## CSCI 145 PA \_\_11\_\_ Submission

## Due Date:\_\_\_\_May 17 2023\_\_\_\_ Late (date and time):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Name(s):\_\_\_\_\_Ivan Leung\_\_\_\_\_ & \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Exercise 1 -- need to submit source code and I/O  
 -- check if completely done \_\_x\_\_ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Staff.java Author: Lewis/Loftus

//

//Represents the personnel staff of a particular business.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Staff {

StaffMember[] staffList;

// -----------------------------------------------------------------

// Sets up the list of staff members.

// -----------------------------------------------------------------

**public** Staff() {

staffList = **new** StaffMember[8];

staffList[0] = **new** Executive("Sam", "123 Main Line", "555-0469", "123-45-6789", 2423.07);

staffList[1] = **new** Employee("Carla", "456 Off Line", "555-0101", "987-65-4321", 1246.15);

staffList[2] = **new** Employee("Woody", "789 Off Rocker", "555-0000", "010-20-3040", 1169.23);

staffList[3] = **new** Hourly("Diane", "678 Fifth Ave.", "555-0690", "958-47-3625", 10.55);

staffList[4] = **new** Volunteer("Norm", "987 Suds Blvd.", "555-8374");

staffList[5] = **new** Volunteer("Cliff", "321 Duds Lane", "555-7282");

staffList[6] = **new** Commission("Ken", "167 Sunset Blvd.", "173-8787", "246-80-2468", 6.25, 0.20);

staffList[7] = **new** Commission("Johnny", "169 Sunrise St.", "173-6767", "135-79-1357", 9.75, 0.15);

((Executive) staffList[0]).awardBonus(500.00);

((Hourly) staffList[3]).addHours(40);

((Commission) staffList[6]).addHours(35);

((Commission) staffList[6]).addSales(400);

((Commission) staffList[7]).addHours(35);

((Commission) staffList[7]).addSales(950);

}

// -----------------------------------------------------------------

// Pays all staff members.

// -----------------------------------------------------------------

**public** **void** payday() {

**double** amount;

**for** (**int** count = 0; count < staffList.length; count++) {

System.***out***.println(staffList[count]);

amount = staffList[count].pay(); // polymorphic

**if** (amount == 0.0)

System.***out***.println("Thanks!");

**else**

System.***out***.println("Paid: " + amount);

System.***out***.println("------------------------------------");

}

}

}

**package** pa11;

/\* Java Class: CSCI 145

Author: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

**public** **class** Commission **extends** Hourly {

**private** **double** totalSales;

**private** **double** commissionRate;

**public** Commission(String eName, String eAddress, String ePhone, String socSecNumber, **double** rate, **double** commissionRate) {

**super**(eName, eAddress, ePhone, socSecNumber, rate);

**this**.commissionRate = commissionRate;

totalSales = 0;

}

**public** **void** addSales(**double** totalSales) {

**this**.totalSales += totalSales;

}

**public** **double** pay() {

**double** payment = **super**.pay() + (totalSales \* commissionRate);

totalSales = 0;

**return** payment;

}

**public** String toString() {

String result = **super**.toString() + "\nTotal Sales: " + totalSales;

**return** result;

}

}

Input/output below:

Name: Sam

Address: 123 Main Line

Phone: 555-0469

Social Security Number: 123-45-6789

Paid: 2923.07

------------------------------------

Name: Carla

Address: 456 Off Line

Phone: 555-0101

Social Security Number: 987-65-4321

Paid: 1246.15

------------------------------------

Name: Woody

Address: 789 Off Rocker

Phone: 555-0000

Social Security Number: 010-20-3040

Paid: 1169.23

------------------------------------

Name: Diane

Address: 678 Fifth Ave.

Phone: 555-0690

Social Security Number: 958-47-3625

Current hours: 40

Paid: 422.0

------------------------------------

Name: Norm

Address: 987 Suds Blvd.

Phone: 555-8374

Thanks!

------------------------------------

Name: Cliff

Address: 321 Duds Lane

Phone: 555-7282

Thanks!

------------------------------------

Name: Ken

Address: 167 Sunset Blvd.

Phone: 173-8787

Social Security Number: 246-80-2468

Current hours: 35

Total Sales: 400.0

Paid: 298.75

------------------------------------

Name: Johnny

Address: 169 Sunrise St.

Phone: 173-6767

Social Security Number: 135-79-1357

Current hours: 35

Total Sales: 950.0

Paid: 483.75

------------------------------------

Exercise 2 -- need to submit source code and I/O  
 -- check if completely done \_\_x\_\_ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//PaintThings.java

//

//Computes the amount of paint needed to paint various

//things. Uses the amount method of the paint class which

//takes any Shape as a parameter.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**import** java.text.DecimalFormat;

**public** **class** PaintThings {

// -----------------------------------------

// Creates some shapes and a Paint object

// and prints the amount of paint needed

// to paint each shape.

// -----------------------------------------

**public** **static** **void** main(String[] args) {

**final** **double** COVERAGE = 350;

Paint paint = **new** Paint(COVERAGE);

Rectangle deck;

Sphere bigBall;

Cylinder tank;

**double** deckAmt, ballAmt, tankAmt;

// Instantiate the three shapes to paint

deck = **new** Rectangle(20, 35);

bigBall = **new** Sphere(15);

tank = **new** Cylinder(10, 30);

// Compute the amount of paint needed for each shape

deckAmt = paint.amount(deck);

ballAmt = paint.amount(bigBall);

tankAmt = paint.amount(tank);

// Print the amount of paint for each.

DecimalFormat fmt = **new** DecimalFormat("0.0");

System.***out***.println("\nNumber of gallons of paint needed...");

System.***out***.println("Deck " + fmt.format(deckAmt));

System.***out***.println("Big Ball " + fmt.format(ballAmt));

System.***out***.println("Tank " + fmt.format(tankAmt));

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Paint.java

//

//Represents a type of paint that has a fixed area

//covered by a gallon. All measurements are in feet.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Paint {

**private** **double** coverage; // number of square feet per gallon

// -----------------------------------------

// Constructor: Sets up the paint object.

// -----------------------------------------

**public** Paint(**double** c) {

coverage = c;

}

// ---------------------------------------------------

// Returns the amount of paint (number of gallons)

// needed to paint the shape given as the parameter.

// ---------------------------------------------------

**public** **double** amount(Shape s) {

System.***out***.println("Computing amount for " + s);

**return** s.area() / coverage;

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

**public** **abstract** **class** Shape {

**private** String shapeName;

**public** Shape(String name) {

shapeName = name;

}

**public** **abstract** **double** area();

**public** String toString() {

**return** shapeName;

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Sphere.java

//

//Represents a sphere.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Sphere **extends** Shape {

**private** **double** radius; // radius in feet

// ----------------------------------

// Constructor: Sets up the sphere.

// ----------------------------------

**public** Sphere(**double** r) {

**super**("Sphere");

radius = r;

}

// -----------------------------------------

// Returns the surface area of the sphere.

// -----------------------------------------

**public** **double** area() {

**return** 4 \* Math.***PI*** \* radius \* radius;

}

// -----------------------------------

// Returns the sphere as a String.

// -----------------------------------

**public** String toString() {

**return** **super**.toString() + " of radius " + radius;

}

}

**package** pa11;

/\* Java Class: CSCI 145

Author: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

**public** **class** Rectangle **extends** Shape {

**private** **double** length;

**private** **double** width;

Rectangle(**double** l, **double** w) {

**super**("Rectangle");

length = l;

width = w;

}

**public** **double** area() {

**return** length \* width;

}

**public** String toString() {

**return** **super**.toString() + " of length " + length + " and width " + width;

}

}

**package** pa11;

/\* Java Class: CSCI 145

Author: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

**public** **class** Cylinder **extends** Shape {

**private** **double** radius;

**private** **double** height;

Cylinder(**double** r, **double** h) {

**super**("Cylinder");

radius = r;

height = h;

}

**public** **double** area() {

**return** 2 \* Math.***PI*** \* radius \* height + 2 \* Math.***PI*** \* radius \* radius;

}

**public** String toString() {

**return** **super**.toString() + " of radius " + radius + " and height " + height;

}

}

Input/output below:

Computing amount for Rectangle of length 20.0 and width 35.0

Computing amount for Sphere of radius 15.0

Computing amount for Cylinder of radius 10.0 and height 30.0

Number of gallons of paint needed...

Deck 2.0

Big Ball 8.1

Tank 7.2

Exercise 3 -- need to submit source code and I/O  
 -- check if completely done \_\_x\_\_ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IntegerListTest.java

//

//Provide a menu-driven tester for the IntegerList class.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**import** java.util.Scanner;

**public** **class** IntegerListTest {

**static** IntegerList *list* = **new** IntegerList(10);

**static** Scanner *scan* = **new** Scanner(System.***in***);

// -------------------------------------------------------

// Create a list, then repeatedly print the menu and do what the

// user asks until they quit

// -------------------------------------------------------

**public** **static** **void** main(String[] args) {

*printMenu*();

**int** choice = *scan*.nextInt();

**while** (choice != 0) {

*dispatch*(choice);

*printMenu*();

choice = *scan*.nextInt();

}

}

// -------------------------------------------------------

// Do what the menu item calls for

// -------------------------------------------------------

**public** **static** **void** dispatch(**int** choice) {

**int** loc;

**int** oldVal;

**int** newVal;

**switch** (choice) {

**case** 0:

System.***out***.println("Bye!");

**break**;

**case** 1:

System.***out***.println("How big should the list be?");

**int** size = *scan*.nextInt();

*list* = **new** IntegerList(size);

*list*.randomize();

**break**;

**case** 2:

*list*.selectionSort();

**break**;

**case** 3:

System.***out***.print("Enter the value to look for: ");

loc = *list*.search(*scan*.nextInt());

**if** (loc != -1)

System.***out***.println("Found at location " + loc);

**else**

System.***out***.println("Not in list");

**break**;

**case** 4:

*list*.print();

**break**;

**case** 5:

System.***out***.print("Enter the value to be replaced: ");

oldVal = *scan*.nextInt();

System.***out***.print("Enter a new value: ");

newVal = *scan*.nextInt();

*list*.replaceFirst(oldVal, newVal);

**break**;

**case** 6:

System.***out***.print("Enter the value to be replaced: ");

oldVal = *scan*.nextInt();

System.***out***.print("Enter a new value: ");

newVal = *scan*.nextInt();

*list*.replaceAll(oldVal, newVal);

**break**;

**default**:

System.***out***.println("Sorry, invalid choice");

}

}

// ------------------------------------------------------

// Print the user's choices

// ------------------------------------------------------

**public** **static** **void** printMenu() {

System.***out***.println("\n Menu ");

System.***out***.println(" ====");

System.***out***.println("0: Quit");

System.***out***.println("1: Create a new list (\*\* do this first!! \*\*)");

System.***out***.println("2: Sort the list using selection sort");

System.***out***.println("3: Find an element in the list using linear search");

System.***out***.println("4: Print the list");

System.***out***.println("5: Replace the first matching element found in the list");

System.***out***.println("6: Replace all the matching elements found in the list");

System.***out***.print("\nEnter your choice: ");

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IntegerList.java

//

//Define an IntegerList class with methods to create, fill,

//sort, and search in a list of integers.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** IntegerList {

**int**[] list; // values in the list

// -------------------------------------------------------

// create a list of the given size

// -------------------------------------------------------

**public** IntegerList(**int** size) {

list = **new** **int**[size];

}

// -------------------------------------------------------

// fill array with integers between 1 and 100, inclusive

// -------------------------------------------------------

**public** **void** randomize() {

**for** (**int** i = 0; i < list.length; i++)

list[i] = (**int**) (Math.*random*() \* 100) + 1;

}

// -------------------------------------------------------

// print array elements with indices

// -------------------------------------------------------

**public** **void** print() {

**for** (**int** i = 0; i < list.length; i++)

System.***out***.println(i + ":\t" + list[i]);

}

// -------------------------------------------------------

// return the index of the first occurrence of target in the list.

// return -1 if target does not appear in the list

// -------------------------------------------------------

**public** **int** search(**int** target) {

**int** location = -1;

**for** (**int** i = 0; i < list.length && location == -1; i++)

**if** (list[i] == target)

location = i;

**return** location;

}

// -------------------------------------------------------

// sort the list into ascending order using the selection sort algorithm

// -------------------------------------------------------

**public** **void** selectionSort() {

**int** minIndex;

**for** (**int** i = 0; i < list.length - 1; i++) {

// find smallest element in list starting at location i

minIndex = i;

**for** (**int** j = i + 1; j < list.length; j++)

**if** (list[j] < list[minIndex])

minIndex = j;

// swap list[i] with smallest element

**int** temp = list[i];

list[i] = list[minIndex];

list[minIndex] = temp;

}

}

**public** **void** replaceFirst(**int** oldVal, **int** newVal) {

**int** index = search(oldVal);

**if** (index != -1)

list[index] = newVal;

}

**public** **void** replaceAll(**int** oldVal, **int** newVal) {

**for** (**int** i = 0; i < list.length; ++i)

**if** (list[i] == oldVal)

list[i] = newVal;

}

**public** **void** sortDecreasing() {

;

}

**public** **int** binarySearchD(**int** target) {

**return** 0;

}

}

Input/output below:

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 1

How big should the list be?

25

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 4

0: 46

1: 31

2: 61

3: 15

4: 35

5: 22

6: 13

7: 54

8: 48

9: 1

10: 4

11: 79

12: 61

13: 64

14: 37

15: 82

16: 31

17: 31

18: 43

19: 20

20: 33

21: 54

22: 86

23: 14

24: 80

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 3

Enter the value to look for: 31

Found at location 1

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 3

Enter the value to look for: 5

Not in list

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 4

0: 1

1: 4

2: 13

3: 14

4: 15

5: 20

6: 22

7: 31

8: 31

9: 31

10: 33

11: 35

12: 37

13: 43

14: 46

15: 48

16: 54

17: 54

18: 61

19: 61

20: 64

21: 79

22: 80

23: 82

24: 86

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 5

Enter the value to be replaced: 54

Enter a new value: 31

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 4

0: 1

1: 4

2: 13

3: 14

4: 15

5: 20

6: 22

7: 31

8: 31

9: 31

10: 33

11: 35

12: 37

13: 43

14: 46

15: 48

16: 31

17: 54

18: 61

19: 61

20: 64

21: 79

22: 80

23: 82

24: 86

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 5

Enter the value to be replaced: 90

Enter a new value: 33

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 4

0: 1

1: 4

2: 13

3: 14

4: 15

5: 20

6: 22

7: 31

8: 31

9: 31

10: 33

11: 35

12: 37

13: 43

14: 46

15: 48

16: 31

17: 54

18: 61

19: 61

20: 64

21: 79

22: 80

23: 82

24: 86

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 6

Enter the value to be replaced: 31

Enter a new value: 40

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 4

0: 1

1: 4

2: 13

3: 14

4: 15

5: 20

6: 22

7: 40

8: 40

9: 40

10: 33

11: 35

12: 37

13: 43

14: 46

15: 48

16: 40

17: 54

18: 61

19: 61

20: 64

21: 79

22: 80

23: 82

24: 86

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 6

Enter the value to be replaced: 88

Enter a new value: 91

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 4

0: 1

1: 4

2: 13

3: 14

4: 15

5: 20

6: 22

7: 40

8: 40

9: 40

10: 33

11: 35

12: 37

13: 43

14: 46

15: 48

16: 40

17: 54

18: 61

19: 61

20: 64

21: 79

22: 80

23: 82

24: 86

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 0

*Add more exercises as needed*

Exercise 4 -- need to submit source code and I/O  
 -- check if completely done \_\_\_\_ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

**package** pa11;

/\* Java Class: CSCI 145

Author: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

**import** java.util.Scanner;

**import** java.util.ArrayList;

**public** **class** MyGarden {

**public** **static** **void** main(String[] args) {

**final** String SENTINEL\_VALUE = "neither";

**int** totalPlants = 0;

**int** totalCost = 0;

String type;

Plant newPlant;

ArrayList<Plant> myGarden = **new** ArrayList<>();

Scanner scan = **new** Scanner(System.***in***);

**do** {

System.***out***.print("Enter plant or flower: ");

type = scan.next();

**if** (type.equalsIgnoreCase("Plant")) {

newPlant = **new** Plant();

newPlant.setPlantName(scan.next());

newPlant.setPlantCost(scan.nextInt());

myGarden.add(newPlant);

++totalPlants;

}

**else** **if** (type.equalsIgnoreCase("Flower")) {

newPlant = **new** Flower();

newPlant.setPlantName(scan.next());

newPlant.setPlantCost(scan.nextInt());

((Flower) newPlant).setPlantType(scan.nextBoolean());

((Flower) newPlant).setColorOfFlowers(scan.next());

myGarden.add(newPlant);

++totalPlants;

}

} **while**(!type.equalsIgnoreCase(SENTINEL\_VALUE));

scan.close();

System.***out***.println();

**for** (Plant plant : myGarden) {

totalCost += plant.getPlantCost();

System.***out***.println(plant);

}

System.***out***.println("You have " + totalPlants + " with a total cost of " + totalCost);

}

**public** **static** **void** printArrayList() {

;

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//CSCI 145 -- Fall 21

**public** **class** Plant {

**protected** String plantName;

**protected** **int** plantCost;

**public** **void** setPlantName(String userPlantName) {

plantName = userPlantName;

}

**public** String getPlantName() {

**return** plantName;

}

**public** **void** setPlantCost(**int** userPlantCost) {

plantCost = userPlantCost;

}

**public** **int** getPlantCost() {

**return** plantCost;

}

**public** String toString() {

**return** "Plant Information:" + "\n Plant name: " + plantName + "\n Cost: " + plantCost + "\n";

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//CSCI 145 -- Fall 21

**public** **class** Flower **extends** Plant {

**private** **boolean** isAnnual;

**private** String colorOfFlowers;

**public** **void** setPlantType(**boolean** userIsAnnual) {

isAnnual = userIsAnnual;

}

**public** **boolean** getPlantType() {

**return** isAnnual;

}

**public** **void** setColorOfFlowers(String userColorOfFlowers) {

colorOfFlowers = userColorOfFlowers;

}

**public** String getColorOfFlowers() {

**return** colorOfFlowers;

}

@Override

**public** String toString() {

**return** **super**.toString() + " Annual: " + isAnnual + "\n Color of flowers: " + colorOfFlowers + "\n";

}

}

Input/output below:

Enter plant or flower: plant Spirea 10

Enter plant or flower: flower Petunia 2 true pink

Enter plant or flower: flower Rose 6 false white

Enter plant or flower: plant Mint 4

Enter plant or flower: neither

Plant Information:

Plant name: Spirea

Cost: 10

Plant Information:

Plant name: Petunia

Cost: 2

Annual: true

Color of flowers: pink

Plant Information:

Plant name: Rose

Cost: 6

Annual: false

Color of flowers: white

Plant Information:

Plant name: Mint

Cost: 4

You have 4 with a total cost of 22

Answer for Question 1

One reason is reusing code. Child class can reuse code or methods from its parent class. We can save a lot of time for not writing similar code for different classes. Another reason is encapsulation. The data in a parent class can be hidden inside its child class where others have no access to it.

Answer for Question 2

Yes, we can achieve polymorphism through interface. An object implements multiple interfaces can refer itself to different type of objects when it is needed. For example, an object of class X can refer itself to Comparable when it is being compared using the compareTo method.

Extra Credit – provide if applicable

Pseudocode below if applicable:

Source code below:

package pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IntegerListTest.java

//

//Provide a menu-driven tester for the IntegerList class.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.util.\*;

public class IntegerListTest {

static IntegerList list = new IntegerList(10);

static Scanner scan = new Scanner(System.in);

// -------------------------------------------------------

// Create a list, then repeatedly print the menu and do what the

// user asks until they quit

// -------------------------------------------------------

public static void main(String[] args) {

printMenu();

int choice = scan.nextInt();

while (choice != 0) {

dispatch(choice);

printMenu();

choice = scan.nextInt();

}

}

// -------------------------------------------------------

// Do what the menu item calls for

// -------------------------------------------------------

public static void dispatch(int choice) {

int loc;

int oldVal;

int newVal;

long begin;

long time;

switch (choice) {

case 0:

System.out.println("Bye!");

break;

case 1:

System.out.println("How big should the list be?");

int size = scan.nextInt();

list = new IntegerList(size);

list.randomize();

break;

case 2:

begin = System.currentTimeMillis();

list.selectionSort();

time = System.currentTimeMillis() - begin;

System.out.println("Total time needed to sort the list is " + time + " miliseconds.");

break;

case 3:

System.out.print("Enter the value to look for: ");

oldVal = scan.nextInt();

begin = System.currentTimeMillis();

loc = list.search(oldVal);

time = System.currentTimeMillis() - begin;

if (loc != -1)

System.out.println("Found at location " + loc);

else

System.out.println("Not in list");

System.out.println("Total time needed to linear search is " + time + " milliseconds.");

break;

case 4:

list.print();

break;

case 5:

System.out.print("Enter the value to be replaced: ");

oldVal = scan.nextInt();

System.out.print("Enter a new value: ");

newVal = scan.nextInt();

list.replaceFirst(oldVal, newVal);

break;

case 6:

System.out.print("Enter the value to be replaced: ");

oldVal = scan.nextInt();

System.out.print("Enter a new value: ");

newVal = scan.nextInt();

list.replaceAll(oldVal, newVal);

break;

default:

System.out.println("Sorry, invalid choice");

}

}

// ------------------------------------------------------

// Print the user's choices

// ------------------------------------------------------

public static void printMenu() {

System.out.println("\n Menu ");

System.out.println(" ====");

System.out.println("0: Quit");

System.out.println("1: Create a new list (\*\* do this first!! \*\*)");

System.out.println("2: Sort the list using selection sort");

System.out.println("3: Find an element in the list using linear search");

System.out.println("4: Print the list");

System.out.println("5: Replace the first matching element found in the list");

System.out.println("6: Replace all the matching elements found in the list");

System.out.print("\nEnter your choice: ");

}

}

**package** pa11;

/\* Java Class: CSCI 145

Modified by: Ivan Leung

Class: Mon/Wed

Date: May 10 2023

Description:

I certify that the code below is my own work.

Exception(s): N/A

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IntegerList.java

//

//Define an IntegerList class with methods to create, fill,

//sort, and search in a list of integers.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** IntegerList {

**int**[] list; // values in the list

// -------------------------------------------------------

// create a list of the given size

// -------------------------------------------------------

**public** IntegerList(**int** size) {

list = **new** **int**[size];

}

// -------------------------------------------------------

// fill array with integers between 1 and 100, inclusive

// -------------------------------------------------------

**public** **void** randomize() {

**for** (**int** i = 0; i < list.length; i++)

list[i] = (**int**) (Math.*random*() \* 100) + 1;

}

// -------------------------------------------------------

// print array elements with indices

// -------------------------------------------------------

**public** **void** print() {

**for** (**int** i = 0; i < list.length; i++)

System.***out***.println(i + ":\t" + list[i]);

}

// -------------------------------------------------------

// return the index of the first occurrence of target in the list.

// return -1 if target does not appear in the list

// -------------------------------------------------------

**public** **int** search(**int** target) {

**int** location = -1;

**for** (**int** i = 0; i < list.length && location == -1; i++)

**if** (list[i] == target)

location = i;

**return** location;

}

// -------------------------------------------------------

// sort the list into ascending order using the selection sort algorithm

// -------------------------------------------------------

**public** **void** selectionSort() {

**int** minIndex;

**for** (**int** i = 0; i < list.length - 1; i++) {

// find smallest element in list starting at location i

minIndex = i;

**for** (**int** j = i + 1; j < list.length; j++)

**if** (list[j] < list[minIndex])

minIndex = j;

// swap list[i] with smallest element

**int** temp = list[i];

list[i] = list[minIndex];

list[minIndex] = temp;

}

}

**public** **void** replaceFirst(**int** oldVal, **int** newVal) {

**int** index = search(oldVal);

**if** (index != -1)

list[index] = newVal;

}

**public** **void** replaceAll(**int** oldVal, **int** newVal) {

**for** (**int** i = 0; i < list.length; ++i)

**if** (list[i] == oldVal)

list[i] = newVal;

}

**public** **void** sortDecreasing() {

;

}

**public** **int** binarySearchD(**int** target) {

**return** 0;

}

}

Input/output below:

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 1

How big should the list be?

10000

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Total time needed to sort the list is 63 miliseconds.

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Total time needed to sort the list is 78 miliseconds.

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 1

How big should the list be?

20000

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Total time needed to sort the list is 142 miliseconds.

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Total time needed to sort the list is 126 miliseconds.

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 1

How big should the list be?

100000

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Total time needed to sort the list is 2846 miliseconds.

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 2

Total time needed to sort the list is 2845 miliseconds.

Menu

====

0: Quit

1: Create a new list (\*\* do this first!! \*\*)

2: Sort the list using selection sort

3: Find an element in the list using linear search

4: Print the list

5: Replace the first matching element found in the list

6: Replace all the matching elements found in the list

Enter your choice: 0

Answer:

It does not run faster for sorting a sorted list, since selection sort will do the same amount of iterations regardless of the list is sorted or not.