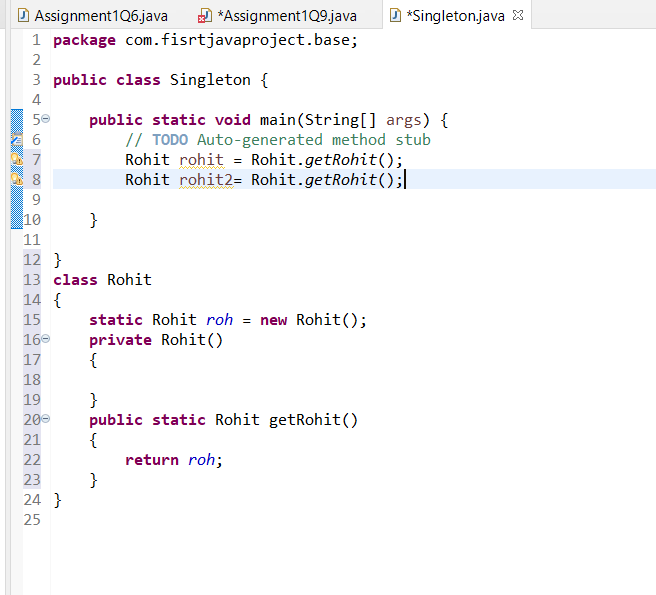
**[Q1](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2149" \o "Q1). Write a singleton class. Confirm that singleton class cannot be inherited.**

**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.

Steps to create :

1. Create a static instance of a class in the class itself .
2. Create a private constructor of the class , not the default one.
3. Create a static method which will return object of a class .



[**Q2**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2260)**. 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.

CODE :

**package** com.fisrtjavaproject.base;

**import** java.util.ArrayList;

**class** Manager **extends** Assignment2Q2 {

@Override

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

**int** incentive = 5000;

**return** salary+incentive;

}

}

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

@Override

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

**int** overtime = 500;

**return** salary+overtime;

}

}

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

**static** **int** *salary* = 10000;

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

{

**return** salary;

}

**public** **int** totalEmployeesSalary(ArrayList<Integer> employeeSalaries){

**int** total = 0;

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

total += employeeSalaries.get(i);

**return** total;

}

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

Manager m=**new** Manager();

**int** msalary=m.getSalary(*salary*);

Labour l=**new** Labour();

**int** lsalary=l.getSalary(*salary*);

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

employeeSalaries.add(msalary);

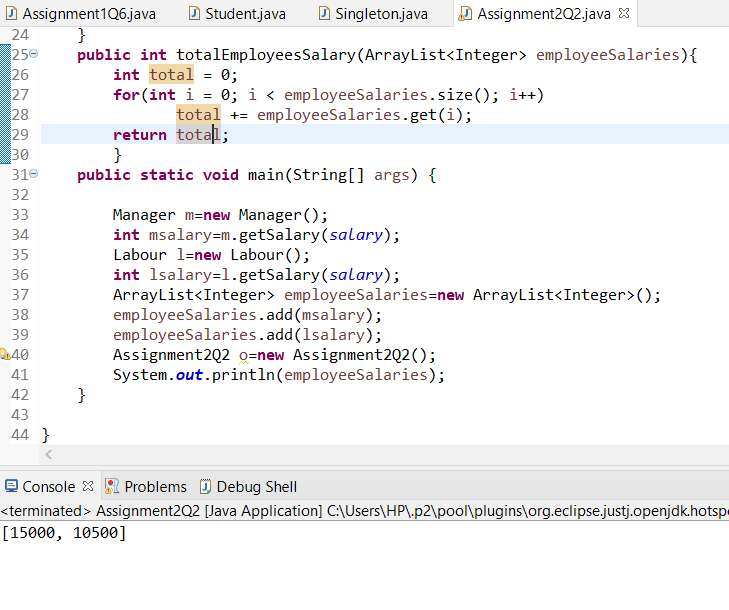
employeeSalaries.add(lsalary);

Assignment2Q2 o=**new** Assignment2Q2();

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

}

}



[**Q3**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2261)**. 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.

CODE :

**package** com.fisrtjavaproject.base;

**import** java.util.ArrayList;

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

**int** totalDeposits = 10000;

**int** creditLimit = 2000;

@Override

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

**return** totalDeposits-creditLimit;}

}

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

**int** totalDeposits = 10000;

**int** fixedDepositAmount = 5000;

@Override

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

**return** totalDeposits+fixedDepositAmount;}

}

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

**public** **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** 0;

}

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

CurrentAccount cAccount=**new** CurrentAccount();

SavingsAccount sAccount=**new** SavingsAccount();

**int** cAmount=cAccount.getCash();

**int** sAmount=sAccount.getCash();

ArrayList<Integer> cash=**new** ArrayList<Integer>(2);

cash.add(cAmount);

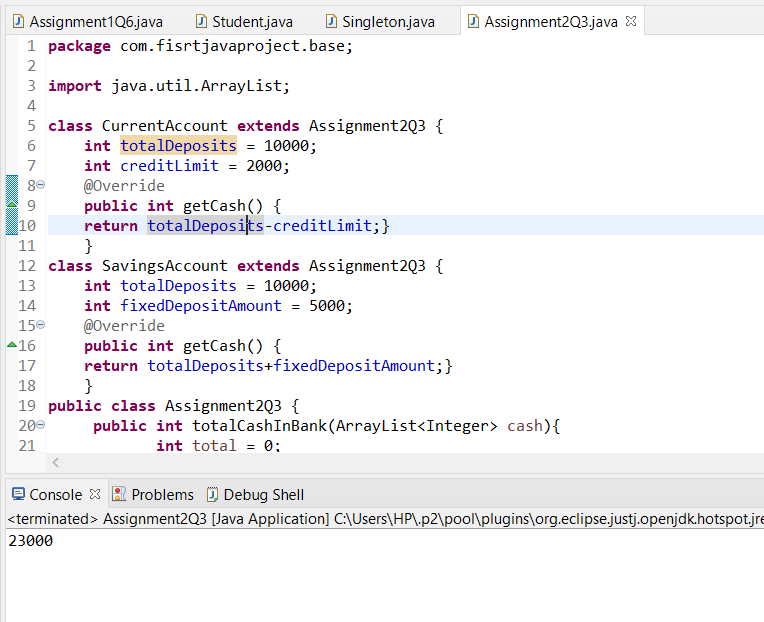
cash.add(sAmount);

Assignment2Q3 o=**new** Assignment2Q3();

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

}

}



**4. 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.**

CODE :

**package** com.fisrtjavaproject.base;

**abstract** **class** X{

**int** abstract\_var;

}

**abstract** **class** Y **extends** X{

**public** **abstract** **void** fun();

}

**public** **class** Assignment2Q4 **extends** Z {

@Override

**public** **void** fun() {

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

}

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

{

B obj =**new** Assignment2Q4();

b.fun();

}

}

[**Q5**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2163)**. 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.**

CODE :

**package** com.fisrtjavaproject.base;

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

@Override

**public** String draw() {

**return** "Rectangle";}

}

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

@Override

**public** String draw() {

**return** "Line";}

}

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

@Override

**public** String draw() {

**return** "Cube";

}

}

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

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

}

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

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

Shape5 s1=**new** Cube5();

Shape5 s2=**new** Line5();

Shape5 s3=**new** Rectangle5();

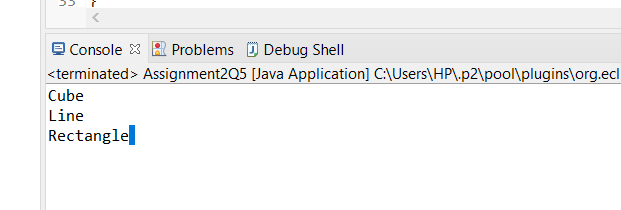
System.***out***.println(s1.draw());

System.***out***.println(s2.draw());

System.***out***.println(s3.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.**

**CODE :**

**package** com.fisrtjavaproject.base;

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

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

}

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

@Override

**public** String persist() {**return** "Persistence File";}

}

**class** DatabasePersistence **extends** Persistence{

@Override

**public** String persist() {**return** "Persistence Database";}

}

**public** **class** Assignment2Q6 {

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

Persistence f=**new** FilePersistence();

Persistence d=**new** DatabasePersistence();

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

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

}

}

