**Muhammad Rizwan Khalid**

**180459**

**BSCS 6A**

**Lab no: 07**

**Warm-Up example:**

The simple example is pet class. In this class we make a super class pet and derived classes of cat and dog and horse. We can add the abstract method “sound of pets”. We can make the pet class abstract.

**Task one:**

package polymorphism;

public class Polymorphism {

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);

}

}

abstract class LivingThing{

private String name;

public LivingThing(String name) {

this.name = name;

}

public void setName(String name){

this.name = name;

}

public String getName(){

return this.name;

}

public void breath(){

System.out.println("The Living thing is breathing");

}

public void eat(){

System.out.println("The living thing is eating");

}

public abstract void walk();

}

class Human extends LivingThing{

public Human(String name) {

super(name);

}

@Override

public void walk(){

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

}

}

class Monkey extends LivingThing{

public Monkey(String name) {

super(name);

}

@Override

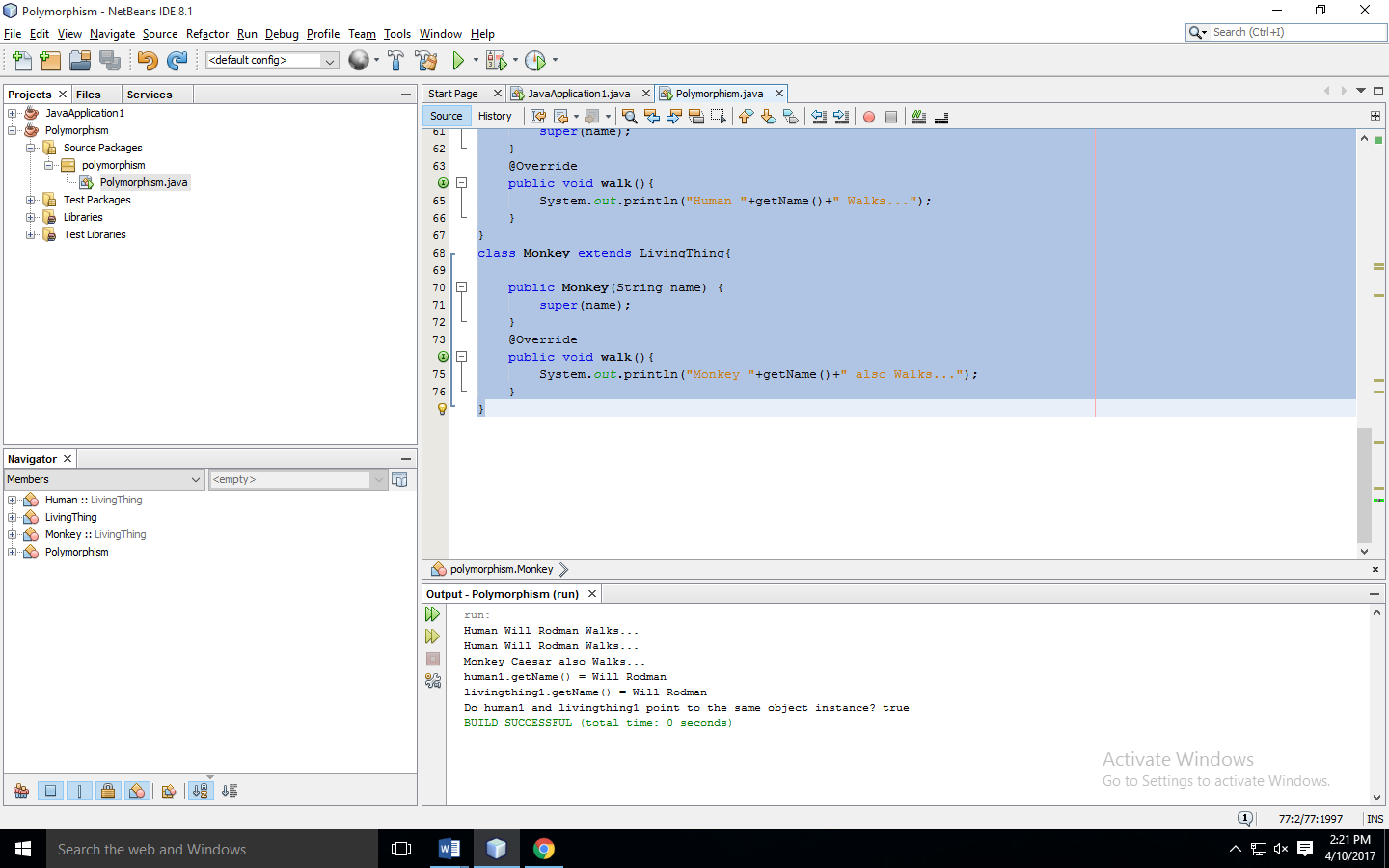
public void walk(){

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

}

}

**Output:**



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

LivingThing z = new LivingThing();

**Answer:** The compiler gives the error because LivingThing is an abstract class. The object of abstract class cannot be istantiated.

**Task two:**

package polymorphism;

public class Polymorphism {

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);

}

}

abstract class Product{

private double regularPrice;

public Product(double regularPrice) {

this.regularPrice = regularPrice;

}

public void setRegularPrice(double regularPrice){

this.regularPrice = regularPrice;

}

public double getRegularPrice(){

return this.regularPrice;

}

public abstract double computeSalePrice();

}

class Book extends Product{

private String publisher;

private int yearPublished;

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

super(regularPrice);

this.publisher = publisher;

this.yearPublished = yearPublished;

}

public void setPublisher(String publisher){

this.publisher = publisher;

}

public void setYearPublished(int yearPublished){

this.yearPublished = yearPublished;

}

public String getPublisher(){

return this.publisher;

}

public int yearPublished(){

return this.yearPublished;

}

@Override

public double computeSalePrice(){

return 0.5 \* getRegularPrice();

}

}

abstract class Electronics extends Product{

private String manufacturer;

public Electronics(double regularPrice, String manufacturer) {

super(regularPrice);

this.manufacturer = manufacturer;

}

public void setManufacturer(String manufacturer){

this.manufacturer = manufacturer;

}

public String getManufacturer(){

return this.manufacturer;

}

}

class MP3Player extends Electronics {

private String color;

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

super(regularPrice,manufacturer);

this.color = color;

}

public void setColor(String color){

this.color = color;

}

public String getColor(){

return this.color;

}

@Override

public double computeSalePrice(){

return 0.9 \* getRegularPrice();

}

}

class TV extends Electronics{

private int size;

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

super(regularPrice,manufacturer);

this.size = size;

}

@Override

public double computeSalePrice(){

return 0.8 \* getRegularPrice();

}

}

**Output:**

