1. **Create a class which is not going to inherit**

**package** newjava;

**final** **class** Singleton{

**int** check;

}

**class** tryingtoinherit **extends** Singleton

{

check=10;

}

**public** **class** Javaassign2 {

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

tryingtoinherit obj = **new** tryingtoinherit();

System.***out***.println(check);

}

}

1. Use of method overriding make 3 class Employee, Manager, Labour. Manager and Labour are the sub class of Employee. Manager have incentives and Labour have overtime.find the Salary of all employee.

**class** Employee{

**int** incentive(**int** Current\_salary)

{

**return** Current\_salary +30;

}

**int** overtime()

{

**return** 40;

}

}

**class** Manager **extends** Employee

{

**int** incentive(**int** Current\_salary)

{

System.***out***.println("Current Salary is--> "+ Current\_salary);

**return** Current\_salary+1000;

}

}

**class** Labour **extends** Employee

{

**int** overtime(**int** Current\_salary)

{

System.***out***.println("Current Salary is--> "+ Current\_salary);

**return** Current\_salary+1500;

}

}

**public** **class** Javaassign2 {

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

Manager objmanager = **new** Manager();

Labour objLabour = **new** Labour();

System.***out***.println("After Adding incentive, total salary is "+(objmanager.incentive(20000)));

System.***out***.println("After Adding Overtime, total salary is "+objLabour.overtime(15000));

}

}

4. ABSTRACT CLASS…

/\*part 1

class check\_abstract{ //The type check\_abstract must be an abstract class to define abstract methods

abstract int jack();

}

part 2

abstract class check\_abstract{

void checkng()

{

System.out.println("checking...");

}}

public class Javaassign2 {

public static void main(String[] args) {

check\_abstract obj = new check\_abstract(); //Cannot instantiate the type check\_abstract

}}

part 3 and 6.

abstract class check\_abstract{

void check()

{

System.out.println("checking !! you hav to override me first");

}

}

class second extends check\_abstract

{

void check() // overriding method here

{

System.out.println("overrided");

}}

public class Javaassign2 {

public static void main(String[] args) {

second obj = new second();

obj.check();

}}

part 4

final abstract class check\_abstract{ //The class check\_abstract can be either abstract or final, not both

void checkng()

{

System.out.println("checking...");

}}

part 5....

private abstract class check\_abstract{ //llegal modifier for the class check\_abstract; only public, abstract & final are permitted

void checkng()

{

System.out.println("checking...");

}}

\*/

5. draw shaped using abstract method….

**abstract** **class** shape{

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

}

**class** rect{

// for accessing abstract method we should override it in child class.......

**void** draw()

{

System.***out***.println("this is rectangle");

}

}

**class** cube{

**void** draw()

{

System.***out***.println("this is a cube");

}

}

**class** line{

**void** draw()

{

System.***out***.println("this is line");

}

}

**public** **class** Javaassign2 {

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

rect obj1 = **new** rect();

line obj2 = **new** line();

cube obj3 = **new** cube();

obj1.draw();

obj2.draw();

obj3.draw();

}}

6. We can not create object of class persistence class because it is initialize as abstract keyword. Hence we can override it in it’s child class and then create object of those child class .

7. Make a dessert name restro and it has two role, owner and customer. I role is owner then it will say ask to enter name of dessert. If role is customer then it will take our order and show the bill as well…

**import** java.util.\*;

**abstract** **class** Desert{

**abstract** **int** getcost();

}

**class** candy{

**int** getcost()

{

**return** 5\*60;

}

}

**class** cookies{

**int** getcost()

{

**return** 6\*70;

}

}

**class** icecream

{

**int** getcost()

{

**return** 100;

}

}

**public** **class** Javaassign2 {

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

String role,order,quit;

candy obj1=**new** candy();

cookies obj2=**new** cookies();

icecream obj3=**new** icecream();

String[] arr=**new** String[3];

Scanner sc= **new** Scanner(System.***in***);

System.***out***.println("=========Welcome to our Dessert Restro=======\n");

System.***out***.println("For quiting our online service please enter 'q' .........\n");

**while**(**true**)

{

System.***out***.println("Who you are--->>>>");

role = sc.next().toLowerCase();

**if**(role.equals("owner"))

{

System.***out***.println("Enter the Desert Name.....\n");

**for**(**int** i=0;i<arr.length;i++)

{

arr[i]=sc.next();

}

System.***out***.println("Your Dessert names are-->>>\n");

**for**(**int** i=0;i<arr.length;i++)

{

System.***out***.println(arr[i]);

}

}

**else** **if**(role.equals("customer"))

{

System.***out***.println("\nWhat would you likw to have sir->");

order = sc.next().toLowerCase();

**if**(order.equals("candy"))

{

System.***out***.println("Here this is your bill......\n");

System.***out***.println(obj1.getcost());

}

**else** **if**(order.equals("cookies"))

{

System.***out***.println("Here this is your bill......\n");

System.***out***.println(obj2.getcost());

}

**else** **if**(order.equals("icecream"))

{

System.***out***.println("Here this is your bill......\n");

System.***out***.println(obj3.getcost());

}

**else**

{

System.***out***.println("wrong choice");

}

}

**else** **if**(role.equals('q'))

{

**break**;

}

}

System.***out***.println("=============Thanks for choosing our service=============");

}}