АБСТРАКТНЫЕ КЛАССЫ, ИНТЕРФЕЙСЫ. ВЛОЖЕННЫЕ КЛАССЫ, АНОНИМНЫЕ,

Абстрактные классы

Класс объявленный со спецификатором abstract

Может содержать абстрактные методы (методы без реализации).

```
abstract class A {
    abstract void m();
}
```

Свойства абстрактного класса

- Нельзя создать экземпляр, но можно объявить переменную данного типа
- Может иметь конструкторы
- Может содержать обычные методы

Предназначение абстрактных классов

Интерфейс к семейству классов.

База для реализации полиморфизма.

Неабстрактные потомки обязаны реализовать абстрактные методы.

Абстрактный метод

Метод, который не содержит реализации.

Обязан быть объявлен со спецификатором

abstract

```
public abstract void m();
```

Вложенные классы

- Элементы класса
  - статические
  - нестатические
- Локальные
  - анонимные
  - с указанием имени

Примеры классов

```
class A { // класс верхнего уровня
    void m(SomeClass p) {…}
    class B {} // класс элемент класса
    void m() {
     class C {} // C – локальный класс
     m(new SomeClass() {    // ан. класс
     });
    }
}
```

Классы - элементы классов

Могут иметь модификаторы/спецификаторы:

- abstract
- static
- final
- private/protected/default/public

Локальные классы

Классы, объявленные внутри методов, конструкторов, блоках инициализации.

Могут быть: abstract, final

Уровень доступа - default, по умолчанию, ограничен телом блока, в котором объявлен класс.

Анонимные классы

Класс, который не имеет имени.

Всегда расширяет класс или реализует интерфейс.

Используется при создании объектов.

Пример анонимного класса

```
class T {}

T t = new T() {
    void m() {...}
}
```

t - переменная типа T, кот. ссылается на экземпляр анонимного класса, наследованного от T.

Свойства внутренних классов

- Не могут объявлять статических полей (кроме констант), методов и классов (но могут наследовать их.
- Имеют доступ к элементам внешнего класса.
- Имеют доступ к локальным переменным и параметрам метода (они должны быть объявлены как final).

Создание объектов внутреннего класса

(нестатические элементы классов)

Расширенный синтаксис оператора new

```
class A {
    class B {}
}
A a = new A();
A.B b = a.new B();
```

Создание объектов вложенных статических классов

```
class A {
    static class B {}
}

A.B b = new A.B();
```

Доступ к объекту внешнего класса (this) из внутреннего

```
class A {
    private int x;
    class B {
     int x = A.this.x;
    }
}
```

Имя файла с байт-кодом вложенного класса

Для каждого класса компилятор (javac) создает отдельный файл класса.

Outer$Inner.class

Интерфейсы

Определяют границы взаимодействия между объектами.

Определяют абстракцию, реализацию которой предоставляет имплементирующая интерфейс сторона.

Использование интерфейсов

- Класс может реализовывать интерфейс
- Можно объявить интерфейсную переменную
- Интерфейс может наследовать несколько других интерфейсов

Элементы интерфейса

- поля (public static final)
- методы (public abstract)
- интерфейсы (public static)
- классы (public static)

Указанные модификаторы и спецификаторы можно не ставить.

Поля интерфейса

- константы (final)
- статические (static)
- публичные (public)
- должны быть проинициализированы при объявлении

Методы интерфейса

- абстрактные (abstract)
- публичные (public)

Вложенные классы и интерфейсы- элементы интерфейсов

- Статические

Множественное наследование интерфейсов

```
interface Interf extends Interf1, Interf2 {
    …
}
```

Реализация интерфейса

```
class A implements Interf1, Interf2 {
    …
}
```

Реализация интерфейсов и расширение класса

```
class A extends B implements Interf1, Interf2 {
    …
}
```

Оператор instanceof для интерфейсов

```
interface I {}
class A implements I {}
...
boolean f = new A() instanceof I; // f = true
```

Частичная реализация интерфейса

```
interface I {
    void m();
    void m2();
}
// ошибка компиляции, A is not abstract!
class A implements I {
    public void m() {}
}
```