centerX + 5);
                break;
            case LEFT:
                circle.setCenterX(centerX - 5);
                break;
            default:
                break;
        }
    }

    /**
     * The main method is only needed for the IDE with limited JavaFX support.
     * Not needed for running from the command line.
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```

STUDENTS

Example 2- Move a circle using mouse:

Reprogram the Example 1 to move the circle using the mouse (left click).

```
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;
```

```
public class MoveCircleMouse extends Application implements EventHandler<MouseEvent>
{

   // Creat a ball
   Circle circle = new Circle(20);
   Pane pane = new Pane();

   @Override
   public void start(Stage primaryStage) throws Exception {
      circle.setCenterX(200);
      circle.setCenterY(250);
      pane.getChildren().add(circle);

      // Create a scene and place it in the stage
      Scene scene = new Scene(pane, 400, 500);
      scene.setOnMousePressed(this);

      primaryStage.setTitle("Move a circle");
      primaryStage.setScene(scene); // Place the scene in the stage
      primaryStage.show(); // Display the stage
   }

   // Method for moving a ball
   @Override
   public void handle(MouseEvent event) {
      double sceneX = event.getX();
      double sceneY = event.getY();

      circle.setCenterY(sceneY);
      circle.setCenterX(sceneX);
   }

   /**
    * The main method is only needed for the IDE with limited JavaFX support.
    * Not needed for running from the command line.
    */
   public static void main(String[] args) {
      launch(args);
   }
}
```

STUDENTS

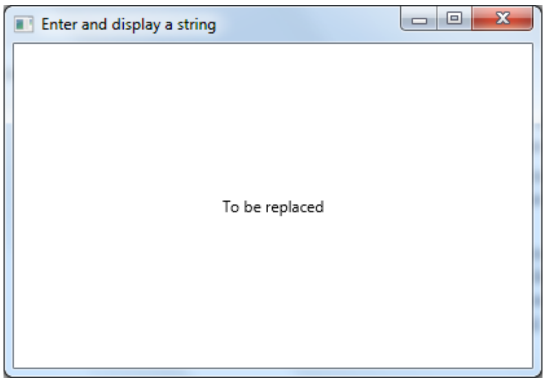
Exercises:

Exercise 1- Enter and display a string:

Write a program that receives a string from the keyboard and displays it on a pane. The *Enter* key signals the end of a string while the *Delete* clears the pane.

Whenever a new string is entered, it is displayed on the pane.

Here is a sample output:

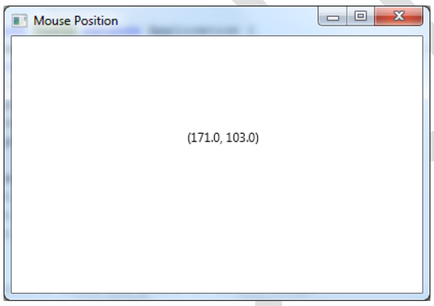


To be replaced

Exercise 2- Display the mouse position:

Write a program, that displays the mouse position when the mouse button is clicked

Here is a sample output:



(171.0, 103.0)

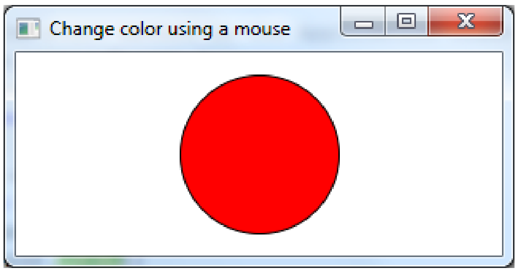
Homework:

Homework 1- Change color using a mouse:

Write a program that displays the color of a circle as red when the mouse button is pressed and as black when the mouse button is released.

Here is a sample output:

STUDENTS

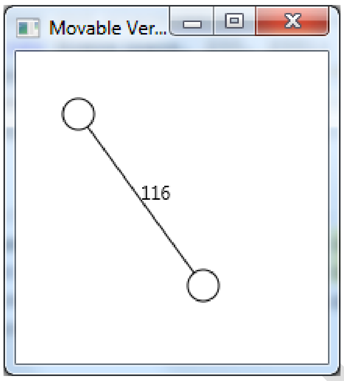


Challenge Question:

Question 1- Two movable vertices and their distances:

Write a program that displays two circles with radius 10 at location (40, 40) and (120, 150) with a line connecting the two circles. The distance between the circles is displayed along the line. The user can drag a circle. When that happens, the circle and its line are moved and the distance between the circles is updated.

Here is a sample output:



STUDENTS

Feedback:

STUDENTS