Контрольная работа - 4 семестр

Модификаторы доступа

Все члены класса в языке Java - поля и методы - имеют модификаторы доступа. В прошлых темах мы уже сталкивались с модификатором public. Модификаторы доступа позволяют задать допустимую область видимости для членов класса, то есть контекст, в котором можно употреблять данную переменную или метод.

В Java используются следующие модификаторы доступа:

public: публичный, общедоступный класс или член класса. Поля и методы, объявленные с модификатором public, видны другим классам из текущего пакета и из внешних пакетов.

private: закрытый класс или член класса, противоположность модификатору public. Закрытый класс или член класса доступен только из кода в том же классе.

protected: такой класс или член класса доступен из любого места в текущем классе или пакете или в производных классах, даже если они находятся в других пакетах

Модификатор по умолчанию. Отсутствие модификатора у поля или метода класса предполагает применение к нему модификатора по умолчанию. Такие поля или методы видны всем классам в текущем пакете.

Рассмотрим модификаторы доступа на примере следующей программы:

```
public class Program{

public static void main(String[] args) {

Person kate = new Person("Kate", 32, "Baker Street", "+12334567");
kate.displayName(); // норм, метод public
kate.displayAge(); // норм, метод имеет модификатор по умолчанию
kate.displayPhone(); // норм, метод protected
//kate.displayAddress(); // ! Ошибка, метод private

System.out.println(kate.name); // норм, модификатор по умолчанию
System.out.println(kate.address); // норм, модификатор public
System.out.println(kate.age); // норм, модификатор protected
//System.out.println(kate.phone); // ! Ошибка, модификатор private
}
}
class Person{

String name;
protected int age;
public String address;
private String phone;

public Person(String name, int age, String address, String phone){
this.name = name;
this.age = age;
this.address = address;
this.phone = phone;
}
public void displayName(){
System.out.printf("Name: %s \n", name);
}
void displayAge(){
System.out.printf("Age: %d \n", age);
}
private void displayAddress(){
```

```
System.out.printf("Address: %s \n", address);
}
protected void displayPhone(){
System.out.printf("Phone: %s \n", phone);
}}
```

В данном случае оба класса расположены в одном пакете - пакете по умолчанию, поэтому в классе Program мы можем использовать все метод и переменные класса Person, которые имеют модификатор по умолчанию, public и protected. А поля и методы с модификатором private в класс Program не будут доступны.

Если бы класс Program располагался бы в другом пакете, то ему были бы доступны только поля и методы с модификатором public.

Модификатор доступа должен предшествовать остальной части определения переменной или метода.

Инкапсуляция

Казалось бы, почему бы не объявить все переменные и методы с модификатором public, чтобы они были доступны в любой точке программы вне зависимости от пакета или класса? Возьмем, например, поле age, которое представляет возраст. Если другой класс имеет прямой доступ к этому полю, то есть вероятность, что в процессе работы программы ему будет передано некорректное значение, например, отрицательное число. Подобное изменение данных не является желательным. Либо же мы хотим, чтобы некоторые данные были достуны напрямую, чтобы их можно было вывести на консоль или просто узнать их значение. В этой связи рекомендуется как можно больше ограничивать доступ к данным чтобы защитить их от нежелательного доступа извне (как для получения значения, так и для его изменения). Использование различных модификаторов гарантирует, что данные не будут искажены или изменены не надлежащим образом. Подобное сокрытие данных внутри некоторой области видимости называется инкапсуляцией.

Так, как правило, вместо непосредственного применения полей используют методы доступа. Например:

```
public class Program{

public static void main(String[] args) {

Person kate = new Person("Kate", 30);
System.out.println(kate.getAge()); // 30
kate.setAge(33);
System.out.println(kate.getAge()); // 33
kate.setAge(123450);
System.out.println(kate.getAge()); // 33
}
}
class Person{

private String name;
private int age = 1;

public Person(String name, int age){

setName(name);
setAge(age);
}
public String getName(){
return this.name;
}
public void setName(String name){
this.name = name;
}
public int getAge(){
return this.age;
}
public void setAge(int age){
if(age > 0 && age < 110)
this.age = age;
}
}
```

И затем вместо непосредственной работы с полями name и age в классе Person мы будем работать с методами, которые устанавливают и возвращают значения этих полей. Методы setName, setAge и наподобие еще называют мьютейтерами (mutator), так как они изменяют значени поля. А методы getName, getAge и наподобие называют аксессерами (accessor), так как с их помощью мы получаем значение поля.

Причем в эти методы мы можем вложить дополнительную логику. Например, в данном случае при изменении возраста производится проверк насколько соответствует новое значение допустимому диапазону.

Ответить на вопросы

◄ Объявления

Перейти на...