## **Laboratory Workbook COMP231 2017/2018 Solutions**

### **Lab 2: Exercise 1 🐼/**

import java.util.Scanner;

public class BMI {

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

public static void main(String[] args) {

System.out.println("Please enter your following weight in kg: ");

double weight = input.nextDouble();

System.out.println("Please enter your following height in cm: ");

double height = input.nextDouble();

double BMI = (kg2lbs(weight)\*703)/(cm2Inch(height)\*cm2Inch(height));

/\* double BMI = kg2lbs(weight)\*703)/(java.lang.Math.pow(cm2Inch(height),2)); \*/

System.out.println("Your BMI is "+ BMI);

intr(BMI);

}

public static double kg2lbs(double kg) {

return (kg\*2.2);

}

public static double cm2Inch(double cm){

return (cm\*0.39);

}

public static void intr(double bmi){

if (bmi<19)

System.out.println("You are underweight :(");

else if (bmi<25)

System.out.println("You are Normal :)");

else

System.out.println("You are Overweight :(");

}

}

### **Lab 2: Exercise 2 🐼/**

import java.util.Scanner;

public class Salary {

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

public static void main(String[] args) {

int hrs =0;

System.out.println("Please enter hours worked: (To end this program, press -1)");

hrs = input.nextInt();

while (hrs !=-1){

System.out.println("Please enter hourly rate of worker: " );

double rate = input.nextDouble();

double Salary = salary(hrs, rate);

System.out.println("Salary is " + Salary+"\n");

System.out.println("Please enter hours worked: (To end this program, press -1)");

}

}

public static double salary(int hrs, double rate ) {

double extra = 0, sum;

if (hrs>40)

extra = (hrs-40)\*(0.5\*rate);

sum= (hrs\*rate)+extra;

return sum;

}

}

### 

### **Lab 2: Exercise 3 🐼/**

import java.util.Scanner;

public class PrimeNumber {

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

public static void main(String[] args) {

System.out.println("Please enter an integer to see if its Prime or not: ");

int n = input.nextInt();

System.out.println("Number is " + isPrime(n));

}

public static boolean isPrime(int n){

for(int x=2; x<n/2; x++) //no need to devise the other half of the number

if(n%x==0) // MOD(%) is remainder

return false;

return true;

}

}

### **Lab 2: Extra 1 🐼/**

public class Average {

public static void main(String[] args) {

int sum=0;

double avg;

for (int x=1; x<=5; x++)

//this only executes one line only without brackets

sum=sum+x;

avg=(double)sum/5;

System.out.println("Average is "+ avg);

}

}

/\*

to run JAVA

search --> cmd

javac ClassName.java

java ClassName

\*/

### **Lab 2: Extra 2** **🐼/**

import java.util.\*;

/\* may replace the star(\*) with a specific class ex: scanner \*/

public class Average {

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

/\* Scanner here is treated as a variable \*/

int sum=0, mark;

double avg;

for (int x=1; x<=5; x++) {

/\* this only executes one line only without brackets \*/

System.out.println("Please enter your mark.");

mark=scan.nextInt();

sum+=mark;

}

avg=(double)sum/5;

System.out.println("Average is "+ avg);

}

}

/\*

to run JAVA

search --> cmd

javac ClassName.java

java ClassName

to use Online Java Compiler, replace "ClassName" in public class with "main"

\*/

### **Lab 2: Extra 3 🐼/**

import java.util.\*;

//may replace \* with a specific class ex: scanner

public class Average

{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

//Scanner here is treated as a variable

int sum=0, mark;

double avg, pass=0; //be careful, pass is a double

for (int x=1; x<=5; x++)

//this only executes one line only without brackets

{

System.out.println("Please enter your mark.");

mark=scan.nextInt();

if (mark>=60)

{

pass++;

}

sum+=mark;

}

avg=(double)sum/5;

pass= (pass/5)\*100;

System.out.println("Average for all grades is "+ avg);

System.out.println("Average of pass is "+ pass+ "%");

}

}

### 

### **Lab 2: Extra 4 🐼/**

//How to calculate average without knowing the number of students

import java.util.\*;

//may replace \* with a specific class ex: scanner

public class Average

{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

int sum=0, mark=0, x=0;

double avg, pass=0;

for (;mark!=-1; x++)

{

System.out.println("Please enter your mark. Press -1 if done.");

mark=scan.nextInt();

if (mark>=60)

{

pass++;

}

sum+=mark;

}

avg=(double)sum/x;

pass= (pass/x)\*100;

System.out.println("Average for all grades is "+ avg);

System.out.println("Average of pass is "+ pass+ "%");

}

}

### **Lab 3: Exercise 1 🐼/**

import java.util.Scanner;

public class ExponentialFactorial {

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

public static void main(String[] args) {

System.out.println("Please enter a number: ");

int x = input.nextInt();

System.out.println("The factorial result is " + expo(x));

}

public static double expo(double x){

double result;

result = 1 +(x/f(1))+((x\*x)/f(2))+((x\*x\*x)/f(3))+((x\*x\*x\*x)/f(4))+((x\*x\*x\*x\*x)/f(5));

return result;

}

public static int f(int n){

/\* factorial is the product of all integers less than than number and is greater than 1. \*/

int sum=1;

for (int x=1; x<=n; x++)

//4! = 1\*2\*3\*4

sum = x\*sum;

return sum;

}

}

### **Lab 3: Exercise 2 🐼/**

public class PerfectNumber

{

public static void main(String[] args) {

for (int x = 1; x <= 1000; x++) {

if (isPerfectNumber(x)) {

System.out.println(x + " is perfect");

}

}

}

public static boolean isPerfectNumber(int number) { //6=3+2+1?

//perfect number is equal to the sum of its divisors

int sum = 0;

for (int x = 1; x <= number/2; x++)

if (number%x == 0)

sum+=x;

if (sum == number)

return true;

return false;

}

}

### **Lab 3: Exercise 3 🐼/**

//binary --|> decimal

import java.util.Scanner;

public class bin2dec {

public static void main(String[] args) {

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

long binaryNumber, decimalNumber = 0, j = 1, remainder;

System.*out*.print("Please input a binary number: ");

binaryNumber = input.nextLong();

input.close();

//closes scanner.

while (binaryNumber != 0) {

remainder = binaryNumber % 10;

decimalNumber = decimalNumber + remainder \* j;

j = j \* 2;

binaryNumber = binaryNumber / 10;

}

System.*out*.println("Its decimal Number: is " + decimalNumber);

}

}

//Extra: decimal --|> binary

import java.util.Scanner;

public class dec2bin {

public static void main(String args[]) {

int dec\_num, quot, i = 1, j;

int bin\_num[] = new int[100];

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

System.*out*.print("Please input a Decimal Number: ");

dec\_num = input.nextInt();

input.close();

quot = dec\_num;

while(quot != 0) {

bin\_num[i++] = quot%2;

quot = quot/2;

}

System.*out*.print("Its binary number is: ");

for(j=i-1; j>0; j--)

System.*out*.print(bin\_num[j]);

}

}

### **Lab 4: Exercise 1 🐼/**

//part1

import java.util.Scanner;

public class Apartment {

//ctrl+shift+f organizes the source code neater in eclipse

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

/\*

scanner here has become global,

be sure to add static in the beginning as well

\*/

public static void main(String[] args) {

System.*out*.print("Please enter total apartment number: ");

int n = *input*.nextInt();

/\*

if it's a double, just add input.nextDouble() and input.nestLine() for string.

\*/

int[] apartment = new int[n];

System.*out*.print("Please fill array in one go:");

for (int i = 0; i < n; i++)

apartment[i] = *input*.nextInt();

System.*out*.println("Number Of People in apartment: " + *numberOfPeople*(apartment));

System.*out*.println("Average number of people per apartment: " + *averageNumPpl*(apartment));

*aboveBelowAverage*(apartment);

}

public static int numberOfPeople(int [] apart) {

int sum=0;

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

sum+=apart[i];

return sum;

}

public static int averageNumPpl (int [] apart) {

return *numberOfPeople*(apart)/apart.length+1;

}

public static void aboveBelowAverage (int [] apart) {

int aboveAvg = 0;

int belowAvg = 0;

int avg = *averageNumPpl*(apart);

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

if (apart[i] < avg)

belowAvg++;

else if (apart[i] > avg)

aboveAvg++;

System.*out*.println("Number of people above average occupance: " + aboveAvg);

System.*out*.println("Number of people below average occupance: " + belowAvg);

}

}

//part 2

import java.util.Scanner;

public class Apartment{

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

public static void main(String[] args) {

System.*out*.print("Please enter total apartments: ");

int rows = *input* .nextInt();

int[][] building = new int[rows][];

for (int i = 0; i < rows; i++)

*readAgesAndResidents*(building, i);

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

}

/\* this method reads the # of occupants and their ages \*/

public static void readAgesAndResidents(int[][] building, int i) {

System.*out*.print("Please enter # of occupants in apartment [" + i + "]: ");

int m = *input*.nextInt();

building[i] = new int[m];

/\* the line above creates additional rows according to # of occupants \*/

for (int j = 0; j < m; j++) {

System.*out*.print("Please enter his/her age: ");

building[i][j] = *input*.nextInt();

}

}

public static double AverageAges(int[][] building) {

int sum = 0, count = 0;

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

/\* this loop is intended for rows only, checks the numbers of residence \*/

for (int j = 0; j < building[i].length; j++, count++)

/\* the loop above is for columns only, gets the ages only \*/

sum += building[i][j];

return (double) sum / count;

}

}

/\*

The goal in this exercise is to create different sets of arrays

using the the number of residence as its rows # and their ages # as its column.

So for example, apartment 0 has two people (60, 65) thus

creating an array of [2][2] (rows x columns). Use a counter

as an index for the apartments in the building and have the

apartments themselves represent an array (residence as its rows # and

their ages # as its column). It would look something like this:

[Row

index]

[0] [60] [65]

and for apartment 1 that has 6 people in it:

[Row

index]

[0] [60] [65]

[1] [50] [50] [20] [18] [15] [10]

and so on......

Code tracing time:

if total apartment # is 4,

first step will create 4 rows:

[Row

index]

[0] .. ..

[1] .. .. .. .. .. ..

[2] .. .. .. ..

[3] .. ..

Note: .. means still undefined

second step with AverageAges() fills them like this:

[Row

index]

[0] [60] [65]

[1] [50] [50] [20] [18] [15] [10]

[2] [45] [25] [21] [15]

[3] [30] [25]

\*/

### **Lab 4: Exercise 2 🐼/**

public class Exc2 {

public static void main(String[] args) {

int[][] data = {{1,3,56,7,9}, {23,65,78},{23,5,6,7}};

System.*out*.println(*linearSearch*(data, 9));

}

public static String linearSearch(int[][] myArray, int key){

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

for(int j = 0; j <myArray[i].length; j++)

if(myArray[i][j] == key)

return "You search result has been found in row " + i + " and column " + j;

return "Your search result has not been found.";

}

}

### **Lab 4: Exercise 3 🐼/**

public class Student {

private int studentId;

private String studentName;

public Student(int stId, String stName){

/\*

For those that already know or are reviewing: yes, this is a constructor.

For those that don’t know: You shall learn about this in the future.

\*/

studentId = stId;

studentName = stName;

}

public void setStudentId(int number){

studentId = number;

}

public int getStduentId(){

return studentId;

}

public void setStudentName(String name){

studentName=name;

}

public String getStudentName(){

return studentName;

}

public String toString() {

return "Student Id is " + studentId + "\tStudent Name is " + studentName;

}

public static void main(String[] args) {

Student [] array = new Student[5];

/\*

This creates an empty (object/reference) array of 5 Students(Type only).

Now we fill this array with content down below:

\*/

array[0] = new Student(120, "Ali Jumah");

array[1] = new Student(130, "Ali Panda");

array[2] = new Student(140, "Panda Jumah");

array[3] = new Student(150, "Mr Panda");

array[4] = new Student(160, "Panda san");

for (int i=0; i<5; i++)

System.*out*.println(array[i].toString());

}

}

/\* OR (Test class)

import java.util.Scanner;

public class Test {

public static void main(String[] args) {

Student[] array = new Student[4];

for(int i=0; i<5; i++){

System.out.println("Please enter student ID: ");

Scanner input = new Scanner(System.in);

int x = input.nextInt();

System.out.println("Please enter student name: ");

int y = input.NextString();

array[i] = new Student(x, y);

}

}

}

\*/

### **Lab 4 Extra 1 incomplete**

import java.util.Scanner;

public class Extraaaaa {

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

public static void main(String[] args) {

int[][]m1 = new int [3][3];

int[][]m2 = new int [3][3];

int[]a = new int [9];

int[]b = new int [9];

InputArray(m1);

InputArray(m2);

//Convert2Dto1D(m1, m2);

//public static boolean equals(int []a, int []b);

}

public static void InputArray(int arr[][]){

/\*

make a loop for moving rows

make a loop for moving columns

I is for rows, j is for columns

\*/

System.out.println("Please enter array: ");

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

for(int j=0; i<arr.length; j++)

arr[i][j]=input.nextInt()

}

public static void Convert2Dto1D(int m1, int m2){

/\*

take row1 and copy to 1D

take row2 and copy it to 1D

use copyArray and decide positioning

reference code:

<https://stackoverflow.com/questions/8935367/convert-a-2d-array-into-a-1d-array>

\*/

System.arraycopy (m1, 00, a, 0, 3);

}

}

/\*

ways to solve this:

first, empty that 2D in a 1D array, sort, then compare

(Or)second, take every element and compare it to the other array

first one is more convenient to use

\*/

### **Lab 5: Exercise 1 🐼/**

import java.util.Date;

public class Employee {

/\* default values are assigned to class variables or at no-arg constructor \*/

private int departmentNo;

private String name;

private long id;

private java.util.Date birthDate;

private java.util.Date hireDate;

private double basicSalary;

public Employee() {

/\* no-argument constructor that initializes default values when constructing the object. \*/

}

public Employee(int departmentNo, String name, long id,

Date birthDate, Date hireDate, double basicSalary) {

/\*

“this” is used here for parameters that have the same name as class variables,

this.class\_variable = parameter\_variable in order to initialize its value(s) directly.

Tip: The constructor was generated automatically from Source menu above.

\*/

this.departmentNo = departmentNo;

this.name = name;

this.id = id;

this.birthDate = birthDate;

this.hireDate = hireDate;

this.basicSalary = basicSalary;

}

/\*

Tip: the setters and getters were generated automatically from Source menu above.

\*/

public int getDepartmentNo() {

return departmentNo;

}

public void setDepartmentNo(int departmentNo) {

this.departmentNo = departmentNo;

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public double getBasicSalary() {

return basicSalary;

}

public void setBasicSalary(double basicSalary) {

this.basicSalary = basicSalary;

}

public String getName() {

return name;

}

public java.util.Date getBirthDate() {

return birthDate;

}

public java.util.Date getHireDate() {

return hireDate;

}

void printEmployeeInfo(){

System.*out*.print("Employee [departmentNo=" + departmentNo + ", name=" + name + ", id=" + id + ", birthDate=" + birthDate

+ ", hireDate=" + hireDate + ", basicSalary=" + basicSalary + "]");

}

}

import java.util.Scanner;

import java.util.Date;

public class Driver {

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

public static void main(String[] args){

Employee [] emp = new Employee[4];

emp[0] = new Employee(1, "Panda 001", 001, new Date(600), new Date(), 1250);

emp[1] = new Employee(1, "Panda 002", 002, new Date(700), new Date(), 1500);

emp[2] = new Employee(1, "Panda 003", 003, new Date(800), new Date(), 1750);

emp[3] = new Employee(1, "Panda 004", 004, new Date(900), new Date(), 2000);

*largestSalary*(emp).printEmployeeInfo();

}

public static Employee largestSalary(Employee [] emp){

Employee max = emp[0];

for (int i = 0; i < emp.length; i++) {

if(emp[i].getBasicSalary() > max.getBasicSalary())

max = emp[i];

}

return max;

}

}

### 

### **Lab 5: Exercise 2 🐼/**

public class MyArray {

private int [] array;

MyArray(int [] array) {

this.array = array;

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

this.array[i] = array[i];

}

int min() {

int min = array[0];

for (int i = 0; i < getSize()-1; i++)

if (array[i] < array[i+1])

min = array[i];

return min;

}

int max() {

int max = array[0];

for (int i = 0; i < getSize()-1; i++)

if (array[i] > array[i+1])

max = array[i];

return max;

}

double average() {

int sum = 0;

for (int i = 0; i < getSize()-1; i++)

sum+=array[i];

return (double)(sum/getSize());

}

int getSize() {

return array.length;

}

void printArray() {

for (int i = 0; i < getSize()-1; i++)

System.*out*.println(array[i]);

}

}

public class Driver {

public static void main(String[] args) {

int [] arr = new int[10];

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

arr[i] = (int) (Math.*random*()\*100);

MyArray integers = new MyArray(arr);

integers.printArray();

}

}

### **Lab 5: Exercise 3 🐼/**

public class City {

private String cityName;

private double longtitude;

private double latitude;

private int temperature;

public City(){

this("Al-Ram", 0.005, 0.0089, 25);

//this serves as a default

}

public City(String name, double longtitude, double latitude, int temp){

this.cityName = name;

this.longtitude = longtitude;

this.latitude = latitude;

this.temperature = temp;

}

public String getCityName() {

return cityName;

}

public void setCityName(String newName) {

cityName = newName;

}

public double getLongtitude() {

return longtitude;

}

public void setLongtitude(double longtitude) {

this.longtitude = longtitude;

}

public double getLatitude() {

return latitude;

}

public void setLatitude(double latitude) {

this.latitude = latitude;

}

public int getTemperature() {

return temperature;

}

public void setTemperature(int temperature) {

this.temperature = temperature;

}

public void printCityInfo(){

System.*out*.println("[City Info: City Name = " + cityName + ", Longtitude = " + longtitude + ", "

+ "Latitude = " + latitude + ", Temperature = " + temperature + "]");

}

}

import java.util.Scanner;

public class Driver {

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

public static void main(String[] args) {

City[] cities = new City [4];

/\* Array must be initialized now \*/

cities[0] = new City("Panda City", 67, 76, 21);

cities[1] = new City("Panda Village", 67, 76, 25);

cities[2] = new City("Panda House", 67, 76, 29);

cities[3] = new City("Panda Land", 67, 76, 31);

System.*out*.println("Please input your city's average temperature: ");

double avgTemp = *input*.nextDouble();

*belowAverage*(cities, avgTemp);

}

static void belowAverage(City [] cities, double avgTemp) {

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

if (cities[i].getTemperature() < avgTemp)

cities[i].printCityInfo();

}

}

/\* OR

public class City {

private String cityName;

private double longitude;

private double latitude;

private int temperature;

public City(){

this("Al-Ram", 0.005, 0.0089, 25);

//this serves as a default

}

public City(String name, double longitude, double latitude, int temp){

this.cityName = name;

this.longitude = longitude;

this.latitude = latitude;

this.temperature = temp;

}

public String getCityName() {

return cityName;

}

public void setCityName(String newName) {

cityName = newName;

}

public double getLongitude() {

return longitude;

}

public void setLongitude(double longitude) {

this.longitude = longitude;

}

public double getLatitude() {

return latitude;

}

public void setLatitude(double latitude) {

this.latitude = latitude;

}

public int getTemperature() {

return temperature;

}

public void setTemperature(int temperature) {

this.temperature = temperature;

}

public void printCityInfo(){

System.*out*.println("[City Info: City Name = " + cityName + ", Longitude = " + longitude + ", "

+ "Latitude = " + latitude + ", Temperature = " + temperature + "]");

}

}

import java.util.Scanner;

public class Driver {

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

public static void main(String[] args) {

/\* A good way to utilize the set and getters \*/

System.out.println("Please enter your array size: ");

int size = input.nextInt();

City[] cities = new City [size];

for(int i = 0; i < size; i++) {

String s = input.nextLine();

//this discards the previous output

System.out.println("Please input your city's name: ");

s = input.nextLine();

/\* How to read more than one word (like Beit haneena for example) \*/

System.out.println("Please input your city's longitude: ");

double longtitude = input.nextDouble();

System.out.println("Please input your city's latitude: ");

double latitude = input.nextDouble();

System.out.println("Please input your city's temperature: ");

int temp = input.nextInt();

cities[i] = new City(s, longtitude, latitude ,temp);

}

System.out.println("Please input your city's average temperature: ");

double avgTemp = input.nextDouble();

belowAverage(cities, avgTemp);

}

static void belowAverage(City [] cities, double avgTemp) {

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

if (cities[i].getTemperature() < avgTemp)

cities[i].printCityInfo();

}

}

\*/

### **Lab 6: Exercise 1 unchecked**

public class Driver {

private String s;

//to use in main

public static String toLowerCase(String f){

String t = f.toLowerCase();

return t;

}

public String getS() {

return s;

}

public String setS(String s) {

this.s = s;

}

public static String reverse(String s){

char c;

char[] chars = s.toCharArray();

for (int i = 0, j = chars.length-1; i < chars.length/2; i++, j--) {

/\* this method is efficient than building say... a million object \*/

c = chars[i];

chars[i] = chars[j];

chars[j] = c;

}

String t = new String(chars);

return t;

/\* OR

String t = "";

for(int i =s.length()-1; i>=0; i--)

t=t+s.charAt(i);

return t;

\*/

}

public static int isPalindrome(){

return 0;

}

}

### **Lab 6: Exercise 2 unchecked**

import java.util.Scanner;  
public class Driver {  
static Scanner input = new Scanner(System.in);  
  
 public static void main(String[] args) {  
 System.out.println("Please enter your String: \n");  
 String s = input.nextLine();  
 s = convertCapital(s);  
 s = reverseString(s);  
 s = toNumbers(s);  
 //s = BeginAndEnd(s);  
 System.out.println("\*\*" + s + "\*\*");  
  
 }  
   
 public static String convertCapital(String s){  
 s = s.toUpperCase();   
 return s;  
 }

public static String reverseString(String s){  
 s = new StringBuilder(s).reverse().toString();  
 return s;  
 }  
   
 public static String toNumbers(String s){  
 /\* O to zero, S to $, L to 1 \*/

s = s.replaceAll("O", "0");  
 s = s.replaceAll("S", "\\$"); // to fix $  
 s = s.replaceAll("L", "1");  
   
 return s;  
 }  
  
 public static String BeginAndEnd(String s){  
 /\* adds \*\* at beginning and at the end \*/

return s;  
 }  
}

### **Lab 6: Exercise 3 unchecked**

import java.util.Scanner;  
public class Driver {  
static Scanner input = new Scanner(System.in);  
  
 public static void main(String[] args) {  
 System.out.println("Please enter your String: \n");  
 String s = input.nextLine();  
 System.out.println(ShortHanded(s));  
 }  
   
 public static String ShortHanded (String s){  
   
 s = s.replaceAll(" and ", " & ");  
 s = s.replaceAll(" to ", " 2 ");  
 s = s.replaceAll(" for ", " 4 ");  
 s = s.replaceAll(" you ", " U ");  
   
 s = s.replaceAll(" u ", "k1");  
 s = s.replaceAll("I ", "k2");  
 s = s.replaceAll(" I ", "k3");  
   
 s = s.replaceAll("[aiueoAIUEO]", "");  
   
 s = s.replaceAll("k1", " u ");  
 s = s.replaceAll("k2", " I ");  
 s = s.replaceAll("k3", " I ");  
 /\* must exclude I and u \*/   
 return s;  
 }  
}

### **Lab 6: Exercise 4 unchecked**

import java.util.Scanner;  
public class Driver {  
static Scanner input = new Scanner(System.in);  
  
 public static void main(String[] args) {  
 System.out.println("Please enter your String: \n");  
 String s = input.nextLine();  
 System.out.println(ShortHanded(s));  
 System.out.println(NumbOfSentences(s));  
 /\* System.out.println(NumOfWords(s)); \*/   
 }  
   
 public static String ShortHanded (String s){  
   
 s = s.replaceAll(" and ", " & ");  
 s = s.replaceAll(" to ", " 2 ");  
 s = s.replaceAll(" for ", " 4 ");  
 s = s.replaceAll(" you ", " U ");  
   
 s = s.replaceAll(" u ", "k1");  
 s = s.replaceAll("I ", "k2");  
 s = s.replaceAll(" I ", "k3");  
   
 s = s.replaceAll("[aiueoAIUEO]", "");  
   
 s = s.replaceAll("k1", " u ");  
 s = s.replaceAll("k2", " I ");  
 s = s.replaceAll("k3", " I ");  
 //must exclude I and u   
   
 return s;  
 }  
   
 public static int NumbOfSentences (String s){  
 // .,!?  
 int counter = 0;  
 s.split("[.,!?]");  
 return counter;  
   
 }

public static int NumOfWords (String s){  
 //word consists of more than 3 characters  
 int counter = 0;  
 if (s.indexOf(" "))  
 counter++;  
 return counter;  
 }

}

/\* use this as reference in future  
   
 String violet;  
 while(Compton.ready())  
 {  
 violet=Compton.readLine();  
 sentenceCount=violet.split("[!?.:]+").length;  
 System.out.println("the number of words in line is " + sentenceCount);  
 }  
\*/

### **Lab 7 🐼/**

// Lab 7 (has only one exercise)  
public class Account {   
 private int id;  
 private double balance;  
 private java.util.Date dateCreated;  
  
 Account() {   
 dateCreated = new java.util.Date();  
 }  
   
 Account(int id, double balance){  
 this.id = id;  
 this.balance = balance;  
 dateCreated = new java.util.Date();  
 }  
   
 public void setId(int id) {  
 this.id = id;  
 }  
   
 public int getId() {  
 return id;  
 }  
   
   
 public double getBalance() {  
 return balance;  
 }  
  
 public void setBalance(double balance) {  
 this.balance = balance;  
 }  
  
 public java.util.Date getDate() {  
 return dateCreated;  
 }  
   
 public void withdraw(double amount) {  
 if (balance-amount >= 0 && amount>0)  
 balance = balance - amount;  
 if (balance-amount < 0)  
 System.out.println("Error");  
 }  
   
 public void deposit(double amount) {  
 balance = balance + amount;  
 }  
  
 public String toString() {  
 return "Account [id=" + id + ", balance=" + balance + ", " + "dateCreated=" + dateCreated + "]";

}   
}  
  
 //UML Class Diagram   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Account \*   
 \* ------------------------------------------------\*   
 \* id: int \*  
 \* balance: double \*  
 \* dateCreated: Date \*  
 \* Account() \*   
 \* Account(id: int, balance: double) \*  
 \* getId(): int \*  
 \* setId(): void \*  
 \* getBalance(): double \*  
 \* setBalance(): void \*  
 \* getDate(): Date \*   
 \* withdraw(amount: double): void \*  
 \* deposit(amount: double): void \*  
 \* toString(): String \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class Checking extends Account {

int OverdraftLimit = 1000;

public Checking(int id, double balance) {

super(id, balance);

}

@Override

public void withdraw(double amount){

if(amount > OverdraftLimit)

System.out.println("Error: Overdraft.");

else if ((getBalance() - amount >= 0 && amount <= 1000 && amount>=0))

setBalance(getBalance() - amount);

else

System.out.println("Error: Incorrect amount.");

}

@Override

public String toString() {

return "Checking: " + super.toString() + "limit is " + OverdraftLimit;

}

}

//UML Class Diagram   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Checking \*   
 \* ------------------------------------------------\*  
 \* Overdraft: int \*   
 \* Checking(id: int, balance: double) \*  
 \* withdraw(amount: double): void \*  
 \* toString(): String \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
  
public class Saving extends Account {  
   
 public Saving() {  
 }  
   
 Saving(int id, double balance){  
 super(id, balance);  
 }  
   
 public String toString() {  
 return "Saving: " + super.toString();  
 }  
}  
  
 //UML Class Diagram   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Savings \*   
 \* ------------------------------------------------\*

\* Saving(id: int, balance: double) \*  
 \* toString(): String \*  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class Driver {

public static void main(String[] args) {

Object test1 = new Account(117, 1500);

/\* declared type is object = actual type is Account, thus casting is necessary. \*/

((Account) test1).deposit(1000);

System.out.println(test1.toString());

Object test2 = new Account(117, 1500);

((Account) test1).withdraw(900);

System.out.println(test2.toString());

Object test3 = new Checking(117, 1500);

((Checking) test3).withdraw(0);

System.out.println(test3.toString());

Object test4 = new Saving(117, 1500);

((Account) test4).deposit(1000);

System.out.println(test4.toString());

}

}

### **Lab 8: Exercise 1 🐼/**

public abstract class Employee implements Comparable<Employee> {

private String firstName;

private String lastName;

private int id;

public Employee(String firstName, String lastName, int id) {

this.firstName = firstName;

this.lastName = lastName;

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public abstract String toString();

public abstract double earning();

@Override

public int compareTo(Employee o) {

return (int)(this.earning() - o.earning());

}

public static double totalEarning(Employee[] ee){

double res= 0;

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

res += ee[i].earning();

return res;

}

public static void sort(Employee[ ] ee){

System.out.println("Before");

for (Employee soso: ee)

System.out.println(soso.toString());

java.util.Arrays.sort(ee);

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

for (Employee e: ee)

System.out.println(e.toString());

}

}

public class SalariedEmployee extends Employee{

private double weeklySalary;

public SalariedEmployee(String firstName, String lastName, int id, double weeklySalary) {

super(firstName, lastName, id);

this.weeklySalary = weeklySalary;

}

public String toString() {

return "SalariedEmployee:["+getId()+"] " + getFirstName() + " " +

getLastName() + " My weekly salray is: " + earning() ;

}

public double earning() {

return weeklySalary;

}

}

public class HourlyEmployee extends Employee{

private double wage;

private int hours;

public HourlyEmployee(String firstName, String lastName, int id, double wage,int hours) {

super(firstName, lastName, id);

this.wage = wage;

this.hours = hours;

}

public String toString() {

return "HourlyEmployee:["+getId()+"] " + getFirstName() + " " +

getLastName() + " My hourly salray is: " + earning() ;

}

public double earning() {

return wage \* hours;

}

}

public class CommisionEmployee extends Employee {

private double rate;

private double grossSale;

public CommisionEmployee(String firstName, String lastName, int id, double rate,double grossSale) {

super(firstName, lastName, id);

this.rate = rate;

this.grossSale = grossSale;

}

public String toString() {

return "CommisionEmployee:["+getId()+"] " + getFirstName() + " " +

getLastName() + " My salray is: " + earning() ;

}

public double earning() {

return rate \* grossSale;

}

}

public class BaseCommisionEmployee extends CommisionEmployee{

private double baseSalary;

public BaseCommisionEmployee(String firstName, String lastName, int id, double rate, double grossSale, double baseSalary) {

super(firstName, lastName, id, rate, grossSale);

this.baseSalary = baseSalary;

}

public String toString() {

return "BaseCommisionEmployee:["+getId()+"] " + getFirstName() + " " +

getLastName() + " My salray is: " + earning() ;

}

public double earning() {

return super.earning() + this.baseSalary ;

}

}

public class Driver {

public static void main(String[] args) {

Employee[] ee = new Employee[5];

ee[0] = new SalariedEmployee("Panda 001", "Abas", 112, 20);

ee[1] = new HourlyEmployee("Panda 002", "Amal", 7657, 1, 700);

ee[2] = new CommisionEmployee("Panda 003", "Soso", 7657, 0.1, 7659);

ee[3] = new BaseCommisionEmployee("Panda 004","fofo", 6565, 0.15, 78678, 500);

ee[4] = new SalariedEmployee("Panda 005", "3o3o", 7657, 7657);

System.out.println("Total earning for all emoloyees is: " + Employee.totalEarning(ee));

Employee.sort(ee);

}

}

### **Lab 8: Exercise 2 🐼/**

public abstract class Shape implements Comparable<Shape> {

protected String color;

protected boolean filled;

public Shape() {

}

public Shape(String color, boolean filled) {

this.color = color;

this.filled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public void setFilled(boolean filled) {

this.filled = filled;

}

public boolean getFilled() {

return filled;

}

public int compareTo(Shape o) {

if (this.getArea() > o.getArea())

/\* this.Shape is equivalent to this (Shape.compareTo) getArea > o's area \*/

return 1;

else if (this.getArea() < o.getArea())

return -1;

return 0;

}

public abstract double getArea();

public abstract String toString();

}

public class Circle extends Shape {

protected double radius;

public Circle() {

}

public Circle(String color, boolean filled, double radius) {

super(color, filled);

this.radius = radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

public double getArea() {

return radius \* radius \* 3.14;

}

public String toString() {

return "Shape: Cirlce [" + "Color: " + color + "\t" + "Filled: " + filled + "\t" +

"Radius: " + radius + "\t" + "Area: " + getArea() + "]";

}

}

public class Rectangle extends Shape {

protected double width;

protected double length;

public Rectangle() {

}

public Rectangle(String color, boolean filled, double width, double length) {

super(color, filled);

this.width = width;

this.length = length;

}

public double getWidth() {

return width;

}

public void setWidth(double width) {

this.width = width;

}

public double getLength() {

return length;

}

public void setLength(double length) {

this.length = length;

}

public double getArea() {

return width \* length;

}

public String toString() {

return "Shape: Rectangle [Color: " + color + "\t" + "Filled: " + filled + "\t"

+ " Width: " + width + "\t" + "Length: " + length + "\t" + "Area: " + getArea() + "]";

}

}

import java.util.ArrayList;

import java.util.Collections;

public class Driver {

public static void main(String[] args) {

ArrayList<Shape> shapes = new ArrayList<>();

shapes.add(new Rectangle("Red", true, 4, 12));

shapes.add(new Circle("Yellow", true, 6));

shapes.add(new Rectangle("Green", true, 4, 7));

shapes.add(new Circle("Blue", true, 4));

shapes.add(new Rectangle("Purple", true, 4, 9));

Collections.sort(shapes);

for(Shape s: shapes )

System.out.println(s);

}

}

/\* OR by a immutable array

public class Driver {

public static void main(String[] args) {

Shape [] shapes = new Shape [5];

shapes[0] = new Circle("Blue", true, 4);

shapes[1] = new Circle("Yellow", true, 6);

shapes[2] = new Rectangle("Green", true, 4, 7);

shapes[3] = new Rectangle("Purple", true, 4, 9);

shapes[4] = new Rectangle("Red", true, 4, 12);

java.util.Arrays.sort(shapes);

}

public static void printShapes(Shape shapes[]) {

for(int i = 0; i < shapes.size(); i++)

if(shapes[i] instanceof Circle)

System.out.println( ( (Circle)shapes[i] ).toString() );

else if(shapes[i] instanceof Rectangle)

System.out.println( ( (Rectangle)shapes[i] ).toString() );

}

}

\*/

### **Lab 9: Exercise 1 🐼/**

public class TestException {  
 public static void main(String[] args) {

try {

System.*out*.println(*bin2dec*("0a11"));

} catch (NumberFormatException e) {

System.*out*.println(e);

}

}

public static int bin2dec(String s) throws NumberFormatException {

int result = 0;

for(int i = s.length()-1, p = 0; i >= 0; i--, p++) {

char c = s.charAt(i);

if(c!= '1' && c != '0')

throw new NumberFormatException ("Error: binary can have 1s and 0s only.");

if(c == '1')

result += Math.*pow*(2, p);

}

return result;

}  
}

### **Lab 9: Exercise 2 🐼/**

public class Circle {

private double radius;

private String color;

public Circle(double newRadius) {

setRadius(newRadius);

}

public void setRadius(double newRadius) throws IllegalArgumentException {

if (newRadius >= 0)

radius = newRadius;

else

throw new IllegalArgumentException("Radius cannot be negative");

}

public static void main(String[] args) {

try {

Circle c1 = new Circle(-5);

Circle c2 = new Circle(5);

System.out.println(c1);

System.out.println(c2);

/\* These statements won't work since it caught

an exception from the beginning.

Consider the objects here as local ones. \*/

}

catch (IllegalArgumentException exception) {

System.out.println(exception);

}

/\* Suppose:

System.out.println(c2);

this won't be able to identify c2 \*/

}

}

### **Lab 9: Exercise 3 🐼/**

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.util.Scanner;

public class TestFiles {

public static void main(String[] args) {

File student = new File("Students.txt");

File output = new File("gradeAverage.txt");

try {

Scanner input = new Scanner(student);

PrintWriter out = new PrintWriter(output);

while (input.hasNext()) {

String s = input.nextLine();

String[] tokens = s.split(" ");

if (tokens.length == 4) {

double avg = ( Integer.parseInt(tokens[1])

+ Integer.parseInt(tokens[2]) + Integer.parseInt(tokens[3]) ) / 3.0;

out.println(tokens[0] + " => " + avg);

}

else

System.out.println("Error: line has missing information.");

}

/\*

If it was for integers:

while(input.hasNext()) {

int i = input.nextInt();

i = i\*2;

out.println(i);

}

\*/

input.close();

out.close();

} catch (FileNotFoundException e) {

System.*out*.println(e);

}

}

}

/\* Another solution to be checked

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.text.DecimalFormat;  
import java.util.Scanner;  
  
public class Average {  
 public static void main (String[]args)throws FileNotFoundException{  
 File student =new File ("C:\\Users\\1171368\\Desktop\\student.txt");  
 File output =new File ("C:\\Users\\1171368\\Desktop\\output.txt");  
 PrintWriter writer= new PrintWriter(output);  
 DecimalFormat d = new DecimalFormat("#.##");  
 try{  
 Scanner in=new Scanner(student);  
 while (in.hasNext()){  
 String s=in.next();  
 int x=in.nextInt();  
 int y=in.nextInt();  
 int z=in.nextInt();  
 int sum=0;  
 //int avg=0;  
 sum=x+y+z;  
 double avg=(double) sum/3;  
 writer.println(s);  
 writer.println(sum);  
 writer.println(d.format(avg));  
 System.out.println(s);  
 System.out.println(sum);  
 System.out.println(d.format(avg));  
 }  
  
 in.close();  
 writer.close();  
 }  
 catch(FileNotFoundException ex ){  
 System.out.println(ex);  
 }  
 }  
}

\*/

### **Lab 10: Exercise 1 🐼/**

import javafx.application.Application;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.RadioButton;  
import javafx.scene.control.ToggleGroup;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Pane;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Circle;  
import javafx.scene.shape.Rectangle;  
import javafx.stage.Stage;  
  
public class exc1 extends Application {  
  
 public static void main(String[] args) {  
 Application.launch();  
 }  
  
 @Override  
 public void start(Stage primaryStage){  
  
 //rectangle + circles  
 Pane p = new Pane();  
 Rectangle r = new Rectangle(100, 20, 50, 160);  
 r.setFill(Color.TRANSPARENT);  
 r.setStroke(Color.BLACK);  
 Circle c1 = new Circle(125, 50, 20);  
 c1.setStroke(Color.BLACK);  
 c1.setFill(Color.RED);  
 Circle c2 = new Circle(125, 100, 20);  
 c2.setStroke(Color.BLACK);  
 c2.setFill(Color.TRANSPARENT);  
 Circle c3 = new Circle(125, 150, 20);  
 c3.setFill(Color.TRANSPARENT);  
 c3.setStroke(Color.BLACK);   
 p.getChildren().addAll(r, c1, c2, c3);  
  
   
 //Hbox + RadioButtons  
 HBox RadioButtons = new HBox();  
 RadioButtons.setSpacing(20);  
 RadioButton rbRed = new RadioButton("Red");  
 rbRed.setSelected(true);  
 RadioButton rbYellow = new RadioButton("Orange");  
 RadioButton rbGreen = new RadioButton("Green");  
 RadioButtons.getChildren().addAll(rbRed, rbYellow, rbGreen);  
 RadioButtons.setAlignment(Pos.CENTER);  
  
 //toggle group  
 ToggleGroup group = new ToggleGroup();  
 rbRed.setToggleGroup(group);  
 rbYellow.setToggleGroup(group);  
 rbGreen.setToggleGroup(group);  
 rbRed.setSelected(true);  
   
  
 //border pane  
 BorderPane pane = new BorderPane();  
 pane.setCenter(p);  
 pane.setBottom(RadioButtons);  
   
 Scene scene = new Scene(pane, 250, 300);  
 primaryStage.setTitle("Traffic Light"); //title  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
}

### **Lab 10: Exercise 2 🐼/**

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.layout.GridPane;

import javafx.scene.paint.Color;

import javafx.scene.shape.Rectangle;

import javafx.stage.Stage;

public class Checkboard extends Application{

public static void main(String[] args) {

Application.launch(args);

}

public void start(Stage primaryStage) throws Exception {

GridPane board = new GridPane();

// Create 8 x 8 rectangles and add to pane

int count = 0;

double s = 50; // side of square

for (int i = 0; i < 8; i++) {

count++;

for (int j = 0; j < 8; j++) {

Rectangle r = new Rectangle(s, s, s, s);

if (count % 2 == 1)

r.setFill(Color.WHITE);

board.add(r, j, i);

count++;

}

}

Scene scene = new Scene(board);

primaryStage.setTitle("Checkboard");

primaryStage.setScene(scene);

primaryStage.show();

}

}

### **Lab 10: Exercise 3 🐼/**

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.control.TextField;

import javafx.scene.control.Label;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.control.RadioButton;

import javafx.scene.control.ToggleGroup;

import javafx.geometry.Pos;

import javafx.geometry.Insets;  
  
public class JavaFX TextField extends Application {  
 protected TextField tfTextField = new TextField();  
 protected TextField tfColumnSize = new TextField();

public static void main(String[] args) {

Application.launch(args);

}

@Override // must override the start method in the Application class  
 public void start(Stage primaryStage) {  
 // Set properties for text fields  
 tfTextField.setText("JavaFX");  
 tfTextField.setAlignment(Pos.BOTTOM\_CENTER);  
 tfColumnSize.setAlignment(Pos.BOTTOM\_RIGHT);  
 tfColumnSize.setPrefColumnCount(3);  
 tfTextField.setPrefColumnCount(12);  
 tfColumnSize.setText("12");  
  
 // Create three radio buttons  
 RadioButton rbLeft = new RadioButton("Left");  
 RadioButton rbCenter = new RadioButton("Center");  
 RadioButton rbRight = new RadioButton("Right");  
 rbCenter.setSelected(true);  
  
 // Create a toggle group  
 ToggleGroup group = new ToggleGroup();  
 rbLeft.setToggleGroup(group);  
 rbCenter.setToggleGroup(group);  
 rbRight.setToggleGroup(group);  
   
 // Create four hbox  
 HBox paneForRadioButtons = new HBox(5);  
 paneForRadioButtons.getChildren().addAll(rbLeft, rbCenter, rbRight);  
 paneForRadioButtons.setAlignment(Pos.BOTTOM\_LEFT);  
  
 HBox paneForColumnSize = new HBox(5);  
 paneForColumnSize.getChildren().addAll(  
 new Label("Column Size"), tfColumnSize);  
  
 HBox paneForTextField = new HBox(5);  
 paneForTextField.setAlignment(Pos.CENTER);  
 paneForTextField.getChildren().addAll(  
 new Label("Text Field"), tfTextField);  
  
 HBox hbox = new HBox(10);  
 hbox.getChildren().addAll(paneForRadioButtons, paneForColumnSize);  
  
 // Create a vBox and place nodes in it  
 VBox pane = new VBox(10);  
 pane.setPadding(new Insets(5, 5, 5, 5));  
 pane.getChildren().addAll(paneForTextField, hbox);

// Create a scene and place it in the stage

Scene scene = new Scene(pane);

// Set the stage title

primaryStage.setTitle("JavaFX TextField");

// Place the scene in the stage

primaryStage.setScene(scene);

// Display the stage

primaryStage.show();   
}

### **Lab 11: Exercise 1 🐼/**

import javafx.application.Application;  
import javafx.event.ActionEvent;  
import javafx.event.EventHandler;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.RadioButton;  
import javafx.scene.control.ToggleGroup;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Pane;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Circle;  
import javafx.scene.shape.Rectangle;  
import javafx.stage.Stage;  
  
public class TrafficLight extends Application {  
  
 Circle c1 = new Circle(125, 50, 20);  
 Circle c2 = new Circle(125, 100, 20);  
 Circle c3 = new Circle(125, 150, 20);  
   
 RadioButton rbRed = new RadioButton("Red");  
 RadioButton rbOrange = new RadioButton("Orange");  
 RadioButton rbGreen = new RadioButton("Green");  
  
 public static void main(String[] args) {  
 Application.launch();  
 }  
  
 @Override  
 public void start(Stage primaryStage){  
  
 //rectangle + circles  
 Pane p = new Pane();  
 Rectangle r = new Rectangle(100, 20, 50, 160);  
 r.setFill(Color.TRANSPARENT);  
 r.setStroke(Color.BLACK);  
 c1.setStroke(Color.BLACK);  
 c1.setFill(Color.RED);  
 c2.setStroke(Color.BLACK);  
 c2.setFill(Color.TRANSPARENT);  
 c3.setFill(Color.TRANSPARENT);  
 c3.setStroke(Color.BLACK);  
 p.getChildren().addAll(r, c1, c2, c3);  
   
 BHandler bh = new BHandler();  
 rbRed.setOnAction(bh);  
 rbOrange.setOnAction(bh);  
 rbGreen.setOnAction(bh);  
  
  
 //Hbox + RadioButtons  
 HBox RadioButtons = new HBox();  
 RadioButtons.setSpacing(20);  
 rbRed.setSelected(true);  
   
 RadioButtons.getChildren().addAll(rbRed, rbOrange, rbGreen);  
 RadioButtons.setAlignment(Pos.CENTER);  
  
 //toggle group  
 ToggleGroup group = new ToggleGroup();  
 rbRed.setToggleGroup(group);  
 rbOrange.setToggleGroup(group);  
 rbGreen.setToggleGroup(group);  
 rbRed.setSelected(true);  
  
  
 //border pane  
 BorderPane pane = new BorderPane();  
 pane.setCenter(p);  
 pane.setBottom(RadioButtons);  
  
 Scene scene = new Scene(pane, 250, 300);  
 primaryStage.setTitle("Traffic Light");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 //inner class  
 class BHandler implements EventHandler<ActionEvent> {  
 public void handle (ActionEvent e){  
 if (e.getSource() == rbRed) {  
 c1.setFill(Color.RED);  
 c2.setFill(Color.TRANSPARENT);  
 c3.setFill(Color.TRANSPARENT);  
 }  
  
 if (e.getSource() == rbOrange) {  
 c1.setFill(Color.TRANSPARENT);  
 c2.setFill(Color.ORANGE);  
 c3.setFill(Color.TRANSPARENT);  
 }  
  
 if (e.getSource() == rbGreen) {  
 c1.setFill(Color.TRANSPARENT);  
 c2.setFill(Color.TRANSPARENT);  
 c3.setFill(Color.GREEN);  
 }  
 }  
 }  
}

### **Lab 11: Exercise 2 🐼/**

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.control.TextField;

import javafx.scene.control.Label;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.control.RadioButton;

import javafx.scene.control.ToggleGroup;

import javafx.geometry.Pos;

import javafx.geometry.Insets;

import javafx.scene.input.KeyCode;

public class Driver extends Application {

protected TextField tfTextField = new TextField();

protected TextField tfColumnSize = new TextField();

public static void main(String[] args) {

Application.launch(args);

}

@Override

public void start(Stage primaryStage) {

// Set properties for text fields

tfTextField.setText("JavaFX");

tfTextField.setAlignment(Pos.BOTTOM\_CENTER);

tfColumnSize.setAlignment(Pos.BOTTOM\_RIGHT);

tfColumnSize.setPrefColumnCount(3);

tfTextField.setPrefColumnCount(12);

tfColumnSize.setText("12");

// Create three radio buttons

RadioButton rbLeft = new RadioButton("Left");

RadioButton rbCenter = new RadioButton("Center");

RadioButton rbRight = new RadioButton("Right");

rbCenter.setSelected(true);

// Create a toggle group

ToggleGroup group = new ToggleGroup();

rbLeft.setToggleGroup(group);

rbCenter.setToggleGroup(group);

rbRight.setToggleGroup(group);

// Create four hbox

HBox paneForRadioButtons = new HBox(5);

paneForRadioButtons.getChildren().addAll(rbLeft, rbCenter, rbRight);

paneForRadioButtons.setAlignment(Pos.BOTTOM\_LEFT);

HBox paneForColumnSize = new HBox(5);

paneForColumnSize.getChildren().addAll(

new Label("Colum Size"), tfColumnSize);

HBox paneForTextField = new HBox(5);

paneForTextField.setAlignment(Pos.CENTER);

paneForTextField.getChildren().addAll(

new Label("Text Field"), tfTextField);

HBox hbox = new HBox(10);

hbox.getChildren().addAll(paneForRadioButtons, paneForColumnSize);

// Create a vBox and place nodes in it

VBox pane = new VBox(10);

pane.setPadding(new Insets(5, 5, 5, 5));

pane.getChildren().addAll(paneForTextField, hbox);

// Create and register the handlers

rbLeft.setOnAction(e -> {

if (rbLeft.isSelected()) {

tfTextField.setAlignment(Pos.BOTTOM\_LEFT);

}

});

rbCenter.setOnAction(e -> {

if (rbCenter.isSelected()) {

tfTextField.setAlignment(Pos.BOTTOM\_CENTER);

}

});

rbRight.setOnAction(e -> {

if (rbRight.isSelected()) {

tfTextField.setAlignment(Pos.BOTTOM\_RIGHT);

}

});

tfColumnSize.setOnKeyPressed(e -> {

if (e.getCode() == KeyCode.ENTER) {

tfTextField.setPrefColumnCount(Integer.parseInt(

tfColumnSize.getText()));

}

});

// Create a scene and place it in the stage

Scene scene = new Scene(pane);

// Set the stage title

primaryStage.setTitle("JavaFX TextField");

// Place the scene in the stage

primaryStage.setScene(scene);

// Display the stage

primaryStage.show();

}

}

### **Lab 11: Exercise 3 🐼/**

/\* Takes in info and saves it in a file \*/

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.RadioButton;

import javafx.scene.control.TextField;

import javafx.scene.control.ToggleGroup;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

public class Driver extends Application {

protected int stuID;

protected String stuName;

protected String DepName;

protected String avg;

protected TextField studentID = new TextField();

protected TextField studentName = new TextField();

protected TextField depName = new TextField();

public static void main(String[] args) {

Application.*launch*(args);

}

public void printInfo() throws FileNotFoundException {

PrintWriter output = new PrintWriter(new File("studentInfo.txt"));

output.print(stuID + " " + stuName + " " + avg + " " + DepName);

output.close();

}

@Override

public void start(Stage primaryStage) {

Button add = new Button("add");

add.setPrefSize(50, 20);

add.setAlignment(Pos.*CENTER*);

// Create three radio buttons and set them up in a HBox

RadioButton excellent = new RadioButton("Excellent");

RadioButton veryGood = new RadioButton("Very Good");

RadioButton good = new RadioButton("Good");

HBox average = new HBox(excellent, veryGood, good);

// Create a toggle group

ToggleGroup gpa = new ToggleGroup();

excellent.setToggleGroup(gpa);

veryGood.setToggleGroup(gpa);

good.setToggleGroup(gpa);

//create GridPane

GridPane gp = new GridPane();

gp.setHgap(5);

gp.setVgap(5);

gp.setPadding(new Insets(15, 15, 15, 15));

gp.add(new Label("Student ID"), 0, 0);

gp.add(studentID, 1, 0);

gp.add(new Label("Student Name"), 0, 1);

gp.add(studentName, 1, 1);

gp.add(new Label("Average"), 0, 2);

gp.add(average, 1, 2);

gp.add(new Label("Deprtment Name"), 0, 3);

gp.add(depName, 1, 3);

gp.add(add, 1, 4);

// Create and register the handlers

excellent.setOnAction(e -> {

if (excellent.isSelected()) {

avg = "Excellent";

}

});

veryGood.setOnAction(e -> {

if (veryGood.isSelected()) {

avg = "Very Good";

}

});

good.setOnAction(e -> {

if (good.isSelected()) {

avg = "Good";

}

});

add.setOnAction(e -> {

// if statements for error handling required here

stuID = Integer.*parseInt*(studentID.getText());

stuName = studentName.getText();

DepName = depName.getText();

try {

printInfo();

} catch (FileNotFoundException e1) {

e1.printStackTrace();

}

//System.out.println("Button pressed\n" + stuID + " " + stuName + " " + avg + " " + DepName);

});

// Create a scene and place it in the stage

Scene scene = new Scene(gp);

// Set the stage title

primaryStage.setTitle("student Info");

// Place the scene in the stage

primaryStage.setScene(scene);

// Display the stage

primaryStage.show();

}

}

### **Lab 12 🐼/**

// Lab 12 (has only one exercise)

public class Student implements Comparable<Student> {

private int id;

private String name;

private Major major;

private double average;

private static int *numOfCreatedStudents*;

public Student(int id, String name, Major major, double average) {

super();

this.id = id;

this.name = name;

this.major = major;

this.average = average;

*numOfCreatedStudents*++;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Major getMajor() {

return major;

}

public void setMajor(Major major) {

this.major = major;

}

public double getAverage() {

return average;

}

public void setAverage(double average) {

this.average = average;

}

public static int getNumOfCreatedStudents() {

return *numOfCreatedStudents*;

}

public static void setNumOfCreatedStudents(int numOfCreatedStudents) {

Student.*numOfCreatedStudents* = numOfCreatedStudents;

}

@Override

public String toString() {

return "Student [id=" + id + ", name=" + name + ", major=" + major + ", average=" + average + "]";

}

@Override

public int compareTo(Student o) {

if (this.getAverage() > o.getAverage())

return 1;

else if (this.getAverage() < o.getAverage())

return -1;

return 0;

}

}

abstract public class Major implements MajorInterface {  
 private String title = CS;  
 private int plan;  
 private int years = FIVE\_YEAR;  
 private String description;  
  
 public Major(String title, int plan, int years, String description) {  
 this.title = title;  
 this.plan = plan;  
 this.years = years;  
 this.description = description;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public int getPlan() {  
 return plan;  
 }  
  
 public void setPlan(int plan) {  
 this.plan = plan;  
 }  
  
 public int getYears() {  
 return years;  
 }  
  
 public void setYears(int years) {  
 this.years = years;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 @Override  
 public abstract String toString();  
  
  
}

public class CS extends Major{  
  
 public CS() {  
 super(CS, 120, FOUR\_YEAR, "CS dept");  
 }  
  
 public void displayInfo() {  
 System.out.println(toString());  
 }  
  
 @Override  
 public String toString() {  
 return " CS Major [title=" + getTitle() + ", plan=" + getPlan() +  
 ", years=" + getYears() + ", description="  
 + getDescription() + "]";  
 }  
}

public class CSE extends Major{  
  
 public CSE() {  
 super(CSE, 160, FIVE\_YEAR, "CSE dept");  
 }  
 public void displayInfo() {  
 System.out.println(toString());  
 }  
  
 @Override  
 public String toString() {  
 return " CSE Major [title=" + getTitle() + ", plan=" + getPlan() +  
 ", years=" + getYears() + ", description="  
 + getDescription() + "]";  
 }  
}

public interface MajorInterface {  
 int FIVE\_YEAR = 5;  
 int FOUR\_YEAR = 4;  
 String CS = "CS";  
 String CSE = "CSE";  
  
 void displayInfo();  
 int getPlan();  
}

import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.RadioButton;

import javafx.scene.control.TextField;

import javafx.scene.control.ToggleGroup;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

public class Driver extends Application {

ArrayList<Student> students = new ArrayList<>();

public static void main(String[] args) {

Application.launch(args);

}

public Student topStudent(String major) {

Collections.sort(students);

Student student = null;

for (int i = students.size()- 1; i >=0 ; i--)

if (students.get(i).getMajor().getTitle().equals(major) ) {

student = students.get(i);

break;

}

return student;

}

public double studentsAverage(String major) {

double sum = 0;

int c = 0;

for (int i = 0; i < students.size() ; i++)

if (students.get(i).getMajor().getTitle().equals(major) ) {

sum += students.get(i).getAverage();

c++;

}

return (sum / c);

}

@Override

public void start(Stage primaryStage) {

// read file and create student objects

try {

Scanner input = new Scanner(new File("students.txt"));

while (input.hasNext()) {

String s = input.nextLine();

String[] tokens = s.split(":");

if (tokens.length == 4) {

int id = Integer.parseInt(tokens[0].trim());

String name = tokens[1].trim();

String majorTitle = tokens[2].trim();

double avg = Double.parseDouble(tokens[3].trim());

Major major = (majorTitle.equals("CS")) ? new CS() : new CSE();

students.add(new Student(id, name, major, avg));

}

}

input.close();

} catch (FileNotFoundException e) {

}

// create new BorderPane

BorderPane bp = new BorderPane();

bp.setPadding(new Insets (2, 2, 2, 2));

//create HBox

Label major = new Label("Major: ");

RadioButton CS = new RadioButton("CS");

RadioButton CSE = new RadioButton("CSE");

HBox hb = new HBox(major, CS, CSE);

hb.setSpacing(20);

hb.setAlignment(Pos.CENTER);

bp.setTop(hb);

//toggle group

ToggleGroup tg = new ToggleGroup();

CS.setToggleGroup(tg);

CSE.setToggleGroup(tg);

CS.setSelected(true);

// create GridPane

GridPane gp = new GridPane();

gp.setHgap(5);

gp.setVgap(5);

Button topMajorStudent = new Button("Top Major Student");

TextField topMajor = new TextField();

topMajor.setPrefWidth(300);

Button majorAverage = new Button("Major Average");

TextField avgField = new TextField();

gp.add(topMajorStudent, 0, 0);

gp.add(topMajor, 1, 0);

gp.add(majorAverage, 0, 1);

gp.add(avgField, 1, 1);

gp.setPadding(new Insets (4, 4, 4, 4));

bp.setCenter(gp);

//create count label

Label total = new Label("The total number of students is: " + Student.getNumOfCreatedStudents() );

bp.setBottom(total);

// Event Handlers

topMajorStudent.setOnAction(e -> {

if(CS.isSelected()){

topMajor.setText(topStudent("CS").getName());

}

else{

topMajor.setText(topStudent("CSE").getName());

}

});

majorAverage.setOnAction(e -> {

if(CS.isSelected()){

avgField.setText(studentsAverage("CS")+"");

}

else{

avgField.setText(studentsAverage("CSE")+"");

}

});

primaryStage.setTitle("COMP231: Practical Exam (23/8/2017)");

Scene scene = new Scene(bp);

primaryStage.setScene(scene);

primaryStage.show();

// Collections.sort(students);

// for (int i = 0; i < students.size(); i++)

// System.out.println(students.get(i));

}

}

### **1st Quiz 🐼/**

public class quiz1 {

/\* Write a code that prints the first 100 digits that are

Prime and Palindrome. Print every 10 digits in one line. \*/

public static void main(String[] args) {

for(int i =0, n = 2; i < 100;n++){

if (isPrime(i) && isPalindrome(i))

System.out.print(n + “\t”);

i++;

if(i%10 == 0)

System.out.print(“\n”);

}

}

public static boolean isPrime(int x){

for(int i= 2; i <= n/2;i++)

if (n%i == 0)

return false;

return true;

}

public static boolean isPalindrome(int n){

int rev = 0, int temp = n;

while(n > 0){

rev = (rev\*10) + (n%10);

n /= 10;

}

return (rev == temp);

}

}

### **2nd Quiz Incomplete**

public class quiz2 {

/\* Write a code that calculates the quadratic formula \*/

}

### **3rd Quiz Incomplete**

public class quiz3 {

/\* Employee sorting blah blah blaah \*/

}

### **4th Quiz 🐼/**

/\* Question: Create a Book clas Book(int isbn, String Author, String Title). Read from a file called books that has a list of books with their (1782 , Ali , Hello World). Create an object of Book while reading from that file and sort them according to their isbn. \*/

public class Book implements Comparable<Book>{

private int isbn;

private String author;

private String title;

Book(int isbn, String author, String title) {

this.isbn = isbn;

this.author = author;

this.title = title;

}

public int getIsbn() {

return isbn;

}

public void setIsbn(int isbn) {

this.isbn = isbn;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String toString() {

return "Book [isbn=" + isbn + ", author=" + author + ", title=" + title + "]";

}

public int compareTo(Book o) {

if (this.getIsbn() > o.getIsbn() )

return 1;

if (this.getIsbn() < o.getIsbn() )

return -1;

return 0;

}

}

import java.io.File;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class Driver {

public static void main(String[] args) {

File books = new File("books.txt");

ArrayList<Book> bookList = new ArrayList<Book>();

try {

Scanner input = new Scanner(books);

while (input.hasNext()) {

String s = input.nextLine();

String tokens[] = s.split(" , ");

if (tokens.length == 3)

bookList.add(new Book(Integer.parseInt(tokens[0]), tokens[1], tokens[2]));

else

System.out.println("Error: Missing line");

}

input.close();

} catch (Exception e) {

System.out.println("Error in File");

}

Collections.sort(bookList);

System.out.println(bookList.toString());

}

}