COSC 436

Fall 2018

# ASSIGNMENT 2 – Java Fundamentals

30 pts.

**Due Thursday, September 27th**

Type up your answers to the following questions, and submit as one Word document to Blackboard.

1. Design and implement a Temperature class able to handle temperatures in either Fahrenheit or   
 Celsius. Include appropriate constructors (including a copy constructor), getters and setters, toString,   
 and belowFreezing methods. (5 pts.)

2. For your Temperature class (from problem 1), implement an appropriate equals method. Provide examples to demonstrate the use of your equals method. (3 pts.)

3. Give example use of objects of type Temperature (from problem 1) for each of the three means of   
 object access discussed in class (and given in the lecture slides). Include sufficient code to   
 demonstrate its use. (3 pts.)

**public class** Temperature {  
 **private int farenheit**;  
 **private double celsius**;  
  
 **public** Temperature(){  
 **farenheit** = 0;  
 **celsius** = 0.0;  
 }  
  
 **public** Temperature(**int** farenheit){  
 **this**.**farenheit** = farenheit;  
 **celsius** = (**double**) (farenheit-32)\*5/9;  
 }  
  
 **public** Temperature(**double** c){  
 **celsius** = c;  
 **farenheit** = (**int**) c\*9/5+32;  
 }  
  
 **public** Temperature(**int** f, **double** c){  
 **farenheit** = f;  
 **celsius** = c;  
 }  
  
 **public** Temperature(Temperature temp){  
 **farenheit** = temp.**farenheit**;  
 **celsius** = temp.**celsius**;  
 }  
  
 **public int** getFarenheit(){  
 **return farenheit**;  
 }  
  
 **public double** getCelsius(){  
 **return celsius**;  
 }  
  
 **public void** setFarenheit(**int** tempF){  
 **farenheit** = tempF;  
 }  
  
 **public void** setCelsius(**double** tempC){  
 **celsius** = tempC;  
 }  
  
 @Override  
 **public** String toString(){  
 **return** String.*format*(**"The temperature is:\nFarenheit: %d\nCelsius: %f\n"**, **farenheit**, **celsius**);  
 }  
  
 **public boolean** belowFreezing(){  
 **return** (**farenheit** <= 32 || **celsius** <= 0.0);  
 }  
  
 @Override  
 **public boolean** equals(Object otherTemp){  
 **return** (otherTemp.getClass() == Temperature.**class**) && (**farenheit** == **new** Temperature((Temperature)otherTemp).**farenheit** && **celsius** == ((Temperature)otherTemp).**celsius**);  
 }  
}

**import** java.util.Scanner;  
  
**public class** Client {  
 **public static void** main(String[] args){  
 Temperature t1 = **new** Temperature();  
 Temperature t2, t3, t4, t5;  
 **int** fahrenheit;  
 **double** celsius;  
 Scanner input = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"Please enter the temperature in fahrenheit: "**);  
 fahrenheit = *getInt*(input);  
  
 System.***out***.println(**"Initializing the second temperature object..."**);  
 t2 = **new** Temperature(fahrenheit);  
  
 System.***out***.println(**"Please enter the temperature in Celsius: "**);  
 celsius = *getDouble*(input);  
  
 System.***out***.println(**"Initializing the third temperature object..."**);  
 t3 = **new** Temperature(celsius);  
  
 System.***out***.println(**"Now initializing another temperature object with the two temperatures entered..."**);  
 t4 = **new** Temperature(fahrenheit, celsius);  
  
 System.***out***.println(**"Now initializing the final temperature object with the previous temperature object..."**);  
 t5 = **new** Temperature(t4);  
  
 System.***out***.println(**"Testing getters and setters:"**);  
 System.***out***.printf(**"getFahrenheit on first object: %d\ngetCelsius on second object: %f\n"**, t2.getFarenheit(), t3.getCelsius());  
  
 System.***out***.println(**"Please enter in another temperature for Fahrenheit: "**);  
 t3.setFarenheit(*getInt*(input));  
  
 System.***out***.println(**"Please enter in another temperature for Celsius: "**);  
 t2.setCelsius(*getDouble*(input));  
  
 System.***out***.println(**"After call to equals method on t2 and t3: "**);  
 **if**(t2.equals(t3)){  
 System.***out***.println(**"The two temperatures are equal!"**);  
 }  
 **else**{  
 System.***out***.println(**"The two temperatures are not equal."**);  
 }  
  
 System.***out***.println(**"After call to equals method on t4 and t5: "**);  
 **if**(t4.equals(t5)){  
 System.***out***.println(**"The two temperatures are equal!"**);  
 }  
 **else**{  
 System.***out***.println(**"The two temperatures are not equal."**);  
 }  
  
 System.***out***.println(**"Testing belowFreezing method: "**);  
 **if**(t4.belowFreezing()){  
 System.***out***.println(**"It is freezing!"**);  
 }  
 **else**{  
 System.***out***.println(**"It is not freezing."**);  
 }  
  
 System.***out***.println(**"Finally, "** + t5);  
 }  
  
 **public static int** getInt(Scanner in){  
 **while**(!in.hasNextInt()){  
 System.***out***.println(**"Please enter an integer value for your temperature: "**);  
 in.next();  
 }  
 **return** in.nextInt();  
 }  
  
 **public static double** getDouble(Scanner in){  
 **while**(!in.hasNextDouble()){  
 System.***out***.println(**"Please enter an integer value for your temperature: "**);  
 in.next();  
 }  
 **return** in.nextDouble();  
 }  
}

4. Give sufficient code to demonstrate how a class can be designed such that it maintains how many   
 instances of itself exist at any given point in time. (3 pts.)

5. Assume that there is a Java interface named Measurable. The only method in the interface is method  
 getMeasure(), which returns a string describing an object’s measurement (if of type Measurement).   
 Assume that there are three classes of type Measureable: Rectangle (with measurement height and   
 width, in inches), Package (with measurement of weight in lbs and ounces), and WindSpeed (with   
 measurement in miles per hour). Give sufficient code to demonstrate the use of polymorphism   
 involving objects of each of these three types. (6 pts.)

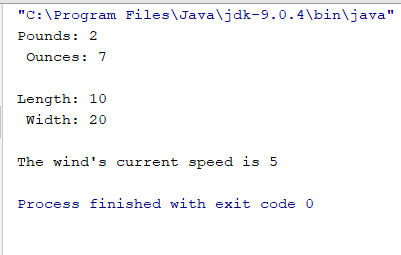
**package** Geometry;  
  
**public interface** Measurable {  
 **public** String getMeasure();  
}

**public class** Rectangle **implements** Measurable {  
 **int weightOZ**, **weightLBS**;  
 **public** Rectangle(**int** weightLBS, **int** weightOZ) {  
 **this**.**weightLBS** = weightLBS;  
 **this**.**weightOZ** = weightOZ;  
 }  
 @Override  
 **public** String getMeasure() {  
 **return** String.*format*(**"Pounds: %d\n Ounces: %d\n"**, **weightLBS**, **weightOZ**);  
 }  
}

**public class** WindSpeed **implements** Measurable {  
 **private int speed**;  
 **private int distance**;  
 **private int timeInSeconds**;  
  
 **public** WindSpeed(**double** v){  
 **speed** = 0;  
 **distance** = 0;  
 **timeInSeconds** = 0;  
 }  
  
 **public** WindSpeed(**int** d, **int** t){  
 **distance** = d;  
 **timeInSeconds** = t;  
 **speed** = **distance**/**timeInSeconds**;  
 }  
  
 @Override  
 **public** String getMeasure() {  
 **return** String.*format*(**"The wind's current speed is %d"**, **speed**);  
 }  
}

**public class** Package **implements** Measurable {  
 **private int length**;  
 **private int width**;  
 **private int height**;  
 **private int weight**;  
 **private int surfaceArea**;  
 **private int volume**;  
  
 **public** Package(**int** length, **int** width){  
 **this**.**length**=length;  
 **this**.**width**=width;  
 }  
  
 @Override  
 **public** String getMeasure() {  
 **return** String.*format*(**"Length: %d\n Width: %d\n"**, **length**, **width**);  
 }  
}

**import** Geometry.Measurable;  
**import** Geometry.Package;  
**import** Geometry.Rectangle;  
**import** Geometry.WindSpeed;  
  
**public class** MeasurableClient {  
 **public static void** main(String[] args) {  
 Measurable m;  
  
 m = **new** Rectangle(2, 7);  
 System.***out***.println(m.getMeasure());  
  
 m = **new** Package(10, 20);  
 System.***out***.println(m.getMeasure());  
  
 m = **new** WindSpeed(10, 2);  
 System.***out***.println(m.getMeasure());  
 }  
}



6. Declare an interface named Sized, containing only one method, getSize, that returns a string. Assume   
 that there exist a number of classes declared of type Sized in a particular application, specifically,   
 StudentRecords (where size is the number of records), SemesterSchedule (where size it the number   
 of credits taking), Department (where size is the number of faculty in a particular department), and   
 Building (where size is the total number of square feet). (10 pts.)

**package** Geometry;  
  
**public interface** Sized {  
 **public** String getSize();  
}

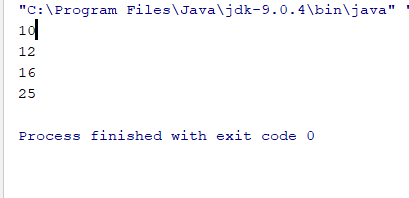
**public class** SemesterSchedule **implements** Sized {  
 String **sized**;  
  
 **public** SemesterSchedule (String inSize) {  
 **this**.**sized** = inSize;  
 }  
  
 @Override  
 **public** String getSize() {  
 **return sized**;  
 }  
}

**public class** StudentRecords **implements** Sized{  
 String **sized**;  
  
 **public** StudentRecords(String inSize) {  
 **this**.**sized** = inSize;  
 }  
  
 @Override  
 **public** String getSize() {  
 **return sized**;  
 }  
}

**public class** Department **implements** Sized {  
 String **sized**;  
  
 **public** Department (String inSize) {  
 **this**.**sized** = inSize;  
 }  
  
 @Override  
 **public** String getSize() {  
 **return sized**;  
 }  
}

**public class** Building **implements** Sized {  
 String **sized**;  
  
 **public** Building(String inSize) {  
 **this**.**sized** = inSize;  
 }  
  
 @Override  
 **public** String getSize() {  
 **return sized**;  
 }  
}

**import** Geometry.Sized;  
**import** Geometry.SemesterSchedule;  
**import** Geometry.StudentRecords;  
**import** Geometry.Building;  
**import** Geometry.UnsizedException;  
**import** Geometry.Department;  
**public class** SizedClient {  
 **public static void** main(String args[]){  
 StudentRecords a=**new** StudentRecords(**"1"**);  
 Building b=**new** Building(**"2"**);  
 SemesterSchedule c=**new** SemesterSchedule(**"3"**);  
 Department d=**new** Department(**"4"**);  
  
 System.***out***.println(a.getSize());  
 System.***out***.println(b.getSize());  
 System.***out***.println(c.getSize());  
 System.***out***.println(d.getSize());  
 }  
}



Also declare a checked exception named UnsizedException.

**package** Geometry;  
  
**public class** UnsizedException **extends** Exception {  
 String **sized**;  
  
 **public** UnsizedException (String message){  
 System.***out***.println(**"Error message"**);  
 }  
}

Create a class named Measurements that contains just one static method named getSize that takes as an argument an object of any type (and returns a string type) and does the following:

If the object is of type String or StringBuffer, then returns a string of the form “16 characters”

If the object is of type Sized, and also of type

StudentRecords, then returns a string of the form “ ”

SemesterSchedule, then returns a string of the form “16 credits”

Department, then returns a string of the form “28 faculty members”

Building, then returns a string of the form “8560 square feet”

Otherwise, an UnsizedException is thrown.

**package** Geometry;  
  
**public class** Measurements {  
 **public** String Measurements(Object m) **throws** UnsizedException  
 {  
 Class name=m.getClass();  
 String string = **null**;  
 **if** (m **instanceof** StringBuffer) {  
 string= m+ **" characters"**;  
 }  
 **else if** (m **instanceof** Sized){  
 **if**(m **instanceof** StudentRecords){  
 string= **" "**;  
 }  
 **else if**(m **instanceof** SemesterSchedule){  
 string = **" "**;  
 }  
 **else if**(m **instanceof** Department){  
 string = m+ **" credits"**;  
 }  
 **else if**(m **instanceof** Sized && m **instanceof** Building){  
 string = m+ **" square feet"**;  
 }  
 }  
 **else** {  
 **throw new** UnsizedException(**"ERROR"**);  
 }  
 **return** string;  
 }  
}

**What to Turn In**

Submit a Word document on Blackboard containing all your answers.

Please include sufficient screenshots and introduction to demonstrate how your programs run. 40% deduction if screenshots and introduction are not provided.