Класс, вложенный в интерфейс, по умолчанию статический. На него не накладывается никаких особых ограничений, и он может содержать поля и методы как статические, так и нестатические.

```
/* пример # 14 : класс вложенный в интерфейс: Faculty.java : University.java */
package chapt06;

public interface University {
      int NUMBER_FACULTY = 20;

      class LearningDepartment {// static по умолчанию
            public int idChief;

            public static void assignPlan(int idFaculty) {
                  // реализация
            }
            public void acceptProgram() {
                  // реализация
            }
      }
}
```

Такой внутренний класс использует пространство имен интерфейса.

Анонимные (anonymous) классы

Анонимные (безымянные) классы применяются для придания уникальной функциональности отдельно взятому объекту для обработки событий, реализации блоков прослушивания и т.д. Можно объявить анонимный класс, который будет расширять другой класс или реализовывать интерфейс при объявлении одного, единственного объекта, когда остальным объектам этого класса будет соответствовать реализация метода, определенная в самом классе. Объявление анонимного класса выполняется одновременно с созданием его объекта посредством оператора **new**.

Анонимные классы эффективно используются, как правило, для реализации (переопределения) нескольких методов и создания собственных методов объекта. Этот прием эффективен в случае, когда необходимо переопределение метода, но создавать новый класс нет необходимости из-за узкой области (или одноразового) применения метода.

Конструкторы анонимных классов нельзя определять и переопределять. Анонимные классы допускают вложенность друг в друга, что может сильно запутать код и сделать эти конструкции непонятными.

```
/* пример # 15 : анонимные классы: TypeQuest.java: RunnerAnonym.java */
package chapt06;

public class TypeQuest {
      private int id = 1;

      public TypeQuest() {
      }
```

```
      public TypeQuest(int id) {
            this.id = id;
      }
      public void addNewType() {
            // реализация
            System.out.println(
            "добавлен вопрос на соответствие");
      }
}
package chapt06;

public class RunnerAnonym {
      public static void main(String[] args) {
       TypeQuest unique = new TypeQuest() {// анонимный класс #1
                  public void addNewType() {
                        // новая реализация метода
                        System.out.println(
                  "добавлен вопрос со свободным ответом");
                  }
            };// конец объявления анонимного класса
            unique.addNewType();

            new TypeQuest(71) {// анонимный класс #2
                  private String name = "Drag&Drop";

                  public void addNewType() {
                        // новая реализация метода #2
            System.out.println("добавлен " + getName());
                  }
                  String getName() {
                        return name;
                  }
            }.addNewType();

            TypeQuest standard = new TypeQuest(35);
            standard.addNewType();
      }
}
```

В результате будет выведено:

```
добавлен вопрос со свободным ответом
добавлен Drag&Drop
добавлен вопрос на соответствие
```

При запуске приложения происходит объявление объекта **unique** с применением анонимного класса, в котором переопределяется метод **addNewType()**. Вызов данного метода на объекте **unique** приводит к вызову версии метода из анонимного класса, который компилируется в объектный модуль с именем **RunnerAnonym$1**. Процесс создания второго объекта с анонимным типом применяется в программировании значительно чаще, особенно при реализации клас-

сов-адаптеров и реализации интерфейсов в блоках прослушивания. В этом же объявлении продемонстрирована возможность объявления в анонимном классе полей и методов, которые доступны объекту вне этого класса.

Для перечисления объявление анонимного внутреннего класса выглядит несколько иначе, так как инициализация всех элементов происходит при первом обращении к типу. Поэтому и анонимный класс реализуется только внутри объявления типа **enum**, как это сделано в следующем примере.

```
/* пример # 16 : анонимный класс в перечислении : EnumRunner.java */
package chapt06;

enum Shape {
      RECTANGLE, SQUARE,
      TRIANGLE {// анонимный класс
            public double getSquare() {// версия для TRIANGLE
                  return a*b/2;
            }
      };
      public double a, b;

      public void setShape(double a, double b){
            if ((a<=0 || b<=0) || a!=b && this==SQUARE)
                  throw new IllegalArgumentException();
            else
                  this.a = a;
                  this.b = b;
      }
      public double getSquare(){// версия для RECTANGLE и SQUARE
            return a * b;
      }
      public String getParameters() {
            return "a=" + a + ", b=" + b;
      }
}
public class EnumRunner {
      public static void main(String[] args) {
            int i = 4;
            for (Shape f : Shape.values()) {
                  f.setShape(3, i--);
      System.out.println(f.name()+"-> " + f.getParameters()
                        + " площадь= " + f.getSquare());
                  }
      }
}
```

В результате будет выведено:

```
RECTANGLE-> a=3.0, b=4.0 площадь= 12.0
SQUARE-> a=3.0, b=3.0 площадь= 9.0
TRIANGLE-> a=3.0, b=2.0 площадь= 3.0
```

Объектный модуль для такого анонимного класса будет скомпилирован с именем **Shape$1**.