ملخص شرح مفهوم ال Abstraction

swad-al-lail **في ٣١ مارس**

ماهو الAbstraction؟

اذا بحثنا في القاموس عن معنى كلمة Abstract (تجريد)، فسنجد انها تعني [خاصية التعامل مع الفكرة لا الحدث]. بمعنى اهمال التفاصيل الغير لازمة واستبدالها بما هو مهم وواضح.

مثال على ذلك: عندما تحاول ان ترسل ايميل الى شخص ما، فانك لن تهتم بالتفاصيل الصغيرة مثل مالذي يحدث بالضبط عندما تضغط على زر ارسال او البروتوكول المستخدم لنقل الرسالة. كل ما تريد عمله هو ان تكتب عنوان الرسالة والمحتوى ومستقبل الرسالة وترسلها.

نفس الشيء ينطبق في مفاهيم ال Object-Oriented. في الabstraction نهدف الى اخفاء تفاصيل الimplementation عن المستخدم، بمعنى اخر، المستخدم سيهتم بما الغرض من الobject بدلا عن كيفية قيامه به.

في الجافا، يمكننا تطبيق هذا المفهوم عن طريق انشاء abstract class او abstract interface، (عكسها concrete class او normal class)

الـAbstract Classes

الكلاس الذي يحتوي على كلمة **abstract** في تعريفه يعتبر **abstract class، وسيتبع القوانين التالية:**

- من الممكن للAbstract class أن يحتوي على abstract methods (ليس اجباريا)، وهي الميثودز التي لا تحتوي على تعريف مثل (public abstract void get();).
- ولكن، اذا احتوى الكلاس على abstract method، فيجب تعريفه ك abstract class
- اذا تم تعريف الكلاس ك abstract، فلا يمكن انشاء objects منه
- تستخدم ال abstract class وخصائصه، يجب عليك ان ترثه (inherit it) من كلاس آخر، ويجب عليك اعادة تعريف كل ال abstract methods

مثال:

سننشئ abstract class باضافة الكلمة abstract الى تعريف الكلاس:

```
public abstract class Employee {
private String name;
private String address;
private int number;

public Employee(String name, String address, int number) {
System.out.println(“Constructing an Employee”);
this.name = name;
this.address = address;
this.number = number;
}

public double computePay() {
System.out.println(“Inside Employee computePay”);
return 0.0;
}
```

```
public void mailCheck() {
System.out.println("Mailing a check to " + this.name + " " + this.address);
}

public String toString() {
return name + " " + address + " " + number;
}

public String getName() {
return name;
}

public String getAddress() {
return address;
}

public void setAddress(String newAddress) {
address = newAddress;
}

public int getNumber() {
return number;
}
}
```

لاحظ ان الabstract class السابق لا يختلف ابدا عن الكلاس العادي، فهو يحتوي على attributes, constructor, normal methods. الفرق الوحيد هو كلمة abstract بالتعريف

الآن لنحاول انشاء اوبجكت من الكلاس السابق:

```
public class AbstractDemo {

public static void main(String [] args) {
Employee e = new Employee(“George W.”, “Houston, TX”, 43);
e.computePay();
}
}
```

عند تشغيل البرنامج سنحصل على **خطأ**
لأنه لا يمكن انشاء اوبجكت من
abstract class حتى وان احتوى على constructor.

الـInheriting Abstract Classes

نستطيع اعادة استخدام خصائص وميثودز الabstract class تماما كأي كلاس آخر عن طريق وراثته (extends):

```
public class Salary extends Employee {
private double salary; // Annual salary

public Salary(String name, String address, int number, double salary) {
super(name, address, number);
setSalary(salary);
}
```

```
public void mailCheck() {
System.out.println("Within mailCheck of Salary class ");
System.out.println("Mailing check to " + 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("Computing salary pay for " + getName());
return salary/52;
}
}
```

هنا لا يمكننا انشاء اوبجكت من كلاس Employee، ولكن يمكننا انشاء اوبجكت من كلاس Salary واستخدام جميع خصائص وميثودز كلاس Employee:

```
public class AbstractDemo {

public static void main(String [] args) {
Salary s = new Salary(“Mohd Mohtashim”, “Ambehta, UP”, 3, 3600.00);
Employee e = new Salary(“John Adams”, “Boston, MA”, 2, 2400.00);
System.out.println(“Call mailCheck using Salary reference --”);
s.mailCheck();
System.out.println("\n Call mailCheck using Employee reference–");
e.mailCheck();
}
}
```

الـAbstract Methods

اذا اردت انشاء ميثود ولكنك تريد تعريفها بالكلاس الإبن (Inheriting class)، يمكنك استخدام كلمة abstract في تعريف الميثود، مثال:

```
public abstract class Employee {
private String name;
private String address;
private int number;

public abstract double computePay();
}
```

تعريف الميثود (مثل computePay()) ك abstract method له آثار يجب الانتباه لها:

- الكلاس الذي يحتويها يجب أن يكون abstract
- الكلاس الذي يرث الكلاس الذي يحتويها يجب أن يعيد تعريفها

مثال، كلاس Salary سيرث كلاس Employee السابق:

```
public class Salary extends Employee {
private double salary; // Annual salary

public double computePay() {
System.out.println("Computing salary pay for " + getName());
return salary/52;
}
}
```

نلاحظ أن الميثود computePay() تمت اعادة تعريفها لتحصل على implementation جديد خاص بكلاس Salary

هنا نصل الى نهاية موضوعنا، أتمنى لي ولكم التوفيق والسداد.

---

kokosyria97ozngz **في ٣١ مارس**

😍😍👌