[**Q1**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2130)**. Write a singleton class. Confirm that singleton class cannot be inherited.**

**Singleton Class:-**

The singleton design pattern is used to restrict the instantiation of a class and ensures that only one instance of the class exists in the JVM. In other words, a singleton class is a class that can have only one object (an instance of the class) at a time per JVM instance.

**import** java.util.Objects;

**class** singletoninheritancecheck{

**public** **static** **void** Check(Singleton A, Singleton B) {

**if**(Objects.*equals*(A , B)){

System.***out***.println("Authenticated Singleton Inheritance");}

**else** {

System.***out***.println("Not Singleton");

}

}

}

**class** Singleton

{

**private** **static** Singleton *instance*;

**public** String str;

**private** Singleton()

{

str = "Yes it is a Singleton Class";

}

**public** **static** Singleton getInstance()

{

**if** (*instance* == **null**)

{

*instance* = **new** Singleton();

}

**return** *instance*;

}

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

{

Singleton text = Singleton.*getInstance*();

Singleton text2 = Singleton.*getInstance*();

System.***out***.println(text.str);

System.***out***.println(text2.str);

singletoninheritancecheck.*Check*(text,text2);

}

}

[**Q2**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2131)**. Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes of the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use polymorphism i.e. method overriding.**

**Description:-**

Write a program to claculate the total salaries given to all the employees by an organization. In this organization there are employees category under which Manager and Labour are working where manager will recieve incentive and labourer will receive overtime benifits.

**import** java.util.ArrayList;

**import** java.util.List;

**class** manager **extends** Assignment2Q2 {

manager(**int** id, **int** salary) {

**super**(id, salary);

}

@Override

**public** **int** getSalary(**int** salary ) {

**int** incentive = 5000;

salary += incentive;

salaryEarned += salary;

*employeeSalaries*.add(salaryEarned);

**return** (salary);

}

}

**class** Labour **extends** Assignment2Q2 {

Labour(**int** id, **int** salary) {

**super**(id, salary);

}

@Override

**public** **int** getSalary(**int** salary) {

**int** overtime = 500 ;

**int** salaryForHours = (salary/365)/9;

salary += (salaryForHours\*1.5)\*overtime;

**return** salary;

}

}

**public** **class** Assignment2Q2 {

// Assignment2Qs -Employee class

**int** salary=10000;

**int** salaryEarned = 0 ;

**int** id;

**static** **int** *total*;

**static** List<Integer> *employeeSalaries*;

Assignment2Q2(**int** id ,**int** salary){

**this**.id = id ;

**this**.salary = salary;

salaryEarned = 0;

}

**public** **int** getSalary(**int** salary){

**return** salary;

}

**public** **int** reciveSalary() {

salary = getSalary(salary);

salaryEarned += salary;

**return** salaryEarned;

}

**public** **static** **int** totalEmployeesSalary(List<Integer> employeeSalaries){

**for**(**int** i =0;i<employeeSalaries.size();i++){

*total* += employeeSalaries.get(i);

}

**return** *total*;

}

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

*employeeSalaries* = **new** ArrayList<>();

Assignment2Q2 emp1 = **new** Assignment2Q2(1,10000);

Assignment2Q2 emp2 = **new** Assignment2Q2(2,40000);

manager emp3 = **new** manager(2,40000);

Labour emp4 = **new** Labour(2,20000);

*employeeSalaries*.add(emp1.reciveSalary());

*employeeSalaries*.add(emp2.reciveSalary());

*employeeSalaries*.add(emp3.reciveSalary());

*employeeSalaries*.add(emp4.reciveSalary());

System.***out***.println(*totalEmployeesSalary*(*employeeSalaries*));

}

}

[**Q3**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2132)**. Write a program to consider saving & current account in the bank. Saving account holder has ‘Fixed Deposits’ whereas Current account holder has cash credit. Apply polymorphism to find out total cash in the bank.**

**Description:-**

Write a program to claculate the total cash available in the bank. In this bank,bank will provide two types of accounts one is savings accounts and another is current account where current account has cash credits and savings account has fixed deposit options.

**import** java.util.ArrayList;

**import** java.util.List;

**class** CurrentAccount **extends** Assignment2Q3 {

**int** creditLimit = 2000;

**public** CurrentAccount(String name) {

**super**(name);

}

@Override

**public** **int** withdraw(**int** amount) {

**if** (getCash() > amount) {

totalDeposits -= amount;

**return** -(amount);

}

**else** **if**((getCash()+creditLimit)>amount){

**this**.creditLimit -= amount-totalDeposits;

totalDeposits -= amount;

**return** -(amount);

}

**else** {

**return** 0;

}

}

@Override

**public** **int** getCash() {

**return** (totalDeposits);

}

}

**class** SavingsAccount **extends** Assignment2Q3 {

**int** fixedDepositAmount = 5000;

**public** SavingsAccount(String name) {

**super**(name);

}

@Override

**public** **int** getCash() {

**return** (fixedDepositAmount+totalDeposits);

}

@Override

**public** **int** withdraw(**int** amount) {

**if** (getCash() > amount) {

totalDeposits -= amount;

**return** -(amount);

} **else** {

**return** 0;

}

}

}

**public** **class** Assignment2Q3 {

**static** ArrayList<Integer> *totalCashInBank*;

**private** String name;

**protected** **int** totalDeposits=0;

**public** Assignment2Q3(String name) {

**this**.name = name;

}

**public** **int** deposit(**int** amount) {

**this**.totalDeposits += amount;

**return** amount;

}

**public** **int** withdraw(**int** amount) {

**if** (totalDeposits > amount) {

totalDeposits -= amount;

**return** -amount;

} **else** {

**return** 0;

}

}

**public** **static** **int** totalCashInBank(ArrayList<Integer> cash){

**int** total = 0;

**for**(**int** i =0;i<cash.size();i++){

total += cash.get(i);

}

**return** total;

}

**public** **int** getCash(){

**return** totalDeposits;

}

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

*totalCashInBank* = **new** ArrayList<Integer>();

Assignment2Q3 acc1 = **new** SavingsAccount("acc1");

Assignment2Q3 acc2 = **new** CurrentAccount("acc2");

SavingsAccount acc3 = **new** SavingsAccount("acc3");

CurrentAccount acc4 = **new** CurrentAccount("acc4");

*totalCashInBank*.add(acc1.deposit(100));

*totalCashInBank*.add(acc2.deposit(3000));

*totalCashInBank*.add(acc3.deposit(400));

*totalCashInBank*.add(acc4.deposit(300));

*totalCashInBank*.add(acc3.withdraw(400));

*totalCashInBank*.add(acc2.withdraw(3300));

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

}

}

**Q4. Test the following principles of an abstract class:**

**• If any class has any of its method abstract then you must declare entire class abstract.**

**• Abstract class cannot be instantiated.**

**• When we extend an abstract class, we must either override all the abstract methods in sub class or declare subclass as abstract.**

**• Abstract class cannot be private.**

**• Abstract class cannot be final.**

**• You can declare a class abstract without having any abstract method.**

**Description:-**

Write a program in such a way that all the conditions above for abstract class should satisfy.

**abstract** **class** name {

String name;

name(String name){

**this**.name =name;

}

**public** **abstract** String getname();

//Point 1 : here the method declared is abstract so the class should be abstract ;

}

**class** person **extends** name{

person(String name) {

**super**(name);

}

@Override

**public** String getname(){

**return** ("Hello,"+**this**.name);

}

}

**abstract** **class** MyClass {

**public** **abstract** **void** dis1();

}

**class** AbstClassP3 **extends** MyClass{

@Override

**public** **void** dis1(){

System.***out***.println("This is the subclass implementation of the display method");

}

}

**abstract** **class** MyClass2 {

**public** **abstract** **void** dis2();

}

**class** AbstClassP4 **extends** MyClass2{

**public** **void** dis2(){

System.***out***.println("This is not the Private Class");

}

}

**abstract** **class** MyClass3 {

**public** **void** dis3(){

System.***out***.println("this is an abstract class without abstract method");

}

}

**class** AbstClassP6 **extends** MyClass3{

}

**public** **class** Assignment2Q4 {

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

person user1 = **new** person("Ayushman jha");

System.***out***.println(user1.getname());

**new** AbstClassP3().dis1();

**new** AbstClassP4().dis2();

**new** AbstClassP6().dis3();

}

}

[**Q5**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2134)**. Write the classes Line, Rectangle, Cube etc. & make the Shape as their base class. Add an abstract draw() method in the class Shape & draw all shapes.**

**Description:-**

Write a java class named Shape which has abstract draw() and also classes Line, Rectangle, Cube etc. extends Shape to implement the draw method.

**class** Rectangle5 **extends** Shape5 {

@Override

**public** String draw() {

**return** "The Rectangle has been drawn : []";

}

}

**class** Line5 **extends** Shape5{

@Override

**public** String draw() {

**return** "The line has been drawn : ------";

}

}

**class** Cube5 **extends** Shape5 {

@Override

**public** String draw() {

**return** "The Cube has been drawn : ()";

}

}

**abstract** **class** Shape5 {

**abstract** **public** String draw();

}

**public** **class** Assignment2Q5{

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

Rectangle5 R1 = **new** Rectangle5 ();

Line5 L1 = **new** Line5 ();

Cube5 C1 = **new** Cube5 ();

System.***out***.println(R1.draw());

System.***out***.println(L1.draw());

System.***out***.println(C1.draw());

}

}

[**Q6**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2145)**. Write an abstract class ‘Persistence’ along with two sub classes ‘FilePersistence’ & ‘DatabasePersistence’. The base class with have an abstract method persist() which will be overridden by its sub classes. Write a client who gets the Persistence object at runtime & invokes persist() method on it without knowing whether data is being saved in File or in Database.**

**Description:-**

Write a program having an abstract class "Persistence" which have two child classes "FilePersistence","DatabasePersistence".The base class will have a persist() method and it is overridden by sub classes.wite a seperate class of client which will get persistence object and invoke persist method on it.(Polymorphism)

**abstract** **class** Persistence{

**abstract** **public** String persist();

}

**class** FilePersistence **extends** Persistence{

@Override

**public** String persist() {

**return** "File Persistence Method";

}

}

**class** DataBasePersistence **extends** Persistence{

@Override

**public** String persist() {

**return** "Data Base Persistence Method";

}

}

**class** Client {

**public** **static** Persistence getPersistence()

{

**return** **new** FilePersistence();

//return new DataBasePersistence();

}

}

**public** **class** q6 {

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

{

Persistence Persis = Client.*getPersistence*();

System.***out***.println(Persis.persist());

}

}

[**Q7**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2146)**. Develop an application for Dessert shop. The application should allow owner to add items like Candy, Cookie or Ice Cream in the shop storage. Also customers should be able to place an order.**

**DessertItem is an abstract class having an abstract method getCost(). Every dessert item has tax associated. Candy item is sold in dollar currency, Cookie in Euro currency & Ice Cream in Rupees currency. The sub classes are supposed to override these methods. When we run the application, it should ask us our role i.e. owner or customer. If role is owner, we should be able to add dessert items in our storage. If role is customer, then we should be able to place an order. The currency conversion rates are:**

**1 dollar = 60 rupees.**

**1 euro = 70 rupees.**

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**abstract** **class** DesertItem {

**abstract** **public** **int** getCost();

**public** List<String> getList() {

List<String> list = **new** ArrayList<>();

list.add("Candy");

list.add("Cooky");

list.add("IceCream");

**return** list;

}

**static** **int** *totalAmount* =0;

}

**class** Candy **extends** DesertItem {

List<String> itemList = **new** ArrayList<>();

**public** List<String> addItem(String name) {

itemList.add(name);

**return** itemList;

}

**public** **static** **int** addCandies(**int** candies){

*totalAmount* +=candies\*60;

**return** *totalAmount*;

}

@Override

**public** **int** getCost() {

**return** *totalAmount*;

}

**public** List<String> getList() {

**return** itemList;

}

}

**class** Cookie **extends** DesertItem {

List<String> itemList = **new** ArrayList<>();

**public** List<String> addItem(String name) {

itemList.add(name);

**return** itemList;

}

**public** **int** addCookies(**int** cookies){

*totalAmount* += cookies\*70;

**return** *totalAmount*;

}

@Override

**public** **int** getCost() {

// **TODO** Auto-generated method stub

**return** *totalAmount*;

}

@Override

**public** List<String> getList() {

// **TODO** Auto-generated method stub

**return** itemList;

}

}

**class** IceCream **extends** DesertItem {

List<String> itemList = **new** ArrayList<>();

**public** List<String> addItem(String name) {

itemList.add(name);

**return** itemList;

}

**public** **int** addIceCreams(**int** iceCream){

*totalAmount* += iceCream\*70;

**return** *totalAmount*;}

@Override

**public** **int** getCost() {

// **TODO** Auto-generated method stub

**return** *totalAmount*;

}

@Override

**public** List<String> getList() {

// **TODO** Auto-generated method stub

**return** itemList;

}

}

**class** TotalCost **extends** DesertItem {

@Override

**public** **int** getCost() {

// **TODO** Auto-generated method stub

**return** *totalAmount*;

}

}

**public** **class** q7 {

**static** Candy *objCandy* = **new** Candy();

**static** Cookie *objCookie* = **new** Cookie();

**static** IceCream *objIceCream* = **new** IceCream();

**static** TotalCost *objCost* = **new** TotalCost();

**static** Scanner *scanner*=**new** Scanner(System.***in***);

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

*objCandy*.addItem("Kimi");

*objCandy*.addItem("Melody");

*objCandy*.addItem("Pulse");

*objCookie*.addItem("Hide and Seek");

*objCookie*.addItem("GoodDay");

*objCookie*.addItem("MariGold");

*objIceCream*.addItem("Amul");

*objIceCream*.addItem("Kwality Wall's");

*objIceCream*.addItem("Arun");

System.***out***.print(*selectRoles*());

String role= *scanner*.nextLine();

String quit = **null**;

**while**(quit!="quit") {

System.***out***.print(*roles*(role));

String select = *scanner*.nextLine();

**if**(select.equals("Add Item")) {

System.***out***.print(*addItems*());

}

**else** **if** (select.equals("Display Items")) {

System.***out***.print(*displayItems*());

}

**else** **if** (select.equals("Place Order")) {

System.***out***.print(*placeOrder*());

}

**else** **if** (select.equals("Get Amount")) {

System.***out***.print(*displayTotalCost*());

}

**else** **if**(select.equals("quit")) {

quit="quit";

}

}

System.***out***.println("Thank You");

}

**private** **static** String selectRoles(){

**return** "Select Role \n\"Owner\" or \"Customer\"\n:";

}

**private** **static** String roles(String role){

String Operations;

**if**(role.equals("Owner")) {

Operations="\"Add Item\" or \"Display Items\"\n(enter quit if you want to exit)\n:";

}

**else** **if**(role.equals("Customer")){

Operations="\"Place Order\" or \"Get Amount\"(enter quit if you want to exit)\n:";

}

**else** {

Operations="enter corrct role as show above";

}

**return** Operations;

}

**private** **static** String addItems() {

System.***out***.print("Enter the Desert you want to add (i.e : Candy , Cooky , IceCream)\n:");

String item = *scanner*.next();

**if**(item.equals("Candy")) {

System.***out***.println("Enter Name of Candy\n:");

String candyName = *scanner*.next();

*objCandy*.addItem(candyName);

}

**else** **if** (item.equals("Cooky")) {

System.***out***.println("Enter Name of Cooky\n:");

String cookyName = *scanner*.next();

*objCookie*.addItem(cookyName);

}

**else** **if**(item.equals("IceCream")) {

System.***out***.println("Enter Name of IceCream\n:");

String iceCreamName = *scanner*.next();

*objIceCream*.addItem(iceCreamName);

}

**return** item+" added successfully\n";

}

**private** **static** List<String> displayItems() {

List<String> itemsList = **new** ArrayList<>();

System.***out***.println("Enter the Desert want you to display (i.e : Candy , Cooky , IceCream)\n:");

String item = *scanner*.next();

**if**(item.equals("Candy"))

itemsList=*objCandy*.getList();

**else** **if** (item.equals("Cooky")) {

itemsList=*objCookie*.getList();

}

**else** **if**(item.equals("IceCream")) {

itemsList=*objIceCream*.getList();

}

**return** itemsList;

}

**private** **static** String placeOrder() {

System.***out***.println("Enter the Desert want you to Place order (i.e : Candy , Cooky , IceCream)\n");

String item = *scanner*.next();

**if**(item.equals("Candy")) {

System.***out***.println(*objCandy*.getList());

System.***out***.print("Enter the candy you want from above list:");

String candy = *scanner*.next();

System.***out***.print("Enter the quantity you want: ");

**int** quantity = *scanner*.nextInt();

*objCandy*.*addCandies*(quantity);

}

**else** **if** (item.equals("Cooky")) {

System.***out***.println(*objCookie*.getList());

System.***out***.print("Enter the cookies you want from above list:");

String cooky = *scanner*.next();

System.***out***.print("Enter the quantity you want: ");

**int** quantity = *scanner*.nextInt();

*objCookie*.addCookies(quantity);

}

**else** **if**(item.equals("IceCream")) {

System.***out***.println(*objIceCream*.getList());

System.***out***.print("Enter the IceCream you want from above list:");

String iceCream = *scanner*.next();

System.***out***.print("Enter the quantity you want: ");

**int** quantity = *scanner*.nextInt();

*objCandy*.*addCandies*(quantity);

}

**return** item+" place Successfully\n";

}

**private** **static** **int** displayTotalCost() {

System.***out***.print("Total cost Rupess: ");

**return** *objCost*.getCost();

}

}