**Name: Abdul Ghaffar Kalhoro**

**Reg: no: 194699**

**Class: BSCS-6C (2nd semester)**

**Lab-8**

**Object Oriented Programming (OOP)**

**Source Code for lab\_8 Task-1**

**Person.java**

**public** **class** Person **implements** PersonInterface, AnotherPersonInterface{

//data fields

**int** cashSaving;

**int** retirementFund;

String firstName;

String lastName;

//constructor for class person

Person(**int** cashS, **int** retirmntF, String FName, String Lname){

//assigning constructor variables to the class variables

cashSaving = cashS;

retirementFund = retirmntF;

firstName = FName;

lastName = Lname;

}

//overriding the method computeTotalWealth.

**public** **int** computeTotalWealth(){

**return** 30000;

}

//method getName.

**public** String getName(){

String Name = firstName + " "+ lastName;

**return** Name;

}

//overriding the method measureIntelligence

**public** **int** measureIntelligence(String x){

**return** 50;

}

}// end class person

**PersonInterface.java**

**public** **interface** PersonInterface {

//methods for overridden.

**int** computeTotalWealth();

String getName();

}

**AnotherPersonInterface.java**

**public** **interface** AnotherPersonInterface {

//method to be overridden.

**int** measureIntelligence(String x);

}

**Main.java**

**public** **class** Main {

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

// Create an object instance of Person class.

Person person1 = **new** Person(10000, 20000, "Quintin", "Tarantino");

// You can assign the object instance to

// PersonInterface type.

PersonInterface personinterface1 = person1;

// Display data from person1 and personinterface1.

// Observe that they refer to the same object instance.

System.***out***.println( "person1.getName() = " + person1.getName() + "," +

" person1.computeTotalWealth() = " + person1.computeTotalWealth() + "," +

" person1.measureIntelligence() = " +

person1.measureIntelligence(person1.getName()));

System.***out***.println( "personinterface1.getName() = " +

personinterface1.getName() + "," + " personinterface1.computeTotalWealth() = " +

personinterface1.computeTotalWealth());

// You can assign the object instance to

// AnotherPersonInterface type.

AnotherPersonInterface anotherpersoninterface1 = person1;

// Check of object instance that is referred by personinterface1 and

// anotherpersoninterface1 is the same object instance.

**boolean** b1 = (personinterface1 == anotherpersoninterface1);

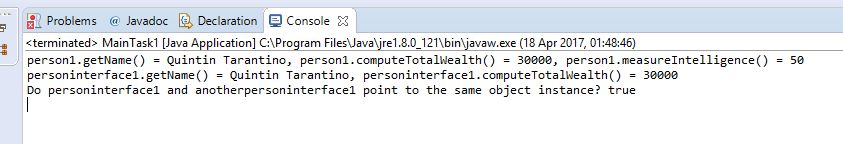
System.***out***.println("Do personinterface1 and anotherpersoninterface1"

+ " point to the same object instance? " + b1);

}

}

**OUTPUT**



**Bonus.** Compile error is expected on the following line of code. What could be causing it?

personinterface1.measureIntelligence(personinterface1.getName());

Ans: when the above line is added to the Main class there would be a syntax error because the personinterface1 is an object of the interface PersonInterface which does not have a method measureIntelligence(), hence it is an error.

**Source Code for lab\_8 Task-2**

**RelationInterface.java**

//creating an interface of name RelationInterface

**public** **interface** RelationInterface {

**boolean** isGreater(Object L1, Object L2);

**boolean** isLess(Object L1, Object L2);

**boolean** isEqual(Object L1, Object L2);

}//end interface.

**Line.java**

//a Line class which is implementing the interface of RelationInterface

**public** **class** Line **implements** RelationInterface

{

//data fields

**private** **double** x1;

**private** **double** x2;

**private** **double** y1;

**private** **double** y2;

//Creating an constructor for class Line

Line(**double** a, **double** b, **double** c, **double** d){

x1 = a;

x2 = b;

y1 = c;

y2 = d;

}

//get method for getting the length of line

**public** **double** getLength(){

**return** Math.*sqrt*((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

}

//the method isGreater to check that either of the line is greater.

**public** **boolean** isGreater(Object object1,Object object2){

Line line1 =(Line) object1; //down casting the object

Line line2 = (Line) object2;

**if**(line1.getLength() > line2.getLength()){

**return** **true**;

}

**else**{

**return** **false**;

}

}

//the method isLess to check that either of the line is less than the other.

**public** **boolean** isLess(Object object3,Object object4){

Line line1 =(Line) object3;

Line line2 = (Line) object4;

**if**(line1.getLength() < line2.getLength()){

**return** **true**;

}

**else**{

**return** **false**;

}

}

//the method isEqual to check that either of the lines are equal or not.

**public** **boolean** isEqual(Object object5,Object object6){

Line line1 = (Line) object5;

Line line2 = (Line) object6;

**if**(line1.getLength() == line2.getLength()){

**return** **true**;

}

**else**{

**return** **false**;

}

}

}//end class Line

**Main.java**

**public** **class** Main{

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

// Create two Line object instances.

Line line1 = **new** Line(1.0, 2.0, 3.0, 4.0);

Line line2 = **new** Line(2.0, 3.0, 7.0, 5.0);

**boolean** b1 = line1.isGreater(line1, line2);

System.***out***.println("line1 is greater than line2: " + b1);

**boolean** b2 = line1.isEqual(line1, line2);

System.***out***.println("line1 is equal with line2: " + b2);

// Note that the line3 is object instance of Line type.

// Because the Line type is also a type of RelationInterface,

// the line3 variable can be declared as RelationInterface type.

// This is a very very important concept you need to understand.

RelationInterface line3 = **new** Line(1.0, 5.0, 7.0, 9.0);

**boolean** b3 = line3.isEqual(line1, line3);

System.***out***.println("line1 is equal with line3: " + b3);

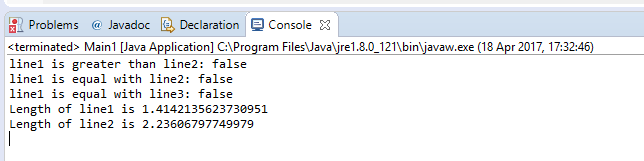
System.***out***.println("Length of line1 is " + line1.getLength());

System.***out***.println("Length of line2 is " + line2.getLength());

}

}//end class Main.

**OUTPUT**



**Bonus. a) What happens when the following line is added to the Main class? Identify the reason.**

System.out.println("Length of line3 is " + line3.getLength());

**Ans: when we add the above mentioned line in our main method it would be a syntax error because in given line it is tried to print the method getLength() which is not the part of our interface class, hence there would be an error. We can correct it by declaring the getLength() method in interface.**

**Source Code for lab\_8 Task-3**

**Product.java**

//The class Product which implements the Interface ProductInterface

**public** **class** Product **implements** ProductInterface

{

//data field

**private** **double** regularPrice;

//creating a constructor of class Product

**public** Product(**double** regPrice){

regularPrice=regPrice;

}

//creating a method for computeSalePrice

**public** **double** computeSalePrice(){

**return** regularPrice;

}

//set method for regularPrice

**public** **void** setRegularPrice(**double** regPrice){

regularPrice=regPrice;

}

//get method for regularPrice

**public** **double** getRegularPrice(){

**return** regularPrice;

}

} //end class Product

**Electronics.java**

// class Electronics extended from Product and implements ElectronicsInterface.

**public** **class** Electronics **extends** Product **implements** ElectronicsInterface {

//data field for class Electronics

**private** String manufacturer;

//Constructor for class Electronics with two arguments.

**public** Electronics(**double** regPrice,String manufacturer){

//invoking the constructor of super class.

**super**(regPrice);

}

//set method for Manufacturer

**public** **void** setManufacturer(String manufacturer){

**this**.manufacturer=manufacturer;

}

//get method for Manufacturer

**public** String getManufacturer(){

**return** **this**.manufacturer;

}

}//end class Electronics

**Book.java**

//creating class Book that is extended from class Product

**public** **class** Book **extends** Product{

//data fields

**private** String publisher;

**private** **int** yearPublished;

//constructor for class Book

**public** Book(**double** regularPrice,String publisher,**int** yearPublished){

//invoking suoer class constructor for class book.

**super**(regularPrice);

}

//method for sale price

**public** **double** computeSalePrice(){

**return** getRegularPrice()/2;//formula for computing sale price.

}

//set method for publisher

**public** **void** setPublisher(String publisher){

**this**.publisher=publisher;

}

//get method for publisher

**public** String getPublisher(){

**return** publisher;

}

//set method for year published

**public** **void** setYearPublished(**int** yPublished){

yearPublished=yPublished;

}

//get method for year published

**public** **int** getYearPublished(){

**return** **this**.yearPublished;

}

}//end class Book

**ElectronicsInterface.java**

//creating public interface with one method.

**public** **interface** ElectronicsInterface {

String getManufacturer();

}

**MP3Player.java**

/\*class MP3Player\*/

**public** **class** MP3Player **extends** Electronics{

//data fields

**private** String color;

//creating constructor for class MP3Player

**public** MP3Player(**double** regularPrice,String manufacturer,String color){

//invoking the super class constructor

**super**(regularPrice,manufacturer);

}

//method for computing the sales price

**public** **double** computeSalePrice(){

**return** getRegularPrice()-getRegularPrice()\*0.1;

}

//set method for color

**public** **void** SetColor(String color){

**this**.color=color;

}

//get method for color

**public** String getColor(){

**return** **this**.color;

}

}//end class MP3Player

**ProductInterface.java**

//product interface having two methods.

**public** **interface** ProductInterface {

**double** computeSalePrice();

**double** getRegularPrice();

}

**TV.java**

//class TV

**public** **class** TV **extends** Electronics{

//data fields

**private** **int** size;

//creating constructor for class TV with three parametres.

**public** TV(**double** regulrPrice,String manufacturer,**int** size){

//invoking super class constructor.

**super**(regulrPrice,manufacturer);

**this**.size = size;

}

//method for computing the sales price

**public** **double** computeSalePrice(){

**return** getRegularPrice()-getRegularPrice()\*0.2;

}

}//end class TV

**Main.java**

//creating a test class for all classes.

**public** **class** Main {

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

// Declare and create Product array of size 5

Product[] pa = **new** Product[5];

// Create object instances and assign them to

// the type of Product.

pa[0] = **new** TV( 1000, "Samsung", 30);

pa[1] = **new** TV( 2000, "Sony", 50);

pa[2] = **new** MP3Player( 250, "Apple", "blue");

pa[3] = **new** Book( 34, "Sun press", 1992);

pa[4] = **new** Book( 15, "Korea press", 1986);

// Compute total regular price and total

// sale price.

**double** totalRegularPrice = 0;

**double** totalSalePrice = 0;

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

// Call a method of the super class to get

// the regular price.

totalRegularPrice += pa[i].getRegularPrice();

// Since the sale price is computed differently

// depending on the product type, overriding (implementation)

// method of the object instance of the sub-class

// gets invoked. This is runtime polymorphic

// behavior.

totalSalePrice += pa[i].computeSalePrice();

System.***out***.println("Item number " + i +

": Type = " + pa[i].getClass().getName() +

", Regular price = " + pa[i].getRegularPrice() +

", Sale price = " + pa[i].computeSalePrice());

}

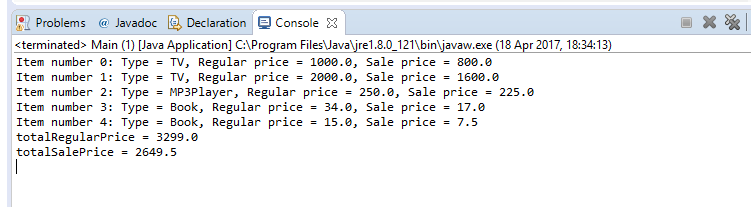
System.***out***.println("totalRegularPrice = " + totalRegularPrice);

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

}

}//end Main class.

**OUTPUT**

****