ASSIGNMENT 2.

1. **Create singleton class and make sure singleton class cannot be inherits.**

ANS:

package practical;

class Abc{

static Abc obj1 = new Abc(); //create static object of singleton class

private Abc() { // create private constructor

}

public static Abc getInstance() {

return obj1;

}

}

public class Singleton extends Abc{ // we cant extends Singlton class Abc here

public static void main(String[] args) {

// TODO Auto-generated method stub

Abc obj = Abc.getInstance();

Abc obj3 = Abc.getInstance();}}

2.

**4. Abstraction**

1. **If any class contain any abstract method then that class must be represent as Abstract class.**

ANS:

package Abstract

abstract class Dog{

public abstract void eat();

public abstract void run();

}

public class AbstractClass {

public static void main(String[] args) {

// TODO Auto-generated method stub

}

}

**Output: if we write abstract method and not declare class as a abstract then compiler expect method body .**

1. **Abstract class cant be instantiate.**

ANS:

package Abstract

abstract class Dog{

public abstract void eat();

public abstract void run();

}

public class AbstractClass{

public void eat(){

System.out.println(“eat”);

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Dog d = new Dog(); // java does not support to create object of abstract class.

Dog d = new AbstractClass(); //reference variable of abstract class and object of subclass. }

}

**OUTPUT: eat**

1. **When we extend an abstract class, we must override all the abstract method in subclass or declare it as a abstract .**

ANS: package Abstraction;

abstract class Dog{

public abstract void eat();

public abstract void run();

}

public class AbstractClass extends Dog {

public static void main(String[] args) {

// TODO Auto-generated method stub

}

@Override

public void eat() {

// TODO Auto-generated method stub

}

@Override

public void run() {

// TODO Auto-generated method stub

}}

**OUTPUT** : yes we have to override it or compile automatically override it.

**iV. Abstract class can not be private.**

Ans:

package Abstraction;

private abstract class Dog{ **// here I Have provide private //access specifier to check the error**

public abstract void eat();

public abstract void run();

}

public class AbstractClass {

public static void main(String[] args) {

// TODO Auto-generated method stub

}

}

**Output :** **Illegal modifier for the class Dog; only public, abstract & final are permitted**

**V. Abstract class can not be Final**

**Ans:**

package Abstraction;

final abstract class Dog{ // here I Have provide final keyword to check the error

public abstract void eat();

public abstract void run();

}

public class AbstractClass {

public static void main(String[] args) {

// TODO Auto-generated method stub

}

}

**Output : Illegal modifier for the class Dog; only public, abstract are permitted**

1. **You can declare a class abstract without having abstract method in that class.**

**ANS:**

package Abstraction;

abstract class Dog{ // here I have created abstract class

public void eat() { // but the method is non abstract

System.out.println("eat");

}

public void run() { //here also method is non abstract

}

}

public class AbstractClass extends Dog {

public static void main(String[] args) {

// TODO Auto-generated method stub

Dog d = new AbstractClass();

d.eat();

d.run();

}

}

OUTPUT: eat

1. **Write a classes Line, Rectangle, cube and make Shape as a base class. Add abstract draw() in subclass shape & draw all shape.**

Ans:

package Abstraction;

public abstract class Shape { // abstract class shape

public abstract void shape(); // abstract method

public static void main(String[] args) {

// TODO Auto-generated method stub

Shape s = new cube(); // we cant able to create object of abstract class so I have //created object of subclass and reference variable of abstract class

s.shape(); // method calling using . operator

Shape s1 = new line();

s1.shape();

Shape s3 = new Rectangle();

s3.shape();

}

}

Class 2:

package Abstraction;

public class Rectangle extends Shape{

@Override

public void shape() { **// override abstract class abstract method** and provide //implementation.

// TODO Auto-generated method stub

int W=10;

int l= 5;

int area= W\*l;

System.out.println("Area of Rectangle= " +area);

}

}

Class 3

package Abstraction;

public class line extends Shape{

@Override

public void shape() {

// TODO Auto-generated method stub

System.out.println("Line");

}

}

Class 4:

package Abstraction;

public class cube extends Shape {

@Override

public void shape() {

// TODO Auto-generated method stub

int size =10;

int volume = size\*size\*size;

System.out.println("volume of cube= "+ volume);

}

}

Output:

volume of cube= 1000

Line

Area of Rectangle= 50

\_ \_ \_ \_ \_ \_ \_ \_ \_\_ \_ \_ \_ \_ \_\_ \_ \_ \_\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_\_ \_ \_ \_ \_\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_\_ \_ \_ \_ \_ \_ \_

6.**Create abstract class as “Persistence” and create two subclasses “DatabasePersistence”** **And “FilePersistence” . the base class contain abstract persist() method which will be overridden by its subclass. Write a program to get the persistence object at runtime & evoke persist() method .**

Ans:

**Class 1:**

package Practice;

public abstract class Persistence {

public abstract void persist();

public static void main(String[] arg) {

Persistence p = new DatabasePersistence(); // here runtime decide which class method needs to run.

p.persist();

Persistence p1 = new FilePersistence(); // here runtime decide which class method needs to run.

p1.persist();

}

}

**CLASS 2: FilePersistence subclass**

package Practice;

public class FilePersistence extends Persistence {

@Override

public void persist() {

// TODO Auto-generated method stub

System.out.println("I am persist method of File");

}}

**CLASS 3: DatabasePersistence subclass**

package Practice;

public class DatabasePersistence extends Persistence{

@Override

public void persist() {

// TODO Auto-generated method stub

System.out.println("I am persist method of Database");

}

}

**OUTPUT:**

**I am persist method of Database**

**I am persist method of File**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_