Оглавление

[Java Core 11](#_Toc41572414)

[статический метод переопределен или перегружен? 11](#_Toc41572415)

[нестатические методы перегрузить статические? 11](#_Toc41572416)

[сузить уровень доступа/тип возвращаемого значения при переопределении метода? 11](#_Toc41572417)

[при переопределении метода изменить: модификатор доступа, возвращаемый тип, тип аргумента или их количество, имена аргументов или их порядок; убирать, добавлять, изменять порядок следования элементов секции throws? 11](#_Toc41572418)

[доступ к переопределенным методам родительского класса? 12](#_Toc41572419)

[объявить метод абстрактным и статическим? 12](#_Toc41572420)

[разница между членом экземпляра класса и статическим членом класса? 12](#_Toc41572421)

[где разрешена инициализация статических/нестатических полей? 13](#_Toc41572422)

[вложенные классы 13](#_Toc41572423)

[«статический класс» 14](#_Toc41572424)

[особенности вложенных классов, разница статических и внутренних 15](#_Toc41572425)

[Что такое «локальный класс»? Каковы его особенности? 15](#_Toc41572426)

[Что такое «анонимные классы»? Где они применяются? 16](#_Toc41572427)

[Каким образом из вложенного класса получить доступ к полю внешнего класса? 18](#_Toc41572428)

[Heap и Stack памяти, разница 18](#_Toc41572429)

[утверждение: примитивные типы данных хранятся в стеке, а экземпляры ссылочных типов — в куче? 19](#_Toc41572430)

[Каким образом передаются переменные в методы, по значению или по ссылке? 19](#_Toc41572431)

[String неизменяемый и финализированный класс? 19](#_Toc41572432)

[char[] предпочтительнее String для хранения пароля? 20](#_Toc41572433)

[Почему строка является популярным ключом в HashMap в Java? 20](#_Toc41572434)

Java Core

статический метод переопределен или перегружен?

перегружен, без переопределения (раннее связывание)

обращение к статическим полям/методам

нестатические методы перегрузить статические?

да, 2 разных метода (статический — классу, нестатический – объекту)

сузить уровень доступа/тип возвращаемого значения при переопределении метода?

при переопределении метода изменить: модификатор доступа, возвращаемый тип, тип аргумента или их количество, имена аргументов или их порядок; убирать, добавлять, изменять порядок следования элементов секции throws?

модификаторы — только расширять

1. тип в сторону сужения

2. изменение типа, количества, порядка следования аргументов — перегрузка метода

3. throws можно не указывать (останутся как у родительского метода), либо наследники или исключения RuntimeException;

порядок не важен

доступ к переопределенным методам родительского класса?

ключевое слово super (к члену родительского класса (метод или поле), если они не private)

super.method();

объявить метод абстрактным и статическим?

Illegal combination of modifiers: ‘abstract’ and ‘static’

abstract — в другом классе, static — по имени класса.

разница между членом экземпляра класса и статическим членом класса?

статические члены:

поля и методы принадлежат классу и доступ без создания объекта; поля инициализируются при инициализации класса.

ограничения:

1. могут обращаться только к статическим методам и должны обращаться тольо к статическим переменным.

2. не могут ссылаться на члены типа this или super

члены экземпляра класса:

поля принадлежат конкретному объекту (разные значения); вызов метода только после создания объекта

где разрешена инициализация статических/нестатических полей?

Статические поля: при объявлении, в статическом или нестатическом блоке инициализации

Нестатические поля: при объявлении, в нестатическом блоке инициализации или в конструкторе.

4.вложенный класс (статический вложенный, простой внутренний, локальный, анонимный)

вложенные классы

определен внутри другого класса, обслуживает внешний класс

стать классом верхнего уровня

имеют доступ ко всем полям и методам (не наоборот) (приводит к нарушению инкапсуляции)

1. статический вложенный класс

2. простой внутренний класс

3. локальный класс

4. анонимный класс

2‑4 — внутренние (ассоциируются не с классом, а с экземпляром)

рекомендации к применению

1. статический — ссылка на внешний класс не требуется

2 нестатический:

- должен быть виден за пределами одного метода

- слишком длинный для того, чтобы его можно было удобно разместить в границах одного метода и каждому экземпляру такого класса необходима ссылка на включающий его экземпляр

3. локальный класс — необходим только внутри какого‑то метода и требуется создавать экземпляры этого класса только в этом методе

4. анонимным классом — если к тому же применение класса сводится к использованию лишь в одном месте и уже существует тип, характеризующий этот класс

«статический класс»

вложенный класс с ключевым словом static (класс верхнего уровня static — неприменим)

особенности вложенных классов, разница статических и внутренних

1. обращаются ко всем членам

2. объект статического вложенного класса создается без объекта внешнего

3. из объекта статического вложенного обращается к нестатическим членам внешнего через ссылку на экземпляр

4. обычные вложенные классы не могут содержать статических методов, блоков инициализации и классов, статические — да

•5. в объекте обычного вложенного класса — ссылка на объект внешнего, доступ Outer.this; внутри статического — нет

Что такое «локальный класс»? Каковы его особенности?

Локальный класс (Local inner class) — это вложенный класс, который может быть декларирован в любом блоке, в котором разрешается декларировать переменные. Как и простые внутренние классы (Member inner class) локальные классы имеют имена и могут использоваться многократно. Как и анонимные классы, они имеют окружающий их экземпляр только тогда, когда применяются в нестатическом контексте.

Локальные классы имеют следующие особенности:

• Видны только в пределах блока, в котором объявлены;

• Не могут быть объявлены как private/public/protected или static;

• Не могут иметь внутри себя статических объявлений методов и классов, но могут иметь финальные статические поля, проинициализированные константой;

• Имеют доступ к полям и методам обрамляющего класса;

• Могут обращаться к локальным переменным и параметрам метода, если они объявлены с модификатором final.

Что такое «анонимные классы»? Где они применяются?

Это вложенный локальный класс без имени, который разрешено декларировать в любом месте обрамляющего класса, разрешающем размещение выражений. Создание экземпляра анонимного класса происходит одновременно с его объявлением. В зависимости от местоположения анонимный класс ведет себя как статический либо как нестатический вложенный класс — в нестатическом контексте появляется окружающий его экземпляр.

Анонимные классы имеют несколько ограничений:

• Их использование разрешено только в одном месте программы — месте его создания;

• Применение возможно только в том случае, если после порождения экземпляра нет необходимости на него ссылаться;

• Реализует лишь методы своего интерфейса или суперкласса, т. е. не может объявлять каких либо новых методов, так как для доступа к ним нет поименованного типа.

Анонимные классы обычно применяются для:

• создания объекта функции (function object), например реализация интерфейса Comparator;

• создания объекта процесса (process object), такого как экземпляры классов Thread, Runnable и подобных;

• в статическом методе генерации;

• инициализации открытого статического поля final, которое соответствует сложному перечислению типов, когда для каждого экземпляра в перечислении требуется отдельный подкласс.

Каким образом из вложенного класса получить доступ к полю внешнего класса?

Статический вложенный класс имеет прямой доступ только к статическим полям обрамляющего класса.

Простой внутренний класс, может обратиться к любому полю внешнего класса напрямую. В случае, если у вложенного класса уже существует поле с таким же литералом, то обращаться к такому полю следует через ссылку на его экземпляр. Например: Outer.this.field.

Heap и Stack памяти, разница

стек (Stack)

- оперативная память (RAM);

- при вызове метода новый блок (примитивы и ссылки на другие объекты) и перестает работать после окончания метода;

- схема LIFO.

Различия:

- размер

- скорость;

- исключения;

- опции определения размера.

утверждение: примитивные типы данных хранятся в стеке, а экземпляры ссылочных типов — в куче?

Не совсем. Примитивное поле экземпляра класса хранится не в стеке, а в куче. Любой объект (все, что явно или неявно создается при помощи оператора new) хранится в куче.

Каким образом передаются переменные в методы, по значению или по ссылке?

В Java параметры всегда передаются только по значению, что определяется как «скопировать значение и передать копию». С примитивами это будет копия содержимого. Со ссылками — тоже копия содержимого, т. е. копия ссылки. При этом внутренние члены ссылочных типов через такую копию изменить возможно, а вот саму ссылку, указывающую на экземпляр — нет.

String неизменяемый и финализированный класс?

Преимущества:

- больше свободного места в Heap (разные строковые переменные указывают на одну в пуле);

- иначе серьезная угроза безопасности приложения;

- избежать синхронизации (одна строка в разных потоках)

- строки используются classloader и неизменность обеспечивает правильность загрузки класса;

- hashCode() кэшируется один раз (ключ в HashMap — обработка быстрее).

char[] предпочтительнее String для хранения пароля?

при использовании массива символов имеется возможность очистить его сразу по окончанию работы с паролем, а строка остается в пуле до работы сборщиком мусора

Почему строка является популярным ключом в HashMap в Java?

строки неизменяемы — хэш‑код вычисляется и кэшируется один раз (в момент создания); в качестве ключа HashMap будут обрабатываться быстрее