Вложенные (nested) классы

Если не существует необходимости в связи объекта внутреннего класса с объектом внешнего класса, то есть смысл сделать такой класс статическим.

Вложенный класс логически связан с классом-владельцем, но может быть использован независимо от него.

При объявлении такого внутреннего класса присутствует служебное слово **static**, и такой класс называется вложенным (nested). Если класс вложен в интерфейс, то он становится статическим по умолчанию. Такой класс способен наследовать другие классы, реализовывать интерфейсы и являться объектом наследования для любого класса, обладающего необходимыми правами доступа. В то же время статический вложенный класс для доступа к нестатическим членам и методам внешнего класса должен создавать объект внешнего класса, а напрямую имеет доступ только к статическим полям и методам внешнего класса. Для создания объекта вложенного класса объект внешнего класса создавать нет необходимости. Подкласс вложенного класса не способен унаследовать возможность доступа к членам внешнего класса, которыми наделен его суперкласс.

```
/* пример # 13 : вложенный класс: Ship.java : RunnerShip.java */
package chapt06;

public class Ship {
      private int id;
      // abstract, final, private, protected - допустимы
      public static class LifeBoat {
            public static void down() {
                  System.out.println("шлюпки на воду!");
            }
            public void swim() {
                  System.out.println("отплытие шлюпки");
            }
      }
}
package chapt06;

public class RunnerShip {
      public static void main(String[] args) {
            // вызов статического метода
            Ship.LifeBoat.down();
            // создание объекта статического класса
            Ship.LifeBoat lf = new Ship.LifeBoat();
            // вызов обычного метода
            lf.swim();
      }
}
```

Статический метод вложенного класса вызывается при указании полного относительного пути к нему. Объект **lf** вложенного класса создается с использованием имени внешнего класса без вызова его конструктора.

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() {
      }
```