Классы и объекты

Классы в языке Java объединяют поля класса, методы, конструкторы и логические блоки. Основные отличия от классов C++: все функции определяются внутри классов и называются методами; невозможно создать метод, не являющийся методом класса, или объявить метод вне класса; ключевое слово inline, как в C++, не поддерживается; спецификаторы доступа **public**, **private**, **protected** воздействуют только на те объявления полей, методов и классов, перед которыми они стоят, а не на участок от одного до другого спецификатора, как в C++; элементы по умолчанию не устанавливаются в **private**, а доступны для классов из данного пакета. Объявление класса имеет вид:

```
[спецификаторы] class ИмяКласса [extends СуперКласс]
    [implements список_интерфейсов]{/*определение класса*/}
```

Спецификатор доступа к классу может быть **public** (класс доступен в данном пакета и вне пакета), **final** (класс не может иметь подклассов), **abstract** (класс может содержать абстрактные методы, объект такого класса создать нельзя). По умолчанию спецификатор устанавливается в дружественный (friendly). Такой класс доступен только в текущем пакете. Спецификатор friendly при объявлении вообще не используется и не является ключевым словом языка. Это слово используется, чтобы как-то определить значение по умолчанию.

Любой класс может наследовать свойства и методы суперкласса, указанного после ключевого слова **extends**, и включать множество интерфейсов, перечисленных через запятую после ключевого слова **implements**. Интерфейсы представляют собой абстрактные классы, содержащие только нереализованные методы.

```
// пример # 5 : простой пример Java Beans класса: User.java
package chapt01;

public class User {
      public int numericCode;//нарушение инкапсуляции
      private String password;

      public void setNumericCode(int value) {
            if(value > 0) numericCode = value;
            else numericCode = 1;
      }
      public int getNumericCode() {
            return numericCode;
      }
      public void setPassword(String pass) {
            if (pass != null) password = pass;
            else password = "11111";
      }
      public String getPassword() {
      //public String getPass() {//некорректно – неполное имя метода
            return password;
      }
}
```

Класс **User** содержит два поля **numericCode** и **password**, помеченные как **public** и **private**. Значение поля **password** можно изменять только при помощи методов, например **setPassword()**. Поле **numericCode** доступно непосредственно через объект класса **User**. Доступ к методам и **public**-полям данного класса осуществляется только после создания объекта данного класса.

```
/*пример # 6 : создание объекта, доступ к полям
 и методам объекта: UserView.java : Runner.java */
package chapt01;

class UserView {
      public static void welcome(User obj) {
  System.out.printf("Привет! Введен код: %d, пароль: %s",
        obj.getNumericCode(), obj.getPassword());
      }
}
public class Runner {
       public static void main(String[] args) {
             User user = new User();
             user.numericCode = 71;//некорректно - прямой доступ
             // user.password = null; // поле недоступно
             user.setPassword("pass"); //корректно
             UserView.welcome(user);
     }
}
```

Компиляция и выполнение данного кода приведут к выводу на консоль следующей информации:

```
Привет! Введен код: 71, пароль: pass
```

Объект класса создается за два шага. Сначала объявляется ссылка на объект класса. Затем с помощью оператора **new** создается экземпляр объекта, например:

```
User user; //объявление ссылки
user = new User(); //создание объекта
```

Однако эти два действия обычно объединяют в одно:

```
User user = new User();/*объявление ссылки и создание объекта*/
```

Оператор **new** вызывает конструктор, поэтому в круглых скобках могут стоять аргументы, передаваемые конструктору. Операция присваивания для объектов означает, что две ссылки будут указывать на один и тот же участок памяти.

Сравнение объектов

Операции сравнения ссылок на объекты не имеют особого смысла, так как при этом сравниваются адреса. Для сравнения значений объектов необходимо использовать соответствующие методы, например, **equals()**. Этот метод наследуется в каждый класс из суперкласса **Object**, который лежит в корне дерева иерархии всех классов и переопределяется в произвольном классе для определения эквивалентности содержимого двух объектов этого класса.