

Большая задача: Class ArrayList

Java Collections
1 уровень, 15 лекция

ОТКРЫТА

- Привет, боец!
- Поздравляю тебя с повышением уровня квалификации. Нам нужны отчаянные парни.
- Уверен, у тебя есть еще много нерешенных задач. Самое время решить парочку из них!

Задача  Java Collections, 1 уровень, 15 лекция

ЗАКРЫТА

MEDIUM

Построй дерево(1)

★★★★☆

 ×14

Амиго, похоже ты уже достаточно окреп. Самое время проверить свои навыки в большой задаче! Сегодня реализуем свое дерево немного нестандартным способом(на базе ArrayList). Вводную информацию можешь получить используя свой любимый поисковик и текст ниже.

Открыть

Задача  Java Collections, 1 уровень, 15 лекция

ЗАКРЫТА

MEDIUM

Построй дерево(2)

★★★★☆

 ×14

Несмотря на то что наше дерево является потомком класса ArrayList, это не список в привычном понимании. В частности нам недоступны принимающие в качестве параметра индекс элемента. Такие методы необходимо переопределить и бросить новое исключение типа UnsupportedOperationException. Вот их список

Открыть

Задача  Java Collections, 1 уровень, 15 лекция

ЗАКРЫТА

MEDIUM

Построй дерево(3)

★★★★☆

 ×14

Класс описывающий дерево мы создали, теперь нужен класс описывающий тип элементов дерева: 1) В классе CustomTree создай вложенный статический параметризированный класс Entry<T> с модификатором доступа по умолчанию. 2) Обеспечь поддержку этим классом интерфейса Serializable. 3) Создай такие поля (мод

Открыть

Задача  Java Collections, 1 уровень, 15 лекция

РЕШЕНА

HARD

Построй дерево(4)

★★★★☆

 ×28

Любое дерево начинается с корня, поэтому не забудь в класс CustomTree добавить поле root типа Entry<String> с модификатором доступа по умолчанию. Инициируй его в конструкторе CustomTree, имя (elementName) не важно. Итак, основа дерева создана, пора тебе поработать немного самому. Вспомним как должно выглядеть наше дерево.

Открыть

Задача  Java Collections, 1 уровень, 15 лекция

РЕШЕНА

HARD

НАЧАТЬ ОБУЧЕНИЕ



Добавлять в дерево элементы мы можем, теперь займись удалением: необходимо реализовать метод remove(Object o), который будет удалять элемент дерева имя которого было полученного в качестве параметра.

Открыть

< Предыдущая лекция



− +107 +

Комментарии (302)

популярные новые старые

JavaCoder

Введите текст комментария

comrade_b Уровень 32, Амстердам, Нидерланды

15 июля, 17:24

Из продвинутых, кто глубоко погрузился в тему, к вам вопрос:
Где на практике сейчас (в 2022) используется бинарный поиск и бинарные деревья? Я про новые технологии, а не про использование ZX Spectrum.

Просто нам еще в Syntax Zero (или при "оттачивании навыка гугления") говорили, что бинарный поиск один из самых медленных. Смысл тогда в бинарных деревьях?

Ответить

− 0 +

tohik Уровень 40, Киев

8 июля, 14:47

вот это жесть конечно задачка... это пожалуй самая долгая моя задача в курсе

Ответить

− 0 +

Игорь РМ в Москва

10 мая, 12:48

В main ошибка: "The expected parent is 9. The actual parent is 1".
В заданном тексте решения прописано удаление 3, то есть, у 1 левый child будет свободен. Потом добавляется 16. При правильном решении его parentom станет 1, он и будет возвращён. Откуда авторы взяли возврат 9 при добавлении 16 после удаления 3, можно только догадываться.
Диего, ты трезв? :)

Ответить

− 0 +

On1k Уровень 37, Krasnogorsk, Russian Federation

30 июня, 12:03

Все правильно, если разобраться.
Если ты удаляешь элемент, это не значит что на его место можно сразу что-то добавлять.
Добавлять можно только в самый низ древа.
Но если так получилось, что в древо нельзя ничего добавить (то есть булевы переменные всех элементов равны false), то всем элементам нижнего уровня нужно разрешить создавать дочки.

Ответить

− 0 +

StringiD Престарелый студент в Жавараш

28 марта, 10:39

Мда... понять что от тебя хотят 99% дела.

Ответить

− +2 +

Дядя Богдан ♂ Механик с ключом на 24 в Клубе SKALA ♂

25 марта, 23:11

Пока единственная задача из рубрики больших задач, от которой получил удовольствие при решении.

Ответить

− 0 +

val Уровень 35, Москва

21 марта, 15:26

показалось что лишнего нагородил(170 строк)

НАЧАТЬ ОБУЧЕНИЕ

Onk

Уровень 37, Krasnogorsk, Russian Federation

30 июня, 12:05

...

148 строк)

Ответить

−

0

+

Eriniya

Уровень 33, Ярославль, Россия

16 марта, 13:54

...

А мне понравилось)

Ответить

−

0

+

Роман Кончалов

Уровень 28, Россия

EXPERT

19 января, 22:17

...

Если это ваше первое бинарное дерево, то лучше его решать рекурсивно, конечно, хоть это и сложнее без хорошего понимания рекурсии. Мне рекурсия по жизни порядком надоела, да и толку от неё в реальном продукте никогда нет: работает она быстрее чем циклы только в алгоритме слияния для быстрой сортировки. Для любопытных можете глянуть реализацию TreeMap: там одни циклы, никакой рекурсии, лол, кек

Ответить

−

+1

+

Andrey Chuev

Уровень 36, Москва, Russian Federation

13 апреля, 20:57

...

При использовании рекурсии, по мере увеличения дерева может переполниться стек. Лучше использовать итеративный подход с классами Queue и Stack. Объекты будут расположены в памяти, соответственно величина дерева будет ограничивается её объёмом, также удобно управлять поведением обхода. Принцип реализации аналогичен рекурсии, но памяти будет больше, да и в целом итеративный подход будет работать быстрее

Ответить

−

0

+

Роман Кончалов

Уровень 28, Россия

EXPERT

15 апреля, 12:54

...

Даже относительно маленький стек в 128 тысяч байт тяжело переполнить. При глубине рекурсии в 50 итераций сбалансированное бинарное дерево содержит более КВАДРИЛЛИОНА элементов. Для переполнения указанного стека с указанной глубиной рекурсии вызываемый метод должен содержать более 300 локальных переменных, чтобы размер фрейма на стеке превысил 2 560 байт. Для глубины 100 будет уже нониллион элементов ($2^{100} \sim 10^{30}$), но хватит и 150 локальных переменных. Как это переполнить, я не представляю, скорее причиной будет ошибка с условием выхода из рекурсии. Тут да, ошибиться легче, нежели с циклом.

Ответить

−

0

+

Andrey Chuev

Уровень 36, Москва, Russian Federation

16 апреля, 14:18

...

Класс может применяться на мобильных устройствах, где размер стека будет существенно отличаться. Если использовать рекурсию в реализации, как потом исправлять баги в связи с переполнением если созданный класс будет идти с какой-нибудь библиотекой? Да и с рекурсией нельзя произвести обход в ширину.

Нет смысла от всех этих расчётов. Безопаснее работать с основной памятью (кучей) через соответствующие объекты, в целом итеративный подход работает быстрее. Не вижу преимуществ рекурсии.

Ответить

−

0

+

Антон

Уровень 23, Москва, Russian Federation

12 января, 22:16

...

Для проходимцев

Задача не требует полной реализации данного дерева поскольку оно бесполезно. С задачей справляется простой АррайЛист из узлов. проходим по нему проверяя лево право, что свободно - добавляем новый элемент прописывая связи. собственно по уровневое заполнение дерева гарантирует очередность самого листа. нахождение узла - элементарно прохождение по Листу удаление находим удаляемый элемент создаем рекурсивное прохождение по веткам от него, удаляя все найденные узлы из основного Листа если туго в рекурсию можно сделать тоже самое с использованием очереди(сложнее для понимания как по мне)

и главное размер - валя не учитывает корневой элемент потому размер это лист.сизе()-1

куда добавлять элемент после удаления решай сам вал принимает с вариантами как в первой свободное место так и добавление в последний уровень

Ответить

−

0

+

Михаил

Full Stack Developer в финтех

8 января, 23:55

...

Предложенное "правильное решение" занимает почти 800 строк. Пусть даже половина из них комментарии, остается 400. Мне удалось всё уместить в 160. При добавлении или удалении элемента записываем элемент не только в соответствующее поле другого элемента, но и кладём в общую коллекцию. По этой коллекции гоняем стримами, фильтруя нужные нам элементы и меняя их свойства в соответствии с заданием. Валидатор принимает, код читаемый, быстроедействие не пострадало, это ли не профит.

Ответить

−

0

+

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ


- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

 Русский

▼

