**Name: Abdul Ghaffar Kalhoro**

**Reg: no: 194699**

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

**Lab-7**

**Object Oriented Programming (OOP)**

**Source Code for lab\_7 Task-1**

**Main.java**

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

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

// Create Human object instance

// and assign it to Human type.

Human human1 = **new** Human( "Will Rodman");

human1.walk();

// Create Human object instance

// and assign it to LivingThing type.

LivingThing livingthing1 = human1;

livingthing1.walk();

// Create a Monkey object instance

// and assign it to LivingThing type.

LivingThing livingthing2 = **new** Monkey( "Caesar");

livingthing2.walk();

// Display data from human1 and livingthing1.

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

System.***out***.println( "human1.getName() = " + human1.getName());

System.***out***.println( "livingthing1.getName() = " + livingthing1.getName());

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

// y is the same object instance.

**boolean** b1 = ( human1 == livingthing1);

System.***out***.println( "Do human1 and livingthing1” + “point to the same object instance? " + b1);

}

}

**LivingThing.java**

**public** **abstract** **class** LivingThing {

// Create data field for LivingThing class

**private** String name;

//constructor

LivingThing(String z){

name = z;

}

//method breath

**public** **void** breath(){

System.***out***.println("LivingThing "+ getName() + " breaths.....");

}

//method eat

**public** **void** eat(){

System.***out***.println("LivingThing "+ getName() + " also eats.....");

}

//abstract function walk

**public** **abstract** **void** walk();

//get method for name

**public** String getName(){

**return** name;

}

//Set method for name

**public** **void** setName(String Name){

name = Name;

}

}

**Human.java**

**public class** Human **extends** LivingThing{

//constructor for class Human

Human(String a){

//invoking the constructor of super class LivingThing

**super**(a);

}

//overriding the method walk() of class livingThing

**public** **void** walk(){

System.***out***.println("Human "+ getName() + " walks.....");

}

}

**Monkey.java**

**public class** Monkey **extends** LivingThing{

//constructor for class Monkey

Monkey(String a){

//invoking the constructor of super class LivingThing

**super**(a);

}

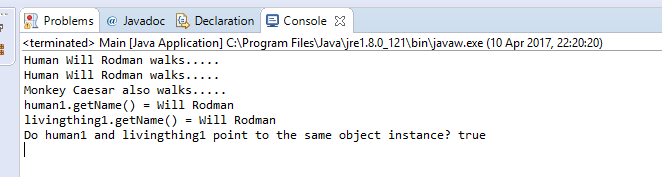
//overriding the method walk() of class livingThing

**public** **void** walk(){

System.***out***.println("Monkey "+ getName() + " also walks.....");

}

}

**OUTPUT**

**Q: What happens when you create a LivingThing object in the Main class? For example using the statement,**

**LivingThing z = new LivingThing();**

**Ans: When we try to create an object of the class LivingThing it would show a syntax error because the class livingThing is created as a abstract which does not require an object.**

**Source Code for lab\_7 Task-2**

**Product.java**

**public** **abstract** **class** Product {

//data field of Product class

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

//creating constructor

Product(**double** price){

regularPrice = price;

}

//declarng abstract method computerSalePrice

**abstract** **double** computeSalePrice();

//get method for Regular Price

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

**return** regularPrice;

}

//get method for Regular Price

**public** **void** setRegularPrice(**double** R\_Price){

regularPrice = R\_Price;

}

} //end class Product

**Electronics.java**

**public** **abstract** **class** Electronics **extends** Product{

//data field

**private** String manufacturer;

//constructor for class Electronics

Electronics(**double** number, String Manufacturer){

//invoking constructor of super class.

**super**(number);

manufacturer = Manufacturer;

}

//get method for manufacturer

**public** String getManufacturer(){

**return** manufacturer;

}

//set method for manufacturer.

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

manufacturer = variable;

}

} //end class Electronics.

**MP3Player.java**

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

//data field.

**private** String color;

//constructor for classMP3Player

MP3Player(**double** number,String ManuFactrr,String theme){

//invoking constructor of super class Electronics

**super**(number,ManuFactrr);

color = theme;

}

//Overriding the abstract method computerSalePrice of super class.

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

**return** getRegularPrice()-getRegularPrice()\*10/100;

}

//get method for color.

**public** String getColor(){

**return** color;

}

//set method for color

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

**this**.color = color;

}

} //end class MP3Player.

**TV.java**

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

//data fields

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

//constructor.

TV(**double** number,String ManuFactrr,**int** size){

//invoking the constructor of super class

**super**(number,ManuFactrr);

**this**.size = size;

}

//Overriding the abstract method computerSalePrice of super class.

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

**return** getRegularPrice()-getRegularPrice()\*20/100;

}

} //end class TV

**Book.java**

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

//data fields

**private** String publisher;

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

//constructor

Book(**double** number, String Publisher, **int** Year\_Of\_Published){

//invoking the constructor of super class Product

**super**(number);

//assigning the values to the variables of constructor

**this**.publisher = Publisher;

**this**.yearPublished = Year\_Of\_Published;

}

//Overriding the abstract method computerSalePrice of super class.

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

**return** getRegularPrice()/2;

}

//get method for publisher

**public** String getPublisher(){

**return** publisher;

}

//set method for publisher

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

publisher = variable;

}

//get method for yearPublished

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

**return** yearPublished;

}

//set method for yearPublished

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

yearPublished = variable;

}

} //end class Book.

**Main.java**

**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());

}

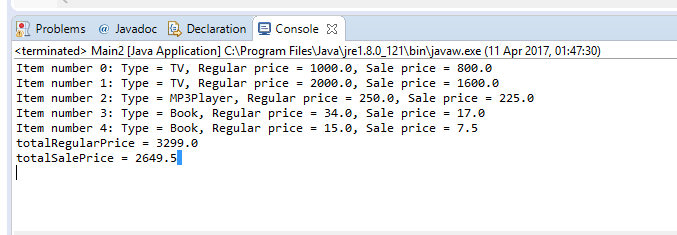
System.***out***.println("totalRegularPrice = " + totalRegularPrice);

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

}

} //end class Main.

**OUTPUT**

****