# Assignment 10

# DATE – 03/11/2022

1. **Create an interface named Shape with a field pie (=3.14). Create two subclasses of it named Circle and Rectangle create object of the two classes and calculate their area.**

# Solution

interface Shape {

public double pi = Math.PI; public double getArea();

}

class Rectangle implements Shape { public double length;

public double breadth;

public Rectangle(double length, double breadth) { this.length = length;

this.breadth = breadth;

}

@Override

public double getArea() { return length \* breadth;

}

}

class Circle implements Shape { public double radius;

public Circle(double radius) { this.radius = radius;

}

@Override

public double getArea() {

return pi \* Math.pow(radius, 2);

}

}

public class ques1 {

public static void main(String[] args) {

Rectangle rectangle = new Rectangle(7.5, 8.7); Circle circle = new Circle(4.5);

System.out.println("The Area of Rectangle: " + rectangle.getArea()); System.out.println("The Area of Circle: " + circle.getArea());

}

}

# Output

1. **Create a class which contains an inner class. Show that inner class can use member of outer class directly, but Outer class can use member of Inner class only through its object. Check the name of class file, you created.**

# Solution

class Outer { int x; Outer(int x) {

this.x = x; System.out.println("Outer Class");

}

void outerShow() {

// System.out.println("The value of Y: " + y); System.out.println("The value of X: " + x);

}

class Inner { int y; Inner(int y) {

this.y = y; System.out.println("Inner Class");

}

void innerShow() {

System.out.println("The value of X: " + x); System.out.println("The value of Y: " + y);

}

}

}

public class ques2 {

public static void main(String[] args) { Outer out = new Outer(10);

Outer.Inner in = new Outer(20).new Inner(25); out.outerShow();

in.innerShow();

}

}

# Output

1. **Create two interfaces, each with two methods. Inherit a new interface from the two, adding a new method. Create a class by implementing the new interface and also inheriting from a concrete class. In main() method, create an object of derived class and call the methods [do all without package statement].**

# Solution

interface A { public void B(); public void C();

}

interface P { public void Q(); public void R();

}

interface X extends A, P {

public void Y();

}

class Z {

public void W() { System.out.println("Class Function W");

}

}

class Inherit extends Z implements X { @Override

public void B() { System.out.println("Interface Function B");

}

@Override public void C() {

System.out.println("Interface Function C");

}

@Override public void Q() {

System.out.println("Interface Function Q");

}

@Override public void R() {

System.out.println("Interface Function R");

}

@Override public void Y() {

System.out.println("Interface Function Y");

}

}

public class ques3 {

public static void main(String[] args) { Inherit in = new Inherit();

in.B(); in.C(); in.Q();

in.R(); in.Y(); in.W();

}

}

# Output

1. **Write a program to demonstrate anonymous inner class (using super class and interface).**

# Solution

interface A { public int x = 21;

public void getValue();

}

public class ques4 {

public static void main(String[] args) { A a = new A() {

@Override

public void getValue() { System.out.println("The Value of X: " + x);

}

};

a.getValue();

}

}

# Output

1. **Show that an ordinary block is executed when an object is created and also the order of execution of these blocks (for multiple blocks/ inherited blocks).**

# Solution

class X {

{

System.out.println("X 1st Block");

}

X() {

System.out.println("Class X construtor");

}

{

System.out.println("X 2nd Block");

}

}

class A extends X {

{

System.out.println("A 1st Block");

}

A() {

super();

System.out.println("Class A construtor");

}

{

System.out.println("A 2nd Block");

}

}

public class ques5 {

public static void main(String[] args) { new A();

}

}

# Output

1. **Show that a static block is executed at the time of class loading and also the order of execution of these blocks (for multiple blocks/ inherited blocks).**

**Solution** class X { static {

System.out.println("X 1st Block");

}

X() {

System.out.println("Class X construtor");

}

static {

System.out.println("X 2nd Block");

}

}

class A extends X { static {

System.out.println("A 1st Block");

}

A() {

super();

System.out.println("Class A construtor");

}

static {

System.out.println("A 2nd Block");

}

}

public class ques6 {

public static void main(String[] args) { new A();

}

}

# Output

1. **Write a program to show the difference between ordinary block and static block.**

# Solution

class X {

{

System.out.print("X 1st Block ");

}

X() {

System.out.println("Class X construtor ");

}

{

System.out.print("X 2nd Block ");

}

}

class A extends X {

{

System.out.print("A 1st Block ");

}

A() {

super();

System.out.print("Class A construtor ");

}

{

System.out.print("A 2nd Block ");

}

}

class Y { static {

System.out.print("Y 1st Block ");

}

Y() {

System.out.print("Class Y construtor ");

}

static {

System.out.print("Y 2nd Block ");

}

}

class B extends Y { static {

System.out.print("B 1st Block ");

}

B() {

super();

System.out.println("Class B construtor ");

}

static {

System.out.print("B 2nd Block ");

}

}

public class ques7 {

public static void main(String[] args) { System.out.println("Ordinary Blocks"); new A();

System.out.println("\nStatic Blocks"); new B();

}

}

# Output

1. **Create a class with variable(s) and method(s) (all will be default accessed) under package pOne. Now create a class under package pTwo, which is a subclass of the firstly created class. In the method here (i.e. class of pTwo) call variable(s) and method(s) of previous class (i.e. class of pOne). If errors come, rectify them. Now from Main (under working directory) check access to second class’s members.**

**Solution** package pOne; public class A {

public int x; public A(int x) {

this.x = x;

}

public void showX() { System.out.println("X value: " + x);

}

}

package pTwo; import pOne.\*;

public class B extends A { public int y;

public B(int x, int y) { super(x);

this.y = y;

}

public void showY() { super.showX(); System.out.println("Y value: " + y);

}

}

import pOne.A; import pTwo.B; public class ques8 {

public static void main(String[] args) { A a = new A(5);

B b = new B(10, 15);

a.showX();

b.showY();

}

}

# Output

1. **Create an interface containing three methods, in a package ‘pkgOne’. Implement the interface from a class under package pkgTwo. From the main(), under the working directory, create an object of the class and call methods of the interface.**

# Solution

package pkgOne; public interface X {

public void A(); public void B(); public void C();

}

package pkgTwo; import pkgOne.\*;

public class Y implements X { @Override

public void A() { System.out.println("Function A");

}

@Override public void B() {

System.out.println("Function B");

}

@Override public void C() {

System.out.println("Function C");

}

}

import pkgTwo.Y; public class ques9 {

public static void main(String[] args) { Y y = new Y();

y.A();

y.B();

y.C();

}

}

# Output