**Assigment 2:**

1.Singleton class

**package** com.cap;

**public** **class** App {

**private** **static** App *app*;

**private** App() {

}

**public** **static** App getInstance() {

**if**(**null**==*app*) {

*app*=**new** App();

}

**return** *app*;

}

}

**package** com.cap;

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

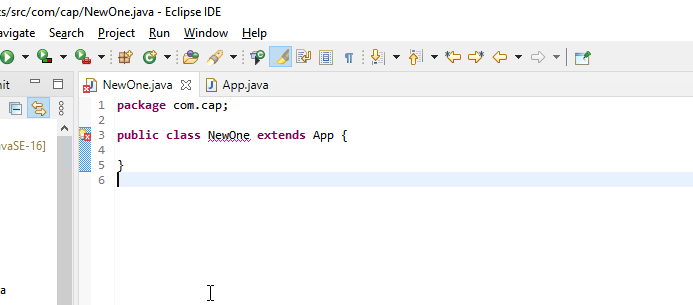
**public** **static** **void** main(String[] args) {

App app=App.*getInstance*();

App app1=App.*getInstance*();

}

}

****

2.

Main class:

**package** com.cap;

**public** **class** Main1 {

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

Manager m1 = **new** Manager("Nikhil",20000,500);

System.***out***.println("Salary of Manager= "+m1.getSalary());

Labour l1 = **new** Labour("Rathod",10000,500);

System.***out***.println("Salary of Labour= "+l1.getSalary());

}

}

Employee Class:

**package** com.cap;

**public** **class** Employee {

String name;

**int** salary;

Employee(){

name= **null**;

salary=0;

}

Employee(String name,**int** salary){

**this**.name=name;

**this**.salary=salary;

}

**int** getSalary() {

**return** salary;

}

}

Manager Class:

**package** com.cap;

**public** **class** Manager **extends** Employee {

**int** incentives;

Manager(){

**super**();

incentives=0;

}

Manager(String n,**int** sal,**int** h){

**super**(n,sal);

incentives=h;

}

**int** getSalary() {

**return** (**super**.getSalary()+incentives);

}

}

Labour class:

**package** com.cap;

**public** **class** Labour **extends** Employee {

**int** overTime;

Labour(){

**super**();

overTime=0;

}

Labour(String n,**int** sal,**int** h){

**super**(n,sal);

overTime=h;

}

**int** getSalary() {

**return** (**super**.getSalary()+overTime);

}

}

3. Savings and current account

Main Class:

**package** com.cap;

**public** **class** Main2 {

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

Saving s = **new** Saving("Nikhil",2000,2000);

System.***out***.println("Salary of Manager= "+s.balance());

Current c = **new** Current("Rathod",0,0);

System.***out***.println("Salary of Labour= "+c.balance());

}

}

Account class:

**package** com.cap;

**public** **class** Account {

String name;

**int** cash;

Account(){

name= **null**;

cash=0;

}

Account(String name,**int** cash){

**this**.name=name;

**this**.cash=cash;

}

**int** balance() {

**return** cash;

}

}

Saving Class:

**package** com.cap;

**public** **class** Saving **extends** Account{

**int** deposit;

Saving(){

**super**();

deposit=0;

}

Saving(String n,**int** bal,**int** h){

**super**(n,bal);

deposit=h;

}

**int** balance() {

**return** (**super**.balance()+deposit);

}

}

Current Class:

**package** com.cap;

**public** **class** Current **extends** Account {

**int** cash;

Current(){

**super**();

cash=0;

}

Current(String n,**int** bal,**int** h){

**super**(n,bal);

cash=h;

}

**int** getSalary() {

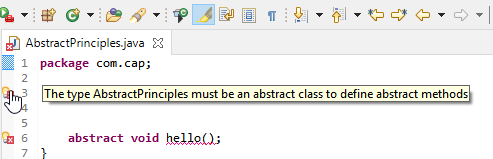
**return** (**super**.balance()+cash);

}

}

4: Abstract Class Principles

1. If a class has abstract methods then its class should be a Abstract class
2. **package** com.cap;
3. **class** AbstractPrinciples {
4. **abstract** **void** hello();
5. }

****

2. Abstract class cannot be instantiated

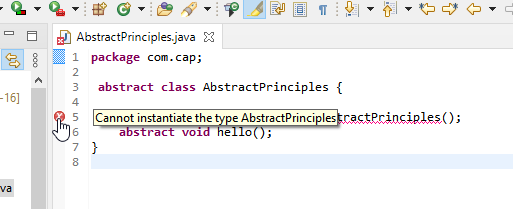
**package** com.cap;

**abstract** **class** AbstractPrinciples {

AbstractPrinciples obj=**new** AbstractPrinciples();

**abstract** **void** hello();

}

****

3: When we extend abstract class we must override all the methods of abstract class.

**package** com.cap;

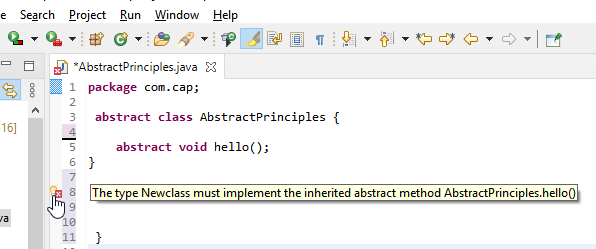
**abstract** **class** AbstractPrinciples {

**abstract** **void** hello();

}

**class** Newclass **extends** AbstractPrinciples{

}

****

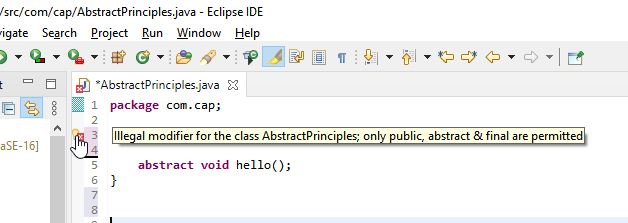
4. Abstract class cannot be private

**package** com.cap;

**private** **abstract** **class** AbstractPrinciples {

**abstract** **void** hello();

}

****

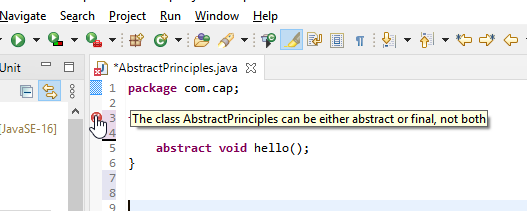
5. Abstract class cannot be final

**package** com.cap;

**final** **abstract** **class** AbstractPrinciples {

**abstract** **void** hello();

}

****

6.You can have an Abstract class without any abstract method in it.

**package** com.cap;

**abstract** **class** AbstractPrinciples {

**public** **void** hello() {

System.***out***.println("Hello World");

}

}

**5.**

//creating a Shape abstract class

**abstract** **class** Shape {

**abstract** **void** draw();

}

//Drawing line shape

**class** line **extends** Shape{

**void** draw() {

System.***out***.println("drawing line...");

}

}

//Drawing cube shape

**class** cube **extends** Shape{

**void** draw()

{

System.***out***.println("drawing cube...");

}

}

//Drawing rectangle shape

**class** rectangle **extends** Shape{

**void** draw()

{

System.***out***.println("drawing rectangle...");

}

}

//calling main method

**class** hello{

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

{

Shape a= **new** line();

a.draw();

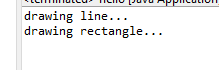
Shape b= **new** rectangle();

b.draw();

}

}

**Output:**

****

**6.**

**package** file;

**abstract** **class** persistance{

**public** **abstract** **void** persist();

}

**class** filePersistance **extends** persistance {

@Override

**public** **void** persist() {

System.***out***.println("data is being saved in in file");

}

}

**class** databasePersistance **extends** persistance {

@Override

**public** **void** persist() {

System.***out***.println("data is not being saved in in databse");

}

}

**public** **class** Persistence{

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

filePersistance p =**new** filePersistance();

p.persist();

databasePersistance p1=**new** databasePersistance();

p1.persist();

}

}

**Output:**

**o6.PNG**