Lớp trừu tượng (Lớp Abstract) trong Java

Một lớp được khai báo với từ khóa abstract được xem như là lớp abstract trong Java. Nó có thể có các phương thức abstract hoặc non-abtract. Trước khi tìm hiểu về lớp trừu tượng trong Java, bạn cần hiểu tính trừu tượng trong Java là gì.

Tính trừu tượng (Abstraction) trong Java

Tính trừu tượng là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Nói cách khác, nó chỉ hiển thị các thứ quan trọng tới người dùng và ẩn các chi tiết nội tại, ví dụ: để gửi tin nhắn, người dùng chỉ cần soạn text và gửi tin. Bạn không biết tiến trình xử lý nội tại về phân phối tin nhắn. Tính trừu tượng giúp bạn trọng tâm hơn vào đối tượng thay vì quan tâm đến cách nó thực hiện.

Lớp abstract trong Java

Một lớp được khai báo là abstract thì đó là lớp trừu tượng. Nó cần được kế thừa và phương thức của nó được triển khai. Nó không thể được thuyết minh.

Sử dụng từ khóa **abstract** để khai báo một lớp abstract. Từ khóa này xuất hiện trước từ khóa class trong khai báo lớp. Ví dụ:

```
abstract class A{}
```

Có hai cách để đạt được tính trừu tượng hóa trong Java:

- Lớp abstract (0 tới 100%)
- Interface (100%)

Phương thức trừu tượng trong Java

Một phương thức được khai báo là abstract và không có trình triển khai thì đó là phương thức trừu tượng.

Nếu bạn muốn một lớp chứa một phương thức cụ thể nhưng bạn muốn triển khai thực sự phương thức đó để được quyết định bởi các lớp con, thì bạn có thể khai báo phương thức đó trong lớp cha ở dạng abstract.

Từ khóa abstract được sử dụng để khai báo một phương thức dạng abstract. Một phương thức gồm một ký số, và không có thân phương thức.

Phương thức abstract sẽ không có định nghĩa, và ký số của nó được theo sau bởi dấu chấm phảy, không có dấu ngoặc móc ôm theo sau:

```
abstract void printStatus();//khong co than phuong thuc va la abstract
```

Ví dụ về lớp trừu tượng và phương thức trừu tượng trong Java

Trong ví dụ này, Bike là lớp trừu tượng chỉ chứa một phương thức trừu tượng là run. Trình triển khai của nó được cung cấp bởi lớp Honda.

```
abstract class Bike{
  abstract void run();
}

class Honda4 extends Bike{
void run(){System.out.println("dang chay mot cach an toan..");}

public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Ví dụ thực về lớp abstract trong Java

Trong ví dụ này, Shape là lớp trừu tượng, trình triển khai của nó được cung cấp bởi lớp Rectangle và lớp Circle. Hầu như chúng ta không biết gì về lớp Implementation (bị ẩn tới người dùng) và đối tượng của lớp Implementation được cung cấp bởi phương thức factory. Phương thức factory là phương thức mà trả về sự thể hiện của lớp. Chúng ta sẽ học về phương thức này sau.

Trong ví dụ sau, nếu chúng ta tạo sự thể hiện của lớp Rectangle, phương thức **draw()** của lớp Rectangle sẽ được triệu hồi.

File: TestAbstraction1.java

```
abstract class Shape{
abstract void draw();
}
```

```
//Trong tinh huong nay, trinh trien khai duoc cung cap boi ai do, vi du: nguoi su dung cuoi cung nao do

class Rectangle extends Shape{

void draw(){System.out.println("Ve hinh chu nhat");}

}


class Circle1 extends Shape{

void draw(){System.out.println("Ve hinh tron");}

}


//Trong tinh huong nay, phuong thuc duoc goi boi lap trinh vien hoac nguoi dung

class TestAbstraction1{

public static void main(String args[]){

Shape s=new Circle1();//Trong tinh huong nay, doi tuong duoc cung cap thong qua phuong thuc, chang han nhu getShape()

s.draw();

}

}
```

Ví dụ khác về lớp trừu tượng trong Java

File: TestBank.java

```
abstract class Bank{

abstract int getRateOfInterest();

}


class SBI extends Bank{

int getRateOfInterest(){return 7;}

}

class PNB extends Bank{

int getRateOfInterest(){return 7;}

}


class TestBank{

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

```
Bank b=new SBI();//Neu doi tuong la PNB, phuong thuc cua PNB se duoc trieu hoi
int interest=b.getRateOfInterest();
System.out.println("Ti le lai suat la: "+interest+" %");
}}
```

Lớp trừu tượng có thể có thành viên dữ liệu, phương thức trừu tượng, constructor, và có thể cả phương thức **main()**.

File: TestAbstraction2.java

```
//vi du ve lop abstract ma co than phuong thuc
 abstract class Bike{
   Bike(){System.out.println("bike duoc tao");}
   abstract void run();
   void changeGear(){System.out.println("gear duoc thay doi");}
 }

 class Honda extends Bike{
 void run(){System.out.println("dang chay mot cach an toan..");}
 }
 class TestAbstraction2{
 public static void main(String args[]){
  Bike obj = new Honda();
  obj.run();
  obj.changeGear();
 }
}
```

Qui tắc: Nếu bạn đang kế thừa bất cứ lớp trừu tượng nào mà có phương thức trừu tượng, thì bạn phải hoặc cung cấp trình triển khai của phương thức hoặc tạo lớp trừu tượng này.

Lớp trừu tượng cũng có thể được sử dụng để cung cấp một số trình triển khai của Interface. Trong tình huống này, người dùng cuối cùng không thể bị bắt buộc phải ghi đè tất cả phương thức của Interface đó.

Ghi chú: Nếu bạn mới học về Java, thì học Interface trước và bỏ qua ví dụ này.

```
interface A{
void a();
void b();
void c();
void d();
}

abstract class B implements A{
public void c(){System.out.println("Toi la C");}
}

class M extends B{
public void a(){System.out.println("Toi la a");}
public void b(){System.out.println("Toi la b");}
public void d(){System.out.println("Toi la d");}
}

class Test5{
public static void main(String args[]){
A a=new M();
a.a();
a.b();
a.c();
a.d();
}}
```

Kế thừa lớp Abstract trong Java

Chúng ta có thể kế thừa lớp Employee theo cách thông thường như sau:

```
/* File name : Salary.java */
public class Salary extends Employee
{
   private double salary; //Luong hang nam
   public Salary(String name, String address, int number, double
```

```
      salary)
   {
       super(name, address, number);
       setSalary(salary);
   }
   public void mailCheck()
   {
       System.out.println("Ben trong mailCheck cua lop Salary ");
       System.out.println("Kiem tra mail cua " + getName()
       + " with salary " + salary);
   }
   public double getSalary()
   {
       return salary;
   }
   public void setSalary(double newSalary)
   {
       if(newSalary >= 0.0)
       {
          salary = newSalary;
       }
   }
   public double computePay()
   {
      System.out.println("Luong tra cho " + getName());
      return salary/52;
   }
}
```

Ở đây, chúng ta không thể thuyết minh một Employee mới, nhưng nếu chúng ta thuyết minh một đối tượng Salary mới, đối tượng Salary này sẽ kế thừa 3 trường, 7 phương thức từ Employee.

```
/* File name : AbstractDemo.java */
public class AbstractDemo
```

```
{
   public static void main(String [] args)
   {
      Salary s = new Salary("Hoang", "HaDong, HN", 3, 8.00);
      Employee e = new Salary("Thanh", "XuanTruong, ND", 2, 9.50);

      System.out.println("Goi mailCheck su dung tham chieu Salary --");
      s.mailCheck();

      System.out.println("\n Goi mailCheck su dung tham chieu Employee --");
      e.mailCheck();
    }
}
```