

# Оглавление

<b>Отзывы .....</b>	<b>14</b>
<b>Листинги.....</b>	<b>18</b>
<b>Предисловие .....</b>	<b>29</b>
Как пользоваться книгой .....	30
Примеры исходного кода .....	32
Благодарности.....	33
От издательства .....	34
<b>Глава 1. Введение .....</b>	<b>35</b>
1.1. Кратчайшая история конкурентности.....	35
1.2. Преимущества потоков .....	37
1.2.1. Задействование множества процессоров .....	37
1.2.2. Простота моделирования .....	38
1.2.3. Упрощенная обработка асинхронных событий .....	39
1.2.4. Более отзывчивые пользовательские интерфейсы .....	39
1.3. Риски для потоков .....	40
1.3.1. Угрозы безопасности .....	40
1.3.2. Сбои жизнеспособности .....	43
1.3.3. Угрозы производительности.....	44
1.4. Потоки есть везде.....	44
<b>ЧАСТЬ I. ОСНОВЫ.....</b>	<b>47</b>
<b>Глава 2. Потокобезопасность .....</b>	<b>48</b>
2.1. Что такое потокобезопасность? .....	51
2.1.1. Пример: сервлет без поддержки внутреннего состояния.....	52
2.2. Атомарность .....	53
2.2.1. Состояния гонки .....	54
2.2.2. Пример: состояния гонки в ленивой инициализации.....	55
2.2.3. Составные действия.....	57

2.3. Блокировка .....	58
2.3.1. Внутренние замки .....	60
2.3.2. Повторная входимость .....	62
2.4. Защита состояния с помощью замков .....	63
2.5. Живучесть и производительность .....	66
<b>Глава 3. Совместное использование объектов .....</b>	<b>70</b>
3.1. Видимость .....	70
3.1.1. Устаревшие данные .....	72
3.1.2. Неатомарные 64-разрядные операции .....	73
3.1.3. Блокировка и видимость .....	74
3.1.4. Волатильные переменные .....	75
3.2. Публикация и ускользание .....	77
3.2.1. Приемы безопасного конструирования .....	80
3.3. Ограничение одним потоком .....	81
3.3.1. Узкоспециальное ограничение одним потоком .....	82
3.3.2. Ограничение стекком .....	82
3.3.3. ThreadLocal .....	83
3.4. Немутуируемость .....	85
3.4.1. Финальные поля .....	87
3.4.2. Пример: использование volatile для публикации немутуируемых объектов .....	88
3.5. Безопасная публикация .....	90
3.5.1. Ненадлежащая публикация: хорошие объекты становятся плохими .....	90
3.5.2. Немутуируемые объекты и безопасность при инициализации .....	91
3.5.3. Приемы безопасной публикации .....	92
3.5.4. Фактически немутуируемые объекты .....	93
3.5.5. Мутуируемые объекты .....	94
3.5.6. Безопасное совместное использование объектов .....	95
<b>Глава 4. Компоновка объектов .....</b>	<b>96</b>
4.1. Проектирование потокобезопасного класса .....	96
4.1.1. Сбор требований к синхронизации .....	97
4.1.2. Операции, зависящие от состояния .....	98
4.1.3. Владение состоянием .....	99
4.2. Ограничение одним экземпляром .....	100
4.2.1. Мониторный шаблон Java .....	102
4.2.2. Пример: трекинг такси .....	103

---

4.3. Делегирование потокобезопасности .....	105
4.3.1. Пример: трекер такси с использованием делегирования .....	107
4.3.2. Независимые переменные состояния .....	109
4.3.3. Случаи безуспешного делегирования .....	110
4.3.4. Публикация базовых переменных состояния .....	111
4.3.5. Пример: трекер такси, публикующий свое состояние .....	112
4.4. Добавление функциональности в существующие потокобезопасные классы .....	114
4.4.1. Блокировка на стороне клиента .....	115
4.4.2. Компоновка .....	117
4.5. Документирование политик синхронизации .....	118
4.5.1. Толкование расплывчатой документации .....	119
<b>Глава 5. Строительные блоки .....</b>	<b>121</b>
5.1. Синхронизированные коллекции .....	121
5.1.1. Проблемы синхронизированных коллекций .....	121
5.1.2. Итераторы и исключение <code>ConcurrentModificationException</code> .....	124
5.1.3. Скрытые итераторы .....	125
5.2. Конкурентные коллекции .....	127
5.2.1. <code>ConcurrentHashMap</code> .....	128
5.2.2. Дополнительные атомарные операции над ассоциативным массивом .....	129
5.2.3. <code>CopyOnWriteArrayList</code> .....	129
5.3. Блокирующие очереди и паттерн «производитель-потребитель» .....	130
5.3.1. Пример: поиск на рабочем столе .....	132
5.3.2. Серийное ограничение одним потоком .....	133
5.3.3. Двухсторонние очереди и кража работы .....	135
5.4. Блокирующие и прерываемые методы .....	136
5.5. Синхронизаторы .....	137
5.5.1. Защелки .....	137
5.5.2. <code>FutureTask</code> .....	138
5.5.3. Семафоры .....	141
5.5.4. Барьеры .....	143
5.6. Создание эффективного масштабируемого кэша результатов .....	145
Итоги .....	153

---

<b>ЧАСТЬ II. СТРУКТУРИРОВАНИЕ КОНКУРЕНТНЫХ ПРИЛОЖЕНИЙ</b> .....	<b>155</b>
<b>Глава 6. Выполнение задач</b> .....	<b>156</b>
6.1. Выполнение задач в потоках .....	156
6.1.1. Последовательное выполнение задач .....	157
6.1.2. Явное создание потоков для задач .....	158
6.1.3. Недостатки создания неограниченных потоков .....	159
6.2. Фреймворк Executor .....	160
6.2.1. Пример: веб-сервер с использованием Executor .....	161
6.2.2. Политики выполнения .....	162
6.2.3. Пулы потоков .....	163
6.2.4. Жизненный цикл исполнителя Executor.....	164
6.2.5. Отложенные и периодические задачи .....	166
6.3. Поиск эксплуатационно-пригодного параллелизма .....	167
6.3.1. Пример: последовательный страничный отрисовщик .....	168
6.3.2. Задачи, приносящие результаты: Callable и Future .....	169
6.3.3. Пример: страничный отрисовщик с объектом Future .....	171
6.3.4. Ограничения параллелизации разнородных задач .....	172
6.3.5. CompletionService: исполнитель Executor встречается с очередью BlockingQueue .....	174
6.3.6. Пример: страничный отрисовщик со службой CompletionService .....	175
6.3.7. Наложение временных ограничений на задачи .....	176
6.3.8. Пример: портал бронирования поездов .....	177
Итоги.....	179
<b>Глава 7. Отмена и выключение</b> .....	<b>180</b>
7.1. Отмена задачи .....	181
7.1.1. Прерывание .....	183
7.1.2. Политики прерывания.....	186
7.1.3. Отклик на прерывание.....	188
7.1.4. Пример: хронометрированный прогон.....	190
7.1.5. Отмена с помощью Future .....	191
7.1.6. Работа с непрерываемым блокированием .....	193
7.1.7. Инкапсуляция нестандартной отмены с помощью newTaskFor.....	194
7.2. Остановка поточной службы .....	196
7.2.1. Пример: служба журналирования .....	197
7.2.2. Выключение службы ExecutorService .....	201
7.2.3. Ядовитые таблетки .....	202

---

7.2.4. Пример: служба однократного выполнения.....	203
7.2.5. Ограничения метода shutdownNow .....	204
7.3. Обработка аномальной терминации потоков.....	207
7.3.1. Обработчики неотловленных исключений.....	208
7.4. Выключение JVM.....	210
7.4.1. Хуки.....	210
7.4.2. Потоки-демоны .....	211
7.4.3. Финализаторы.....	212
Итоги.....	213
<b>Глава 8. Применение пулов потоков.....</b>	<b>214</b>
8.1. Неявные стыковки между задачами и политиками выполнения.....	214
8.1.1. Взаимная блокировка с ресурсным голоданием.....	215
8.1.2. Длительные задачи.....	217
8.2. Определение размера пула потоков.....	217
8.3. Конфигурирование класса ThreadPoolExecutor .....	219
8.3.1. Создание и удаление потоков .....	219
8.3.2. Управление задачами очереди.....	221
8.3.3. Политика насыщения .....	223
8.3.4. Фабрики потоков .....	224
8.3.5. Настройка класса ThreadPoolExecutor после конструирования.....	227
8.4. Расширение класса ThreadPoolExecutor .....	228
8.4.1. Пример: добавление статистики в пул потоков .....	229
8.5. Параллелизация рекурсивных алгоритмов .....	230
8.5.1. Пример: фреймворк головоломки .....	232
Итоги.....	238
<b>Глава 9. Приложения с GUI .....</b>	<b>239</b>
9.1. Почему GUI-интерфейсы являются однопоточными?.....	239
9.1.1. Последовательная обработка событий.....	241
9.1.2. Ограничение одним потоком в Swing .....	241
9.2. Кратковременные задачи GUI .....	242
9.3. Длительные задачи GUI.....	245
9.3.1. Отмена .....	247
9.3.2. Индикация хода выполнения и завершения .....	248
9.3.3. SwingWorker.....	249
9.4. Совместные модели данных .....	249
9.4.1. Потокобезопасные модели данных.....	252
9.4.2. Раздвоенные модели данных .....	252
9.5. Другие формы однопоточных подсистем.....	253
Итоги.....	254

**ЧАСТЬ III. ЖИЗНЕСПОСОБНОСТЬ, ПРОИЗВОДИТЕЛЬНОСТЬ  
И ТЕСТИРОВАНИЕ..... 255**

<b>Глава 10.</b> Предотвращение сбоев жизнеспособности .....	256
10.1. Взаимная блокировка.....	256
10.1.1. Взаимные блокировки из-за порядка блокировки.....	258
10.1.2. Взаимная блокировка из-за динамического порядка следования замков.....	260
10.1.3. Взаимные блокировки между взаимодействующими объектами.....	264
10.1.4. Открытые вызовы .....	266
10.1.5. Ресурсные взаимные блокировки .....	269
10.2. Предотвращение и диагностирование взаимной блокировки.....	270
10.2.1. Хронометрированные замки .....	270
10.2.2. Анализ взаимной блокировки с помощью поточных дампов .....	271
10.3. Другие сбои жизнеспособности.....	274
10.3.1. Голодание.....	274
10.3.2. Слабая отзывчивость .....	275
10.3.3. Активная блокировка .....	276
Итоги.....	277
<b>Глава 11.</b> Производительность и масштабирование .....	278
11.1. Некоторые мысли о производительности .....	279
11.1.1. Производительность и масштабируемость.....	280
11.1.2. Оценивание компромиссов производительности .....	282
11.2. Закон Амдала.....	284
11.2.1. Пример: сериализация, скрытая в фреймворках .....	287
11.2.2. Качественное применение закона Амдала.....	289
11.3. Стоимость, вносимая потоком .....	290
11.3.1. Переключение контекста .....	290
11.3.2. Синхронизации памяти .....	291
11.3.3. Блокирование .....	294
11.4. Сокращение конфликта блокировки.....	295
11.4.1. Сужение области действия замка («вошел, вышел»).....	296
11.4.2. Сокращение степени детализации замка.....	298
11.4.3. Чередование блокировок .....	301
11.4.4. Недопущение горячих полей.....	302
11.4.5. Альтернативы исключаящим блокировкам .....	305
11.4.6. Мониторинг задействованности процессоров .....	305
11.4.7. Объединению объектов в пул — «нет!».....	307

11.5. Пример: сравнение производительности ассоциативного массива Map .....	309
11.6. Сокращение издержек на переключение контекста.....	311
Итоги.....	313
<b>Глава 12. Тестирование конкурентных программ .....</b>	<b>315</b>
12.1. Тестирование на правильность .....	317
12.1.1. Базовые модульные тесты.....	319
12.1.2. Тестирование блокирующих операций .....	320
12.1.3. Тестирование на безопасность.....	322
12.1.4. Тестирование на управление ресурсами .....	328
12.1.5. Использование обратных вызовов.....	330
12.1.6. Генерирование большого числа расслоений.....	331
12.2. Тестирование на производительность .....	332
12.2.1. Расширение теста PutTakeTest за счет хронометрирования.....	333
12.2.2. Сравнение многочисленных алгоритмов.....	337
12.2.3. Измерение отзывчивости .....	338
12.3. Предотвращение ошибок при тестировании.....	340
12.3.1. Сбор мусора.....	341
12.3.2. Динамическая компиляция.....	341
12.3.3. Нереалистичный отбор ветвей кода .....	343
12.3.4. Нереалистичные уровни конфликта .....	344
12.3.5. Устранение мертвого кода .....	345
12.4. Комплементарные подходы к тестированию.....	347
12.4.1. Ревизия кода.....	348
12.4.2. Инструменты статического анализа.....	348
12.4.3. Аспектно-ориентированные методы тестирования .....	351
12.4.4. Средства профилирования и мониторинга .....	351
Итоги.....	352
<b>ЧАСТЬ IV. ПРОДВИНУТЫЕ ТЕМЫ.....</b>	<b>353</b>
<b>Глава 13. Явные замки .....</b>	<b>354</b>
13.1. Lock и ReentrantLock.....	354
13.1.1. Опрашиваемое и хронометрируемое приобретение замка .....	356
13.1.2. Прерываемое приобретение замка .....	359
13.1.3. Неблочно структурированная замковая защита.....	360
13.2. Соображения по поводу производительности .....	360
13.3. Справедливость .....	362

13.4. Выбор между <code>synchronized</code> и <code>ReentrantLock</code> .....	365
13.5. Замки чтения-записи.....	366
Итоги.....	371
<b>Глава 14.</b> Построение настраиваемых синхронизаторов.....	372
14.1. Управление зависимостью от состояния.....	373
14.1.1. Пример: распространение сбоя предусловия на вызывающие элементы кода .....	374
14.1.2. Пример: грубая блокировка с помощью опрашивания и сна.....	377
14.1.3. Очереди условий на освобождение.....	379
14.2. Использование очередей условий .....	382
14.2.1. Условный предикат .....	382
14.2.2. Слишком раннее пробуждение.....	384
14.2.3. Пропущенные сигналы .....	386
14.2.4. Уведомление.....	386
14.2.5. Пример: шлюзовый класс.....	389
14.2.6. Вопросы безопасности подклассов .....	390
14.2.7. Инкапсулирование очередей условий .....	392
14.2.8. Протоколы входа и выхода .....	393
14.3. Явные объекты условий .....	393
14.4. Анатомия синхронизатора .....	397
14.5. <code>AbstractQueuedSynchronizer</code> .....	399
14.5.1. Простая защелка.....	401
14.6. AQS в классах синхронизатора библиотеки <code>java.util.concurrent</code> .....	403
14.6.1. <code>ReentrantLock</code> .....	403
14.6.2. <code>Semaphore</code> и <code>CountDownLatch</code> .....	405
14.6.3. <code>FutureTask</code> .....	406
14.6.4. <code>ReentrantReadWriteLock</code> .....	407
Итоги.....	407
<b>Глава 15.</b> Атомарные переменные и неблокирующая синхронизация .....	409
15.1. Недостатки замковой защиты.....	410
15.2. Аппаратная поддержка конкурентности .....	412
15.2.1. Сравнить и обменять.....	413
15.2.2. Неблокирующий счетчик.....	415
15.2.3. Поддержка операции CAS в JVM.....	417
15.3. Классы атомарных переменных.....	417
15.3.1. Атомарные компоненты в качестве «более качественных волатильных» .....	419
15.3.2. Сравнение производительности: замки против атомарных переменных .....	420



---

15.4. Неблокирующие алгоритмы .....	424
15.4.1. Неблокирующий стек .....	425
15.4.2. Неблокирующий связный список .....	427
15.4.3. Обновители атомарных полей .....	431
15.4.4. Проблема АВА .....	433
Итоги.....	434
<b>Глава 16. Модель памяти Java.....</b>	<b>435</b>
16.1. Что такое модель памяти и зачем она нужна?.....	435
16.1.1. Платформенные модели памяти.....	437
16.1.2. Переупорядочивание .....	438
16.1.3. Модель памяти Java в менее чем 500 словах.....	440
16.1.4. Совмещение за счет синхронизации.....	443
16.2. Публикация.....	445
16.2.1. Небезопасная публикация.....	446
16.2.2. Безопасная публикация.....	447
16.2.3. Идиомы безопасной инициализации.....	448
16.2.4. Блокировка с двойной проверкой.....	450
16.3. Безопасность инициализации.....	452
Итоги.....	454
<b>ПРИЛОЖЕНИЕ А. АННОТАЦИИ ДЛЯ КОНКУРЕНТНОСТИ.....</b>	<b>456</b>
А.1. Аннотации классов .....	456
А.2. Аннотации полей и методов.....	457
<b>Библиография.....</b>	<b>459</b>