Lab 1 –Introduction to Java Programming

Answer the following questions.

Create a source file containing a Java program. Perform the following steps to compile the program and run it.

1. Create a file named `Welcome.java`. You can use any editor that will save your file in text format.

```
public class JADemo1 {

/**
 @param args the command line arguments
*/
public static void main(String[] args) {
     System.out.println("Welcome to Java.");
}
}
```

2. Compile the source file.
3. Run the bytecode.
4. Replace “Welcome to Java” with “My first program” in the program; save, compile, and run the program. You will see the message “My first program” displayed.
5. Replace `main` with `Main`, and recompile the source code. The compiler returns an error message because the Java program is case-sensitive.
6. Change it back, and compile the program again.
7. If you use command-line,
    a. Instead of the command `javac Welcome.java`, use `javac welcome.java`. What happens?
    b. Instead of the command `java Welcome`, use `java Welcome.class`. What happens?

Lab 2 –Primitive Data Types and Operations
Answer the following questions.

Use Scanner class for prompting the users for input.

Instructor-led Demo:

1. Write a program that reads a number in feet, converts it to meters, and displays the result. One foot is 0.305 meters.

Exercise:

1. Write a program that reads a Fahrenheit degree in double, then converts it to Celsius and displays the result on the console. The formula for the conversion is as follows:
   celsius = Fahrenheit – 32 * 5 / 9

2. Write a program that reads in the radius and length of a cylinder and computes volume using the following formulas:
   area = radius * radius * PI
   volume = area * length

3. Write a program that reads an integer between 0 and 1000 and adds all the digits in the integer. For example, if an integer is 943, the sum of all its digit is 16.

4. Write a program that converts an uppercase letter to a lowercase letter.

5. Write a program that receives an ASCII code (an integer between 0 and 128) and displays its character. For example, if the user enters 97, the program displays character ‘a’.

6. Write a program that reads an integer and checks whether it is even. For example, if your input is 25, the output should be :
   Is 25 an even number? false
   If your input is 2500, the output should be:
   Is 2500 an even number? true

7. Write a program that prompts the user to enter an integer and determines whether it is divisible by 5 or 6, whether it is divisible by 5 or 6, and whether it is divisible by 5 or 6, but not both. For example, if your input is 10, the output should be:
   Is 10 divisible by 5 and 6? false
   Is 10 divisible by 5 or 6? true
   Is 10 divisible by 5 or 6, but not both? true

8. Write a program that reads in investment amount, annual interest rate, and number of years, and displays the future investment value using the following formula.
   futureInvestmentVal = investmentAmount x $(1 + \text{monthlyInterestRate})^{\text{numberOfYears*12}}$

Lab 3 –Control Structure
Answer the following questions.

Instructor-led Demo:

1. Demonstrate the use of selection, looping, and enhance-for on a given scenario.

Exercise:

1. Write a program that sorts three integers. The integers are entered from the console and stored in variables, `num1`, `num2` and `num3`, respectively. The program sorts the numbers so that `num1 <= num2 <= num3`.

2. Write a program that reads three edges for a triangle and computes the perimeter if the input is valid. Otherwise, display that the input is invalid. The input is valid if the sum of any two edges is greater than third edge.

3. Write a program that prompts the user to enter the month and year, and displays the number of days in the month. For example, January is 31 days, February is 28 days, March is 31 and etc.

4. Write a program that prompts the user to enter assignment marks and displays the grade of the keyed in marks. The grading table is as follows:

Marks	**Grade**	**Description**
0-40	F	Fail
40-49	F+	Marginal Fail
50-54	D	Pass
55-64	C	
65-69	B	Credit
70-74	B+	
75-79	A	Distinction
80-100	A+	

5. Write a program that sum up all the values in double typed of an array. The array capacity is 100. You are required to use for-each construct (enhanced for).

6. Suppose that the tuition of a university is RM10000 this year and this tuition fee increases 5% every year. Write a program that uses a loop to compute the tuition in ten years.

7. Use do-while construct, write a program that prompts the users to continue the program execution. “Yes” to continue the program and “No” to terminate the program.

Lab 4 –Objects and Classes
Answer the following questions.

Instructor-led Demo:

1. Write a class named `Account` to model accounts. The UML diagram for the class is shown in Figure 4.0. Write a test program to test the `Account` class. In the client program, create an `Account` object with an account ID of 1222, a balance of 20000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2500, use the deposit method to deposit $3000, and print the balance and the monthly interest.

Account
-id:int -balance:double -annualInterestRate:double
+Account() +getId():int +getBalance():double +getAnnualInterestRate:double +setId(id:int):void +setBalance(bal:double):void +setAnnualInterestRate(rate:double):void +getMonthlyInterestRate():double +withdraw(amount:double):void +deposit(amount:double):void

Figure 4.0

Exercise:

1. Write a class named `Rectangle` to represent rectangles. The UML diagram for the class is shown in Figure 4.1 Suppose that all the rectangles are the same colour. Use a static variable for colour.

Rectangle
-width:double -height:double -color:String
+Rectangle() +Rectangle(width:double, height:double, color:String) +getWidth():double +setWidth(width:double):void +getHeight():double +setHeight(height:double):void +getColor:String +setColor(color:String):void +findArea():double +findPerimeter():double

Figure 4.1

Write a client program to test the class `Rectangle`. In the client program, create two `Rectangle` objects. Assign width 5 and height 50 each of the objects. Assign colour yellow. Display the properties of both objects and their areas.

2. Write a class named `Fan` to model fans. The properties, as shown in Figure 4.2, are speed, on, radius, and color. You need to provide the accessor and mutator methods for the properties, and the `toString` method for returning a string consisting of all the values of all the properties in this class. Suppose the fan has three fixed speeds. Use constants 1, 2, and 3 to denote slow, medium, and fast speed.

Fan
-speed:int -on:Boolean -radius:double -color:String

//constructor
//accessor methods
//toString method

Figure 4.2

   Write a client program to test the `Fan` class. In the client program, create a `Fan` object. Assign maximum speed, radius 10, color yellow, and turn it on. Display the object by invoking its `toString` object.

3. Java API has the `GregorianCalendar` class in the `java.util` package that can be used to obtain the year, month and day of a date. The no-arg constructor constructs an instance for the current date and the methods `get(GregorianCalendar.YEAR)`, `get(GregorianCalendar.MONTH)`, and `get(GregorianCalendar.DAY)` return the year, month and day. Write a program to test this class to display the current year, month and day.

4. Write a class called `Time`. The `Time` class contains data fields hour, minute and second with their respective get methods. The no-arg constructor sets the hour, minute, and second for the current time in GMT. The current time can be obtained using `System.currentTime()`. Write a client program to test the `Time` class. In the client program, create a `Time` object and display hour, minute and second using the get methods.

5. Using the `Time` class above, create an array storing `Time` object with its associated data (hour, minute, and second). `Time` object is created for every 5 seconds. Display the Time using `toString` method. The `toString` method returns hour:minute:second e.g., 1:30:30.

6. Consider the UML diagram below:

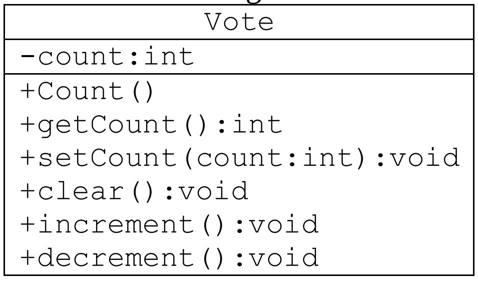


Figure 4.3

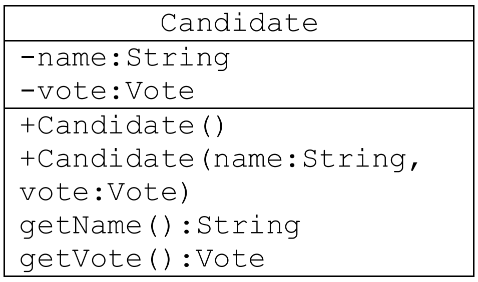


Figure 4.4

Develop a program that counts votes for two candidates for student body president. The input of vote is as follows:

Input	**Vote**
1	Increment Candidate1
2	Increment Candidate2
-1	Decrement Candidate1
-2	Decrement Candidate2
0	End the vote

Lab 5 –Inheritance and Polymorphism
Answer the following questions.

Instructor-led Demo:

1. Given any requirements, demonstrate inheritance, polymorphism, overriding and overloading program.

Exercise:

1. Implement a class named `Person` and two subclasses of `Person` named `Student` and `Employee`. Make `Faculty` and `Staff` subclasses of `Employee`. A person has a name, address, phone number, and email address. A student has a status (freshman, sophomore, junior, or senior). Define the status as a constant (Hint: Use Enum). An employee has an office, salary, and date-hired. Define a class named `MyDate` that contains the fields year, month, and day. A faculty member has office hours and a rank. A staff member has a title, override the `toString` method in each class to display the class name and the person's name.
    a. Furthermore from Q1, make `FullTime` and `PartTime` subclasses of `Staff`. Full time staff has a fixed salary whereas part time staff has a salary depending on worked hour. Implement this requirement that demonstrate the earning for both staff.
    b. Test your program. Demonstrate the result to the instructor.

2. The `Account` class is to model a bank account. An account has the properties account number, balance, and annual interest rate, and methods to deposit and withdrawal. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot go overdrawn. Test your program.

3. Enabling `GeometricObject` comparable, `Circle` and `Cylinder` are subclasses of `GeometricObject`. Modify the `GeometricObject` class to implement the `Comparable` interface, define the `max` method in the `GeometricObject` class. Write a test program that uses the `max` method to find the larger of two circles and the larger of two cylinders.
    a. Create a class named `ComparableCylinder` that extends `Cylinder` and implements `Comparable`. Implement the `compareTo` method to compare the cylinders on the basic of volume. Write a test class to find the larger of two instances of `ComparableCylinder` objects.
    b. Create an interface named `Colorable` having an abstract method named `howtoColor` method. Every class of a colorable object must implement the `Colorable` interface. Create a class named `Square` that extends `GeometricObject` and implements `Colorable`. Implement `howToColor` to display a message on how to color the square.

Lab 6 –Abstract Class and Interfaces

Answer the following questions.

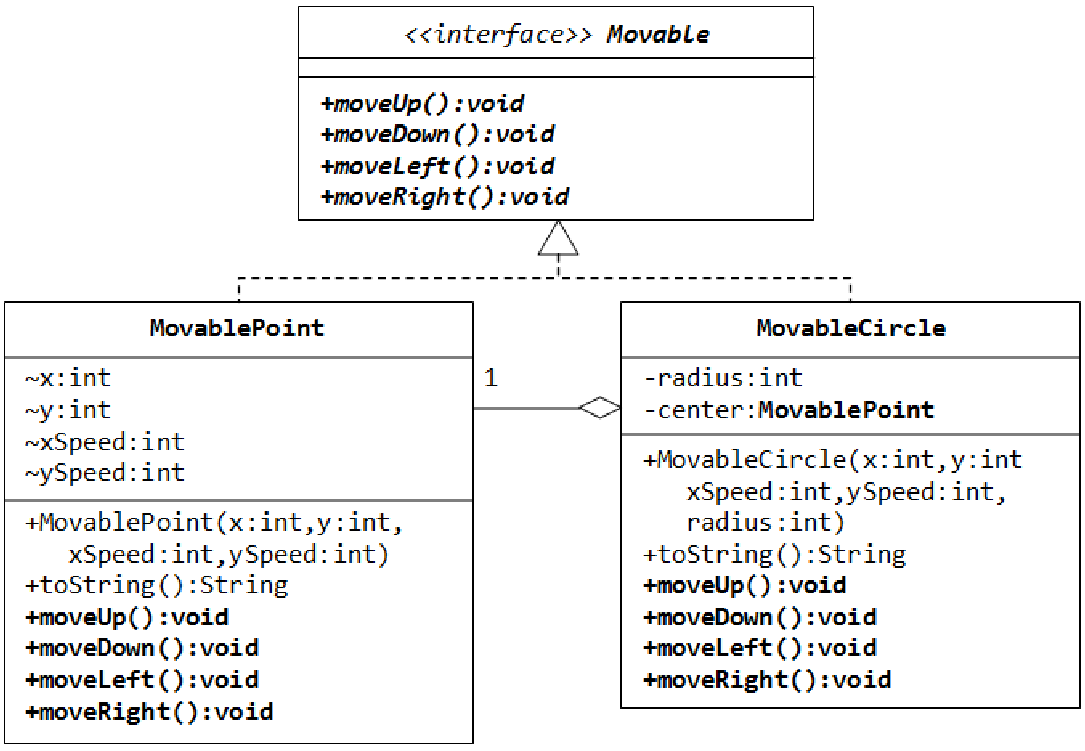
Instructor-led Demo:

1. Write a program that handles employees' salaries and workloads. There are part time and full time employee and they are payable with salaries. The salary for full time employee will be paid as monthly basis and part time employee will be based on hourly worked. There is a kind of employee (Assistant) which is not payable. All employees are required to work a certain number of hours depending on the type of employee. Part time work not more than 6 hour a week, full time work 40 hours a week and assistant work only 2 hours a week.

Exercise:

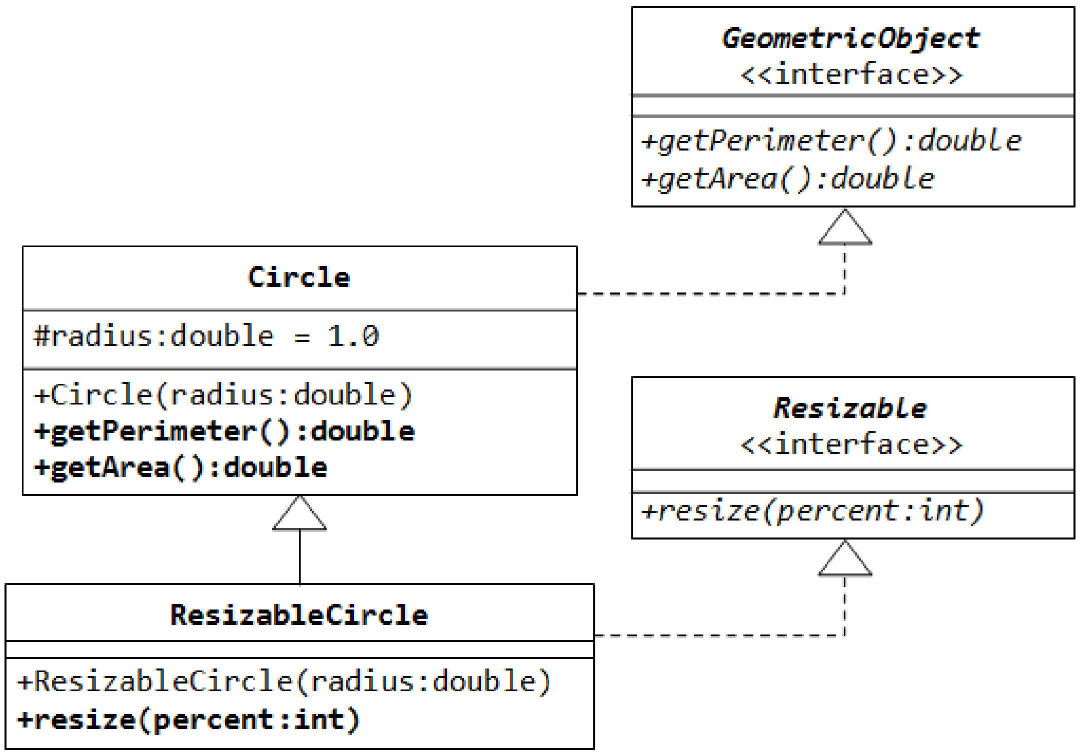
1. Write a method that returns the largest object in an array of objects. The method header is `public static Object max(Comparable[] a)`. All the objects are instances of the `Comparable` interface. The order of the objects in the array is determined using the `compareTo` method. Write a test program that creates an array of ten strings, an array of ten integers, and an array of ten dates, and find the largest string, integer, and date in the arrays.

2. Consider the UML diagram:



Write a program that demonstrate Movable interface and MovablePoint and MovableCircle classes. State any assumption for your program.

3. Modify the Lab 5-Q3, according to the UML diagram below:



Write a test program called TestResizableCircle to test the method defined in ResizeableCircle.

4. Write two comparator classes that demonstrate how to sort elements in an array using the `Comparator` interface. The example creates an array of Customer objects. The sort criteria are by age and DOB. Write a test program called TestCustomerSort to test the sorting by comparator.

More Exercise on OOP:

5. You are asked to write a discount system for a beauty saloon, which provides services and sells beauty products. It offers 3 types of memberships: Premium, Gold and Silver. Premium, gold and silver members receive a discount of 20%, 15%, and 10%, respectively, for all services provided. Customers without membership receive no discount. All members receives a flat 10% discount on products purchased (this might change in future). Your system shall consist of three classes: Customer, Discount and Visit, as shown in the class diagram. It shall compute the total bill if a customer purchases $x of products and $y of services, for a visit. Also write a test program to exercise all the classes.

Lab 7 –Handling Exceptions
Answer the following questions.

Instructor-led Demo:

1. Write a program that meets the following requirements:
   a. Create an array with one hundred randomly chosen integers.
   b. Cause an exception, *ArrayIndexOutOfBoundsException*, display the message "Out Of Bound". You can display all the array elements using looping.

Exercise:

1. "Passing Command-Line Arguments" is a simple command-line calculator. Note that the program terminates if any operand is non-numeric. Write a program with an exception handler that deals with non-numeric operands. Your program should display a message that informs the user of the wrong operand type before existing. For example,

Command arguments	**Output**
3 + 4	3 + 4 = 7
3/2 + 4	Wrong input: 3/2

2. Given the `Loan` class below:

```java
package loan;

import java.util.Date;

public class Loan {

    private double annualInterestRate;
    private int numberOfYears;
    private double loanAmount;
    private java.util.Date loanDate;

    public Loan() {
        // TODO Auto-generated constructor stub
    }

    public Loan(double annualInterestRate, int numberOfYears,
            double loanAmount) {
        super();
        this.annualInterestRate = annualInterestRate;
        this.numberOfYears = numberOfYears;
        this.loanAmount = loanAmount;
        this.loanDate = new java.util.Date();
    }

    public double getAnnualInterestRate() {
        return annualInterestRate;
    }

    public void setAnnualInterestRate(double annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    public int getNumberOfYears() {
        return numberOfYears;
    }

    public void setNumberOfYears(int numberOfYears) {
        this.numberOfYears = numberOfYears;
    }

    public double getLoanAmount() {
        return loanAmount;
    }

    public void setLoanAmount(double loanAmount) {
        this.loanAmount = loanAmount;
    }

    public java.util.Date getLoanDate() {
        return loanDate;
    }

    public double monthlyPayment(){
        return 0.0;//return actual monthly payment
    }

    public double totalPayment(){
        return 0.0;//return total payment
    }

}
```

Modify the `Loan` class to throw `IllegalArgumentException` if the loan amount, interest rate or number of years is less than or equal to zero.

3. Consider a Calculator program, note that number 1 and number 2 were a non-numeric string, the program would report exceptions. Modify the program with an exception handler to catch `ArithmeticException` (e.g., divided by 0) and `NumberFormatException` (e.g., input is not an integer), and display the errors in a message dialog box.

Lab 8 –GUI-based Development

Answer the following questions.

Note: You may use the netbeans IDE toolbox for designing the graphical user interface.

Exercise:

1. Write a program that adds a group of radio buttons to select background colours. The available colours are red, yellow, white, gray, and green.

2. Write a program that creates a simple calculator performs add, subtract, multiply and divide operations.

3. Write a program that converts miles and kilometres. If you enter a value in the Mile text field and press that Enter key, the corresponding kilometre is displayed in the Kilometer text field.

4. Write a program that calculates the future value of an investment at a given interest rate for a specified number of years. The formula for the calculation is as follows:

   $$\text{futureValue} = \text{investmentAmount} * (1 + \text{montlyInterestRate})^{\text{years}*12}$$

   Use text fields for interest rate, investment amount, and years. Display the future amount in a text field when the user clicks the Calculator button.

Lab 9 –File Input/Output
Answer the following questions.

Instructor-led Demo:

1. Given an array of integers, write a program that writes these integers into the file. Prompt the users to read the integers from the same file.

Exercise:

1. Write a program that counts the number of characters including words and lines in a file. The program prompts the user for inputting the filename. Sample output as follows:
   ```
   Please enter the filename: narrative.txt
   File Sample.txt has
   1732 characters,
   204 words and 70 lines.
   ```

2. Suppose that a text file **scores.txt** contains an unspecified number of scores. Write a program that reads the scores from the file and displays their total and average. Scores are separately by blanks.
   Hint: Read the scores one line at a time until all the lines are read. For each line, use `StringTokenizer` or `Scanner` to extract the scores and convert them into double values using the `Double.parseDouble` method.

3. Write a program that removes a specified string from a text file. Your program reads the file and generates a new file without the specified string, copies the new file to the original file. Prompt the user for a string to be removed and the filename. For example, remove "Java" string in **datafile.txt**.

4. Write a program to create a file named **ints.txt** if it does not exist. Write one hundred integers created randomly into the file using text I/O. Integers are separated by spaces in the file. Using `StringTokenizer` or `Scanner` to read the data back from the file and display the sorted data.

5. Write a program to create a file named **binaryint.dat** if it does not exist. If it exists, append new data to it. Write one hundred integers created randomly into the file using binary I/O.

6. Suppose a binary data file created in Q5 (**binaryint.dat**). Write a program to find the total of integers.

7. Write a program that stores an array of five `int` values 1, 2, 3, 4, and 5, a `Date` object for current time, and the double value 5.5 into the file named **objfile.dat.**

8. Given a *`Loan.java`* class. Rewrite the `Loan` class to implement `Serializable`. Write a program that creates five `Loan` objects and stores them in a file named **loanobj.dat.**

9. Given two files, write a program that concatenates these files and prints all content of these files. You are required to use SequenceInputStream class with its associated constructor as follows:

```
SequenceInputStream(InputStream s1, InputStream s2)
```

10. Give five files, modify the program in Q9 to read these files and print them on the console.

```
SequenceInputStream(Enumeration<? extends InputStream> e)
```

Lab 10 – Data Driven Development
Answer the following questions.

Instructor-led Demo for JavaDB:

1. Given any requirement, demonstrate insert, update, and select records from JavaDB in an application. This demonstration includes database, table, configuration, connection etc setup.

Exercise for MS Access:

1. Create a MS Access database namely, *DemoDB.accdb*.
    a. Make a Staff table which contains ID, FNAME, LNAME, AGE, DOB, DEPARTMENT, DATE_JOINED, etc.
    b. Make a Customer table which contains ID, FNAME, LNAME, DOB, USERNAME, PASSWORD, etc.
    c. Make a Product table which contains ID, CODE, NAME, DESC, COST, RETAIL_PRICE, QUANTITY, etc.

    Note: Recommended to put sample data for testing purpose.

2. Setup the data source for the accessing the *DemoDB.accdb*.
    a. Enter to the ODBC Data Source Administrator by either of these ways:
        i. Start menu → Control Panel →System and Security →Administrative Tools → Data Source (ODBC)
        OR
        ii. Type **Data Source** in the search textbox on the Start menu.
    b. In the ODBC Data Source Administrator, click **Add** button.
    c. Select the driver: **Microsoft Access Driver (*.mdb, *.accdb)**.
    d. In the ODBC Microsoft Access Setup dialog, enter a DATASOURCE_NAME. Click on **Select** to choose the targeted database, *DemoDB.accdb* in this case. Click **OK**.
    e. It will then add a new data source for users in the ODBC Data Source Administrator.

    Note: The data source name will be used in Database URL in the program.

3. Based on *DemoDB.accdb*, answer the following questions:
    a. Write a program that views, inserts and updates staff information stored in a database.
    b. Write a program that validates the customer login information. This program prompts the user for username and password input.

    Write a program that lists all the product information in an ascending order.

Lab 11 – Java Collection Framework

Answer the following questions.

Exercise:

1. Suppose that `set1` is a set that contains the strings “red”, “yellow”, and “green”, and that `set2` is another set that contains the strings “red”, “yellow”, and “blue”. Answer the following questions:
   a. What are `set1` and `set2` after executing `set1.addAll(set2)`?
   b. What are `set1` and `set2` after executing `set1.add(set2)`?
   c. What are `set1` and `set2` after executing `set1.removeAll(set2)`?
   d. What are `set1` and `set2` after executing `set1.remove(set2)`?
   e. What are `set1` and `set2` after executing `set1.retainAll(set2)`?

2. Suppose that `list1` is a list that contains the strings “red”, “yellow”, and “green”, and that `list2` is another list that contains the strings “red”, “yellow”, and “blue”. Answer the following questions.
   a. What are `list1` and `list2` after executing `list1.addAll(list2)`?
   b. What are `list1` and `list2` after executing `list1.add(list2)`?
   c. What are `list1` and `list2` after executing `list1.removeAll(list2)`?
   d. What are `list1` and `list2` after executing `list1.remove(list2)`?
   e. What are `list1` and `list2` after executing `list1.retainAll(list2)`?

3. Write a program that reads words from a text file and displays all the nonduplicate words in ascending order.

4. Write a program that reads words from a text file and displays all the words (duplicates allowed) in ascending alphabetical order.

5. Write a program that lets the user enter a set of numbers on the console. Use a linked list to store the numbers. Do not store duplicate numbers. Add operations that sort, shuffle, and reverse the list.

6. Write a program that demonstrates how to sort the elements in a tree set using the `Comparator` interface. The example creates a tree set of geometric objects. The geometric objects are sorted using the compare method in the `Comparator` interface based on their computed area.

7. Use the `Collections` class, find the minimum and maximum value in the list. Assume that the list is [2,12,98,77,55,34,7,23,5,33,77,89,12,34,5].