Внутренние (inner) классы

Нестатические вложенные классы принято называть внутренними (inner) классами. Доступ к элементам внутреннего класса возможен из внешнего класса только через объект внутреннего класса, который должен быть создан в коде метода внешнего класса. Объект внутреннего класса всегда ассоциируется (скрыто хранит ссылку) с создавшим его объектом внешнего класса – так называемым внешним (enclosing) объектом. Внешний и внутренний классы могут выглядеть, например, так:

```
public class Ship {
      // поля и конструкторы
      // abstract, final, private, protected - допустимы
      public class Engine {  // определение внутреннего класса
            // поля и методы
            public void launch() {
               System.out.println("Запуск двигателя");
            }
      }// конец объявления внутреннего класса
      public void init() {// метод внешнего класса
        // объявление объекта внутреннего класса
            Engine eng = new Engine();
            eng.launch();
      }
}
```

При таком объявлении объекта внутреннего класса **Engine** в методе внешнего класса **Ship** нет реального отличия от использования какого-либо другого внешнего класса, кроме объявления внутри класса **Ship**. Использование объекта внутреннего класса вне своего внешнего класса возможно только при наличии доступа (видимости) и при объявлении ссылки в виде:

```
Ship.Engine obj = new Ship().new Engine();
```

Основное отличие от внешнего класса состоит в больших возможностях ограничения видимости внутреннего класса по сравнению с обычным внешним классом. Внутренний класс может быть объявлен как **private**, что обеспечивает его полную невидимость вне класса-владельца и надежное сокрытие реализации. В этом случае ссылку **obj**, приведенную выше, объявить было бы нельзя. Создать объект такого класса можно только в методах и логических блоках внешнего класса. Использование **protected** позволяет получить доступ к внутреннему классу для класса в другом пакете, являющегося суперклассом внешнего класса.

После компиляции объектный модуль, соответствующий внутреннему классу, получит имя **Ship$Engine.class**.

Методы внутреннего класса имеют прямой доступ ко всем полям и методам внешнего класса, в то же время внешний класс может получить доступ к содержимому внутреннего класса только после создания объекта внутреннего класса. Внутренние классы не могут содержать статические атрибуты и методы, кроме констант (**final static**). Внутренние классы имеют право наследовать другие классы, реализовывать интерфейсы и выступать в роли объектов наследования. Допустимо наследование следующего вида:

```
public class WarShip extends Ship {
     protected class SpecialEngine extends Engine {}
}
```

Если внутренний класс наследуется обычным образом другим классом (после **extends** указывается **ИмяВнешнегоКласса.ИмяВнутреннегоКласса**), то он теряет доступ к полям своего внешнего класса, в котором он был объявлен.

```
public class Motor extends Ship.Engine {
     public Motor(Ship obj) {
          obj.super();
     }
}
```

В данном случае конструктор класса **Motor** должен быть объявлен с параметром типа **Ship**, что позволит получить доступ к ссылке на внутренний класс **Engine**, наследуемый классом **Motor**.

Внутренние классы позволяют окончательно решить проблему множественного наследования, когда требуется наследовать свойства нескольких классов.

При объявлении внутреннего класса могут использоваться модификаторы **final**, **abstract**, **private**, **protected**, **public**.

Простой пример практического применения взаимодействия класса-владельца и внутреннего нестатического класса проиллюстрирован на следующем примере.

```
/* пример # 11 : взаимодействие внешнего и внутреннего классов : Student.java :
AnySession.java */
package chapt06;

public class Student {
     private int id;
     private ExamResult[] exams;

     public Student(int id) {
          this.id = id;
     }

     private class ExamResult {// внутренний класс
          private String name;
          private int mark;
          private boolean passed;

          public ExamResult(String name) {
               this.name = name;
               passed = false;
          }
          public void passExam() {
               passed = true;
          }
          public void setMark(int mark) {
               this.mark = mark;
          }
```

```
        public int getMark() {
             return mark;
        }
        public int getPassedMark() {
             final int PASSED_MARK = 4;// «волшебное» число
             return PASSED_MARK;
        }
        public String getName() {
             return name;
        }
        public boolean isPassed() {
             return passed;
        }
    } // окончание внутреннего класса

public void setExams(String[] name, int[] marks) {
            if (name.length != marks.length)
                  throw new IllegalArgumentException();
            exams = new ExamResult[name.length];
            for (int i = 0; i < name.length; i++) {
                  exams[i] = new ExamResult(name[i]);
                  exams[i].setMark(marks[i]);
      if (exams[i].getMark() >= exams[i].getPassedMark())
                        exams[i].passExam();
            }
public String toString() {
            String res = "Студент: " + id + "\n";
      for (int i = 0; i < exams.length; i++)
            if (exams[i].isPassed())
            res += exams[i].getName() + " сдал \n";
      else
            res += exams[i].getName() + " не сдал \n";

      return res;
      }
}
package chapt06;

public class AnySession {
     public static void main(String[] args) {
          Student stud = new Student(822201);
          String ex[] = {"Механика","Программирование"};
          int marks[] = { 2, 9 };
          stud.setExams(ex, marks);
          System.out.println(stud);
     }
}
```

В результате будет выведено:

```
Студент: 822201
Механика не сдал
Программирование сдал
```

Внутренний класс определяет сущность предметной области "результат экзамена" (класс **ExamResult**), которая обычно непосредственно связана в информационной системе с объектом класса **Student**. Класс **ExamResult** в данном случае определяет только методы доступа к своим атрибутам и совершенно невидим вне класса **Student**, который включает методы по созданию и инициализации массива объектов внутреннего класса с любым количеством экзаменов, который однозначно идентифицирует текущую успеваемость студента.

Внутренний класс может быть объявлен также внутри метода или логического блока внешнего класса. Видимость такого класса регулируется областью видимости блока, в котором он объявлен. Но внутренний класс сохраняет доступ ко всем полям и методам внешнего класса, а также ко всем константам, объявленным в текущем блоке кода. Класс, объявленный внутри метода, не может быть объявлен как **static**, а также не может содержать статические поля и методы.

```
/*пример # 12 : внутренний класс, объявленный внутри метода:
TeacherLogic.java*/
package chapt06;

public abstract class AbstractTeacher {
      private int id;
      public AbstractTeacher(int id) {
            this.id = id;
      }
      public abstract boolean excludeStudent(String name);
}
package chapt06;

public class Teacher extends AbstractTeacher {

      public Teacher(int id) {
            super(id);
      }
      public boolean excludeStudent(String name) {
            return false;
      }
}
package chapt06;

public class TeacherCreator {
      public TeacherCreator(){}

      public AbstractTeacher createTeacher(int id) {
            // объявление класса внутри метода
         class Dean extends AbstractTeacher {
```

```
                Dean(int id){
                      super(id);
                }
          public boolean excludeStudent(String name) {
                      if (name != null) {
                      // изменение статуса студента в базе данных
                            return true;
                      }
                      else return false;
                }
          }// конец внутреннего класса

          if (isDeanId(id))
                return new Dean(id);
          else return new Teacher(id);
    }
    private static boolean isDeanId(int id) {
          // проверка декана из БД или
          return (id  == 777);
    }
}
package chapt06;

public class TeacherLogic {
    public static void excludeProcess(int deanId,
                String name) {
          AbstractTeacher teacher =
          new TeacherCreator().createTeacher(deanId);

          System.out.println("Студент: " + name
          + " отчислен:" + teacher.excludeStudent(name));
    }
    public static void main(String[] args) {
          excludeProcess(700, "Балаганов");
          excludeProcess(777, "Балаганов");
    }
}
```

В результате будет выведено:

```
Студент: Балаганов отчислен:false
Студент: Балаганов отчислен:true
```

Класс **Dean** объявлен в методе **createTeacher(int id)**, и соответственно объекты этого класса можно создавать только внутри этого метода, из любого другого места внешнего класса внутренний класс недоступен. Однако существует возможность получить ссылку на класс, объявленный внутри метода, и использовать его специфические свойства. При компиляции данного кода с внутренним классом ассоциируется объектный модуль со сложным именем **TeacherCreator$1Dean**, тем не менее однозначно определяющим связь между внешним и внутренним классами. Цифра **1** в имени говорит о том, что в других методах класса могут быть объявлены внутренние классы с таким же именем.