Конспект по JLS 8

Вложенные классы (inner, local и anonymous)

Inner class может быть nested:

- enum class
- record class
- public class
- abstract class
- sealed class
- final class
- различие классов и интерфейсов
- generic class
- аннотации (посмотреть какие)

Члены класса

- поля (instance и class), могут быть `final` (нельзя менять значение), допускается инициализатор
- методы (`final | abstract | native | synchronized`); перегрузка; абстрактные методы только в абстрактных классах; нельзя скрывать абстрактные методы
- классы (static или inner)
- интерфейсы
- инициализаторы
- статические инициализаторы
- конструкторы (можно перегружать)

Члены класса:

- декларированные
- наследованные

Access modifiers

- public
- protected
- private

Наследование

Можно скрывать члены-интерфейсы и члены-классы, а также скрывать методы, реализовывать их и делать перегрузку (имеются в виду члены, объявленные в реализованных интерфейсах и наследованных классах).

Синтаксис

ClassDeclaration:

- NormalClassDeclaration
- EnumDeclaration
- RecordDeclaration

NormalClassDeclaration:

```
{ClassModifier} class TypeIdentifier [TypeParameters]
[ClassExtends] [ClassImplements] [ClassPermits] ClassBody
```

Нельзя называть классы так же, как окружающие их другие классы/интерфейсы.

ClassModifier:

```
(one of)
Annotation public protected private
abstract static final sealed non-sealed
```

- public — только для top-level и классов-членов, не для локальных или анонимных
- protected | private — только для классов-членов
- static — только для классов-членов

Нельзя повторять модификаторы или использовать несколько из списка `private|protected|public`. Аналогично с `sealed|non-sealed|final`. Нельзя совмещать `private + final`.

- non-sealed нужен, чтобы можно было свободно наследовать класс-наследник sealed
- enum class автоматически final или sealed
- record class автоматически final
- local enum и record class — автоматически static
- local и anonymous class — не могут быть static

TypeParameters

```
<TypeParameterList>
TypeParameterList:
TypeParameter {, TypeParameter}

TypeParameter:
{TypeParameterModifier} TypeIdentifier [TypeBound]
TypeParameterModifier:
Annotation
TypeBound:
extends TypeVariable
extends ClassOrInterfaceType {AdditionalBound}
AdditionalBound:
& InterfaceType
```

- Нельзя делать циклические зависимости
- getClass() даёт одинаковый результат для всех вариантов параметризации одного класса
- generic class не может реализовывать Throwable

ClassExtends

```
extends ClassType
```

- enum может наследоваться только от enum
- record — только от record

ClassImplements

```
implements InterfaceTypeList
```

```
InterfaceTypeList:
InterfaceType {, InterfaceType}
```

- Нельзя реализовывать две реализации (generic) одного интерфейса
- Если есть методы сразу в двух интерфейсах, то они реализуются оба

ClassPermits

```
permits TypeName {, TypeName}
```

- `permits` можно указывать только для sealed-классов
- в списке явно перечисляются наследники

ClassBody

```
{ {ClassBodyDeclaration} }
ClassBodyDeclaration:
ClassMemberDeclaration
InstanceInitializer
StaticInitializer
ConstructorDeclaration
ClassMemberDeclaration:
FieldDeclaration
MethodDeclaration
ClassDeclaration
InterfaceDeclaration
```

Object

Методы класса `Object`:

- `clone()` — возвращает копию объекта
- `equals(Object obj)` — проверка равенства
- `finalize()` — `deprecated`, будет удалён
- `getClass()` — возвращает runtime-класс объекта
- `hashCode()` — возвращает hash
- `notify()`, `notifyAll()`, `wait()` и варианты — синхронизация

- `toString()` — строковое представление

Контракты:

- equals/hashCode — рефлексивность, симметричность, транзитивность, консистентность, неравенство с null
- clone — отдельный контракт

`https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/Object.html`

Class<T>

```
public final class Class<T>
extends Object
implements Serializable, GenericDeclaration, Type,
AnnotatedElement, TypeDescriptor.OfField<Class<?>>, Constable
```

Методы и свойства `Class<T>` приведены в спецификации.

Конструкторы и инициализаторы

- не наследуются, как и private-члены

Тип метода определяется:

- типовыми параметрами
- типами параметров
- возвращаемым типом
- спецификацией `throw`

FieldDeclaration

```
{FieldModifier} UnannType VariableDeclaratorList ;
...
FieldModifier:
(one of)
Annotation public protected private
static final transient volatile
```

- `transient` — поле не входит в persistent-состояние
- `volatile` — согласованное состояние для всех потоков (без перестановки операций); не может быть `final`

MethodDeclaration

```
{MethodModifier} MethodHeader MethodBody
...
MethodModifier:
(one of)
Annotation public protected private
abstract static final synchronized native
```

- нельзя создавать varargs-параметры для non-reifiable типов (если не использовать `@SafeVarargs` или `@SuppressWarnings`)

InstanceInitializer

```
Block
```

StaticInitializer

```
static Block
```

ConstructorDeclaration

```
{ConstructorModifier} ConstructorDeclarator [Throws] ConstructorBody
...
ConstructorInvocation:
[TypeArguments] this ( [ArgumentList] ) ;
[TypeArguments] super ( [ArgumentList] ) ;
ExpressionName . [TypeArguments] super ( [ArgumentList] ) ;
Primary . [TypeArguments] super ( [ArgumentList] ) ;
```

Конструктор по умолчанию.

Остановились на **enum**.