23.Что вы знаете о коллекциях типа Queue? Могут ли нестатические методы перегрузить статические? Покажите примеры использования модификаторов volatile и метода yield().

Queue — коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки. В дополнение к базовым операциям интерфейса Collection, очередь предоставляет дополнительные операции вставки, получения и контроля.

Очереди обычно, но не обязательно, упорядочивают элементы в FIFO (first-in-first-out, «первым вошел — первым вышел») порядке.

Метод offer() вставляет элемент в очередь, если это не удалось — возвращает false. Этот метод отличается от метода add() интерфейса Collection тем, что метод add() может неудачно добавить элемент только с использованием unchecked исключения.

Методы remove() и poll() удаляют верхушку очереди и возвращают ее. Какой элемент будет удален (первый или последний) зависит от реализации очереди. Методы remove() и poll() отличаются лишь поведением, когда очередь пустая: метод remove() генерирует исключение, а метод poll() возвращает null.

Методы element() и peek() возвращают (но не удаляют) верхушку очереди.

Могут ли нестатические методы перегрузить статические?

Да. Это будут просто два разных метода для программы. Статический будет доступен по имени класса.

Покажите примеры использования модификаторов volatile и метода yield()

Помещение модификатора volatile перед определением переменной заставляет принудительно всегда читать и писать значение только в обычную (медленную) память (а не кэшировать). Записывается как: private volatile boolean varName;

Статический метод Thread.yield() заставляет процессор переключиться на обработку других потоков системы. Метод может быть полезным, например, когда поток ожидает наступления какого-либо события и необходимо чтобы проверка его наступления происходила как можно чаще. В этом случае можно поместить проверку события и метод Thread.yield() в цикл:

**public** **class** Main {

**public** **volatile** **long** x = 2222222222222222222L;

**public** **static** **void** main(String[] args) {

}

}

Java-машина не будет помещать ее в кэш. Так что ситуация, когда 10 потоков работают со своими локальными копиями исключена.

**public** **class** ThreadExample **extends** Thread {

**public** ThreadExample() {

**this**.start();

}

**public** **void** run() {

System.out.println(Thread.currentThread().getName() + " уступает свое место другим");

Thread.yield();

System.out.println(Thread.currentThread().getName() + " has finished executing.");

}

**public** **static** **void** main(String[] args) {

**new** ThreadExample();

**new** ThreadExample();

**new** ThreadExample();

}

}

Когда мы вызываем метод yield у потока, он фактически говорит другим потокам: «Так, ребята, я никуда особо не тороплюсь, так что если кому-то из вас важно получить время процессора — берите, мне не срочно»