Afli

41 уровень

Санкт-Петербург

21.11.2016 17468 13

Уровень 35. Ответы на вопросы к собеседованию по теме уровня

Вопросы/дополнения/критика приветствуются.

1. **Какие системы контроля версий вы знаете?**

   Git, SVN, Bazaar, Mercurial

2. **Чем отличаются SVN и Git?**

   1. GIT — распределенная СКВ, а SVN — нет. Другими словами, если есть несколько разработчиков работающих с репозиторием у каждого на локальной машине будет ПОЛНАЯ копия этого репозитория. Разумеется есть и центральная машина, с которой можно клонировать репозиторий. Это напоминает SVN. Основной плюс Git в том, что если вдруг у вас нет доступа к интернету, сохраняется возможность работать с репозиторием. Потом только один раз сделать синхронизацию и все остальные разработчики получат полную историю.

   2. GIT сохраняет метаданные изменений, а SVN целые файлы. Это экономит место и время.

3. **Что такое GitHub? У вас есть проекты на GitHub?**

   GitHub — веб-сервис хостинга проектов с использованием системы контроля версий git, а также как социальная сеть для разработчиков. Пользователи могут создавать неограниченное число репозиториев, для каждого из которых предоставляется wiki, система issue tracking-a, есть возможность проводить code review и т.п. Кроме Git, сервис

4. **Зачем нужны системы контроля версий?**

   СКВ дает возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внес в код какую-то ошибку, и т.п.. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить.

5. **Что такое generic? Как они реализованы в Java?**

   Generics — это параметризованные типы. С их помощью можно объявлять классы, интерфейсы и методы, где тип данных указан в виде параметра. Обобщения добавили в язык безопасность типов.

   Пример реализации:

```
class MyClass<T>{
  T obj;
  public MyClass(T obj){
    this.obj = obj;
  }
}
class MyClass<T>
```

В угловых скобках используется **T** — имя параметра типа. Это имя используется в качестве заполнителя, куда будет подставлено имя реального типа, переданного классу `MyClass` при создании реальных типов. То есть параметр типа `T` применяется в классе всякий раз, когда требуется параметр типа. Угловые скобки указывают, что параметр может быть обобщен. Сам класс при этом называется обобщенным классом или параметризованным типом.

Далее тип `T` используется для объявления объекта по имени `obj`:

```
T obj;
```

Вместо `T` подставится реальный тип, который будет указан при создании объекта класса `MyClass`. Объект `obj` будет объектом типа, переданного в параметре типа `T`. Если в параметре `T` передать тип `String`, то экземпляр `obj` будет иметь тип `String`.

Рассмотрим конструктор `MyClass()`:

```
public MyClass(T obj){
  this.obj = obj;
}
```

Параметр `obj` имеет тип `T`. Это значит, что реальный тип параметра `obj` определяется типом, переданным параметром типа `T` при создании объекта класса `MyClass`.

Параметр типа `T` также может быть использован для указания типа возвращаемого значения метода.

В именах переменных типа принято использовать заглавные буквы. Обычно для коллекций используется буква `E`, буквами `K` и `V` — типы ключей и значение (Key/Value), а буквой `T` (и при необходимости буквы `S` и `U`) — любой тип.

Обобщения работают только с объектами. Поэтому нельзя использовать в качестве параметра элементарные типы вроде `int` или `char`.

*Так же считаю нужным упомянуть generic методы. Это методы вида:

модификаторы <T, ...> возвращаемыйТип имяМетода(T t, ...)

Как я понял, если в качестве типа в сигнатуре метода используются параметры, необходимо перед типом возвращаемого

- Обобщения (Generic)
- Дженерики (Java, обучающая статья)

6. **Что такое стирание типов?**

   Внутри класса-дженерика не хранится информация о его типе параметре. Это и называется стиранием типов. На стадии компиляции происходит приведение объекта класса к типу, который был указан при объявлении.

   Пример:

Код с generic'ами	Что происходит на самом деле

`List<String> strings = new ArrayList<String>();`
`strings.add("abc");`
`strings.add("abc");`
`strings.add( 1); // тут ошибка компиляции`
	`List strings = new ArrayList();`
	
	`strings.add((String)"abc");`
	`strings.add((String)"abc");`
	`strings.add((String) 1); //ошибка компиляции`

7. **Что такое wildcard?**

   Wildcard — это дженерик вида <?>, что означает, что тип может быть чем угодно. Используется, например, в коллекциях, где для всех коллекций базовым типом является Collection<?>.

   Полезная ссылка: Теория и практика Java. Эксперименты с generic-методами

8. **Расскажите про extends и super в Generic'ax?**

   Чтобы наложить ограничение на wildcard необходимо использовать конструкции типа:

   - `? extends SomeClass` — означает, что может быть использован любой класс-наследник SomeClass
   - `? super SomeClass` — означает, что может быть использован класс SomeClass, либо класс-родитель (или интерфейс) SomeClass

   Это называется bounded wildcard.

   Для того, чтобы определиться с выбором между `extends` и `super` был придуман метод PECS.

   Подробно про это можно прочитать по ссылке ниже: Использование generic wildcards для повышения удобства Java API

9. **Как использовать wildcard?**

   Пример использования wildcard:

```
1  List<?> numList = new ArrayList<Integer>();
```

   Вопрос я не понял, но в принципе использование wildcard’ов рассматривается в материалах по ссылкам выше.

10. **В чем отличие ArrayList и ArrayList<?>**

    Запись вида ArrayList называется raw type (обычный тип). Она эквивалентна записи вида ArrayList<T> и используется для обратной совместимости, т.к. до Java 1.5 не было дженерик коллекций. По возможности такой формы записи следует избегать.

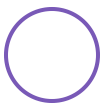
    ArrayList<?> является супертипом для ArrayList.

Введите текст комментария

Андрей Шевченко Junior Java Developer в **CUSTIS** 30 ноября 2020, 23:38

Про 10 вопрос не до конца понял, может кто пояснить, зачем wildcard <?> вообще нужен.
Прочел https://docs.oracle.com/javase/tutorial/java/generics/unboundedWildcards.html
и все равно не вижу смысла в этом <?>, т.к. в примерах по ссылке все бы работало и с raw types

Ответить +2



Даниил Salesforce Developer в **Viseven** MASTER 10 сентября 2019, 10:41

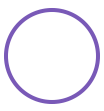
Несоответсвие в вопросе 2.2. Согласно вики:
Модель работы
Subversion — централизованная система (в отличие от распределённых систем, таких как Git или Mercurial), то есть данные хранятся в едином хранилище. Хранилище может располагаться на локальном диске или на сетевом сервере.

Работа в Subversion мало отличается от работы в других централизованных системах управления версиями. Клиенты копируют файлы из хранилища, создавая локальные рабочие копии, затем вносят изменения в рабочие копии и фиксируют эти изменения в хранилище. Несколько клиентов могут одновременно обращаться к хранилищу. Для совместной работы над файлами в Subversion преимущественно используется модель копирование — изменение — слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель блокирование — изменение — разблокирование.

При сохранении новых версий используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных.

При использовании доступа с помощью WebDAV также поддерживается прозрачное управление версиями — если любой клиент WebDAV открывает для записи и затем сохраняет файл, хранящийся на сетевом ресурсе, то автоматически создаётся новая версия.

Ответить +1



Ярослав Java Developer MASTER 4 июля 2018, 21:37

ArrayList list === ArrayList<Object> list; // То есть, туда можно положить вообще всё.
ArrayList<?> list === ArrayList<? extends Object> list; // Сюда вообще ничего нельзя положить.

Почему нельзя положить: у нас будет коллекция, тип которой может быть любым типом от Object. А это значит, что там может быть и коллекция String, и коллекция Number, и в такой анархии просто ничего нельзя положить, ибо мы можем положить String, а коллекция будет Number'ов.

Чтобы можно было положить, нужно использовать ? super Object, которая по действию будет эквивалентна просто сырому типу. Так как у нас коллекция будет этого же типа, который мы указали, или же типа родителя, то нам точно известно, что мы можем добавлять в коллекцию Object.

Более нормальным пример: ? super String. У нас коллекция может быть только List<Object> или List<String>. В любом случае, мы можем добавлять туда String.

Достаточно запомнить правило PECS: Producer - extends, Consumer - super.

Ответить +15

lichMax Уровень 40, Санкт-Петербург, Россия 12 июля 2017, 22:14

Ещё одно. Цитата из этих ответов:
В чем отличие ArrayList и ArrayList<?>

Запись вида ArrayList называется raw type (обычный тип). Она эквивалентна записи вида ArrayListи используется для обратной совместимости, т.к. до Java 1.5 не было дженерик коллекций. По возможности такой формы записи следует избегать.

ArrayList<?> является супертипом для ArrayList.

Как-то туманно написано, особенно это: «Запись вида ArrayList называется raw type (обычный тип). Она эквивалентна записи вида ArrayList». Либо опечатка, либо бездумный копипаст какой-нибудь.

Сам для себя пока нашёл одно отличие между ArrayList и ArrayList<?>: в список типа ArrayList можно вставить элемент любого типа, а в список типа ArrayList<?> — только null. А вот подстановка списков с конкретным типом переменных вместо параметров с данными типами — происходит одинаково. Кроме того, чтение данных из списков этих двух типов тоже проходит одинаково (и в обоих случаев это происходит без проблем).

Ответить +1

lichMax Уровень 40, Санкт-Петербург, Россия 12 июля 2017, 20:46

Как использовать wildcard?

Тоже не совсем понял вопрос. Но как я понял, здесь надо рассказать про весь синтаксис с использованием wildcard (то есть где используется такая контрукция, в каких местах в языке).

lichMax Уровень 40, Санкт-Петербург, Россия — 12 июля 2017, 20:42

Расскажите про extends и super в Generic'ах?

Дополнение: эти ключевые слова могут использовать не только вместе со знаком вопроса (в wildcard), но и с буквами (то есть в обычном параметрическом типе).

Ответить +2

lichMax Уровень 40, Санкт-Петербург, Россия — 12 июля 2017, 22:17

*могут использоваться

Ответить 0

LuneFox инженер по сопровождению в **BIFIT** EXPERT — 5 февраля, 19:50

Здесь есть кнопочка "редактировать", если что

Ответить 0

lichMax Уровень 40, Санкт-Петербург, Россия — 12 июля 2017, 20:40

Опять-таки, ответ на вопрос про WildCard: «Wildcard — это дженерик вида <?>». Дженерик это класс, который имеет параметрический тип (записываемый в угловых скобках). А wildcard, как я понял, это разновидность типа-параметра (но не дженерик).

Ответить +2

Константин Уровень 40, Харьков, Украина — 23 августа 2021, 13:09

Я считал что Generic - это параметризация, это не класс. Дженерик классы и методы - это параметризованные классы и методы! Я ошибаюсь?

Ответить +2

lichMax Уровень 40, Санкт-Петербург, Россия — 12 июля 2017, 20:32

Небольшие замечания по ответу на пятый вопрос. Там есть такое предложение: «Это имя используется в качестве заполнителя, куда будет подставлено имя реального типа, переданного классу MyClass при создании реальных типов.» Что значит «создание типов»? Вообще-то конкретный тип вместо типа-параметра подставляется при конкретном использовании класса или метода. Первое — это объявление перемменной, у которой в качестве типа выступает данный дженерик-класс, или наследование от данного обобщённого класса или интерфейса. Второе — это вызов метода.

И ещё, такое предложение: «Сам класс при этом называется обобщенным классом или параметризованным типом.» Параметрическим типом как раз является тип, указываемый в угловых скобках (его ещё называют типо-параметром).

Ответить 0

lichMax Уровень 40, Санкт-Петербург, Россия — 12 июля 2017, 20:16

Пункт «б» ответа на первый вопрос неверный. Правильный ответ такой:

«Git хранит „слепки“ файлов в сжатом виде (причём, если файл не менялся, то хранится ссылка на него), а SVN хранит изменения файлов.» Тут например есть про это. И вот ещё ссылки про сравнение этих двух СКВ: первая, вторая

Но если говорить вообще, то чёткого сравнения в интернете я и не нашёл. А некоторые статьи и высказывания, как я понял, уже устарели. Вот что у меня получилось по пунктам:

а) SVN — централизованная СКВ, а Git — распределённая СКВ. Соответственно, гит хранит все файлы и всю историю на локальном диске, SVN — же на сервере.

б) Git хранит «слепки» файлов в сжатом виде (причём, если файл не менялся, то хранится ссылка на него), а SVN хранит изменения файлов.

в) В Git коммит может иметь несколько родителей (разных веток) и несколько потомков (разных веток), а в SVN — нет. Соответственно, в SVN — линейная история коммитов, а в Git — нет.

г) В Git ветка — это просто указатель на какой-то коммит, в SVN — это полноценная копия всего репозитория. (?)

д) Git создают одну скрытую папку в корне проекта, в которой хранит всю свою служебнуюинформацию, а SVN в каждой папке проекта создаёт скрытую папку со своей служебной информацией. (?)

Последние два пункт с вопросами, потому что по ним несколько противоречивая информация (думаю, из-за разных версий SVN, которыми пользовались люди).

Ответить +3

lichMax Уровень 40, Санкт-Петербург, Россия — 12 июля 2017, 22:19

Ещё можно добавить такую ссылку к пункту «б» ответа на первый вопрос: ссылка

Ответить 0