Abstract class in java

* Abstract classes may or may not contain *abstract methods*, i.e., methods without body ( public void get(); )
* But, if a class has at least one abstract method, then the class **must** be declared abstract.
* If a class is declared abstract, it cannot be instantiated.
* To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
* If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Declaring a method as abstract has two consequences −

* The class containing it must be declared as abstract.
* Any class inheriting the current class must either override the abstract method or declare itself as abstract.

Let’s create a scenario when to use abstract class first. We know that all banks have some minimum interest rate and minimum loan amount that customer can use. Banks also have rules that they can give loan upto certain age group only. Now every bank must these features but the minimum interest rate and minimum loan amount can be different for different banks. For example SBI bank has its own minimum interest rate and minimum load rate while Nepal Investment bank has its own. That’s where abstract class comes in handy.

In the example below we create an abstract class called Bank and gave it two methods getMinimumInterestRate() and getMinimumLoanAmounts(). We set these two abstract methods to abstract class Bank so that classes that extend Bank class can implement these two abstract methods and set their own implementations.

Consider two classes SBIBank and NepalInvestmentBank are the two classes to extend class Bank and must implement abstract methods. And each these two classes will have their own function body for the methods they just implemented from the abstract class. Such way two different banks can specify their own minimum interest rate and minimum load amounts for the same method.

Another feature of abstract class is that every method inside it is not necessarily abstract, i.e abstract class can have non abstract method too and for that non abstract method it should have method body as well just like any other method. This non abstract method can be used as common method for both SBIBank and NepalInvestmentBank.

Abstractclass can also have constructors.

Bank.java

**public** **abstract** **class** Bank {

**abstract** **void** getInterestRate();

**abstract** **void** getMinimumLoanAmount();

**void** ageOfLoad() {

System.*out*.println("The minimum age of all banks is: 80 years");

}

}

NepalInvestment.java

**public** **class** NepalInvestment **extends** Bank {

**void** getInterestRate() {

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

System.*out*.println("Nepal Investment bank: Interest rate = 7%");

}

**void** getMinimumLoanAmount() {

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

System.*out*

.println("Nepal Investment bank: Minimum Load Amout = Rs 5000");

}

}

SBIBank.java

**public** **class** SBIBank **extends** Bank {

**void** getInterestRate() {

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

System.*out*.println("SBI bank: Interest rate = 12%");

}

**void** getMinimumLoanAmount() {

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

System.*out*.println("SBI bank: Interest rate = Rs 20000");

}

}

UserClass.java

**public** **class** UserClass {

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

Bank nepalInvest = **new** NepalInvestment();

nepalInvest.getMinimumLoanAmount();

nepalInvest.getInterestRate();

Bank sbiBank = **new** SBIBank();

sbiBank.getMinimumLoanAmount();

sbiBank.getInterestRate();

sbiBank.ageOfLoad();// this is common for both classes

}

}

The abstract class can also be used to provide some implementation of the interface. In such case, the end user may not be forced to override all the methods of the interface.

interface A{

void a();

void b();

void c();

void d();

}

abstract class B implements A{

public void c(){System.out.println("I am C");} //now this is non abstract method

}

class M extends B{

public void a(){System.out.println("I am a");}

public void b(){System.out.println("I am b");}

public void d(){System.out.println("I am d");}

}

class Test5{

public static void main(String args[]){

A a=new M();

a.a();

a.b();

a.c(); //now this is non abstract method

a.d();

}}

How to use Abstract class as interface (for callbacks)?

* Do this program in eclipse not android studio.

MyAbstractClass.java

**public** **abstract** **class** MyAbstractClass {

**public** **abstract** **boolean** isInternet(**boolean** status);

}

CheckInternet.java

**public** **class** CheckInternet {

MyAbstractClass show;

**public** CheckInternet(MyAbstractClass show) {

**this**.show = show;

}

Runnable run = **new** Runnable() {

@Override

**public** **void** run() {

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

**try** {

System.*out*.println("Checking for internet, wait for 2 sec");

Thread.*sleep*(2000);

show.isInternet(**false**);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

};

Thread thr = **new** Thread(run);

}

MainClass.java

**public** **class** MainClass {

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

MyAbstractClass show = **new** MyAbstractClass() {

@Override

**public** **boolean** isInternet(**boolean** fetchedstatus) {

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

System.*out*.print("the status of internet is: " + fetchedstatus);

**return** fetchedstatus;

}

};

**new** CheckInternet(show).thr.start();

}

}

How to use multiple abstract classes in single class?

* Can be used if you have multiple callback methods and you want to group them in single class making less number of classes.
* Make sure your imports are correct while creating instances of inner classes.
* Same program below can be done with interfaces instead of abstract classes. Example is in github.

MyAbstractClasses.java

**package** multipleabstractions;

**public** **class** MyAbstractClasses {

**public** **abstract** **class** InternetAbstract {

**public** **abstract** **void** checkInternetStatus(**boolean** status);

}

**public** **abstract** **class** WifiAbstract {

**public** **abstract** **void** checkWifiStatus(**boolean** status);

}

**public** **abstract** **class** BattiAbstract {

**public** **abstract** **void** checkBattiStatus(**boolean** status);

}

}

InternetChecker.java

**package** multipleabstractions;

**import** multipleabstractions.MyAbstractClasses.InternetAbstract;

**public** **class** InternetChecker {

InternetAbstract internetAbs;

**public** InternetChecker(InternetAbstract abs1) {

**this**.internetAbs = abs1;

}

Thread thr = **new** Thread(**new** Runnable() {

**public** **void** run() {

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

**try** {

System.*out*

.println("Wait for 2 seconds, we're checking for the internet.");

Thread.*sleep*(2000);

internetAbs.checkInternetStatus(**true**);

} **catch** (InterruptedException e) {

// **TODO**: handle exception

}

}

});

}

WifiChecker.java

**package** multipleabstractions;

**import** multipleabstractions.MyAbstractClasses.WifiAbstract;

**public** **class** WifiChecker {

WifiAbstract wifiAbs;

**public** WifiChecker(WifiAbstract wifiAbs) {

**this**.wifiAbs = wifiAbs;

}

Thread thr = **new** Thread(**new** Runnable() {

**public** **void** run() {

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

System.*out*

.println("Wait for 2 seconds, we're checking wifi status");

**try** {

Thread.*sleep*(2000);

wifiAbs.checkWifiStatus(**false**);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

});

}

BattiChecker.java

**package** multipleabstractions;

**import** multipleabstractions.MyAbstractClasses.BattiAbstract;

**public** **class** BattiChecker {

BattiAbstract battiaAbs;

**public** BattiChecker(BattiAbstract battiAbs) {

**this**.battiaAbs = battiAbs;

}

Thread thr = **new** Thread(**new** Runnable() {

**public** **void** run() {

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

**try** {

System.*out*

.println("Wait for 2 seconds while we check for batti: ");

Thread.*sleep*(2000);

battiaAbs.checkBattiStatus(**true**);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

});

}

MainClass.java

package multipleabstractions;

import multipleabstractions.MyAbstractClasses.InternetAbstract;

import multipleabstractions.MyAbstractClasses.WifiAbstract;

public class MainClass {

public static void main(String[] args) {

InternetAbstract internetAbs = new MyAbstractClasses().new InternetAbstract() {

public void checkInternetStatus(boolean status) {

// TODO Auto-generated method stub

System.out

.println("Congratulations !!! there is internet status: "

+ status);

}

};

new InternetChecker(internetAbs).thr.start();

// now check for wifi status

WifiAbstract wifiAbs = new MyAbstractClasses().new WifiAbstract() {

public void checkWifiStatus(boolean status) {

// TODO Auto-generated method stub

System.out.println("Oops !!! No wifi for you fucker.. "

+ status);

}

};

new WifiChecker(wifiAbs).thr.start();

}

}

MainClass2.java

**package** multipleabstractions;

**import** multipleabstractions.MyAbstractClasses.BattiAbstract;

**public** **class** MainClass2 {

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

BattiAbstract battiAbs = **new** MyAbstractClasses().**new** BattiAbstract() {

**public** **void** checkBattiStatus(**boolean** status) {

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

System.*out*

.println("Congratulations !!! aja ghar ma batti cha.");

}

};

**new** BattiChecker(battiAbs).thr.start();

}

}