**Name: Hamad Nasir**

**Roll Number: 120312**

**Section: BSCS-6C**

**Lab No: 7**

**Task 1:**

**Source Code:**

package lab.pkg7;

//Living thing is main class

public abstract class LivingThing {

//private instant variable

private String name;

//1 argument constructor

public LivingThing(String name){

this.name=name;

}

//breath method with no return type

public void breath(){

}

//eat method with no return type

public void eat(){

}

//abstract method with no return type

public abstract void walk();

//set name method

public void setName(String name){

this.name=name;

}

//get name method

public String getName(){

return this.name;

}

}

package lab.pkg7;

//Human is subclass of LivingThing

public class Human extends LivingThing{

//public Human constructor

public Human(String name){

super(name);

}

//as walk is abstract method so must be overriden

@Override

public void walk(){

System.out.println("Human "+getName() + " walk...");

}

}

package lab.pkg7;

//Monkey class is subclass of LivingThing

public class Monkey extends LivingThing {

//publib Monkey constructor

public Monkey(String name){

super(name);

}

//as walk is abstract method so must be overriden

@Override

public void walk(){

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

}

}

**Test Class:**

package lab.pkg7;

public class Lab7 {

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.

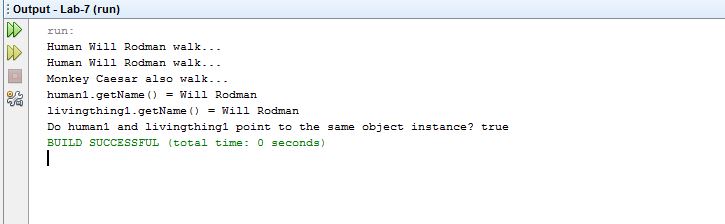
boolean b1 = ( human1 == livingthing1);

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

}

}

**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 a LivingThing object is created in the Main class the compiler will give a syntax error. Because it is understood that LivingThing class is an abstract class and object of an abstract can’t be made directly in the main class.

**Task 2:**

**Source Code:**

package lab7\_2

//Product is abstract main class

public abstract class Product {

//private instant variable

private double regularPrice;

//1 argument constructor

public Product(double regularPrice){

this.regularPrice=regularPrice;

}

//abstract compute sale price method

public abstract double computeSalePrice();

//set regular price method

public void setRegularPrice(double regularPrice){

this.regularPrice=regularPrice;

}

//get regular price method

public double getRegularPrice(){

return this.regularPrice;

}

}

package lab7\_2;

//Book is subclass of product

public class Book extends Product{

//private instant variables

private String publisher;

private int yearPublished;

//three aargument constructor

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

super(regularPrice);

}

//compute sale price method

public double computeSalePrice(){

return getRegularPrice()/2;

}

//set publisher method

public void setPublisfer(String publisher){

this.publisher=publisher;

}

//get publisher method

public String getPublisher(){

return this.publisher;

}

//set year published method

public void setYearPublished(int yeatPublished){

this.yearPublished=yearPublished;

}

//get year published method

public int getYearPublished(){

return this.yearPublished;

}

}

package lab7\_2;

// Electronics is abstract subclass of product

public abstract class Electronics extends Product {

//private instant variable

private String manufacturer;

//2 argument constructor

public Electronics(double regularPrice,String manufacturer){

super(regularPrice);

}

//set manufacturer method

public void setManufacturer(String manufacturer){

this.manufacturer=manufacturer;

}

//get manufacturer method

public String getManufacturer(){

return this.manufacturer;

}

}

package lab7\_2;

//MP3 is subclass of Electronics

public class MP3Player extends Electronics{

//private instant variable

private String color;

//3 argument constructor

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

super(regularPrice,manufacturer);

}

//compute sale price method

public double computeSalePrice(){

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

}

//set color method

public void SetColor(String color){

this.color=color;

}

//get color method

public String getColor(){

return this.color;

}

}

package lab7\_2;

//Electronics is subclass of Electronics

public class TV extends Electronics{

//private instant variable

private int size;

//3 argument constructor

public TV(double regularPrice,String manufacturer,int size){

super(regularPrice,manufacturer);

this.size = size;

}

//compute sale price method

public double computeSalePrice(){

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

}

}

**Test Class:**

package lab7\_2;

public class Lab7\_2 {

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);

}

}

**Output:**

