КАК ЗАГУБИТЬ ПРОИЗВОДИТЕЛЬНОСТЬ ENTERPRISE-ПРИЛОЖЕНИЯ С ПОМОЩЬЮ НЕЭФФЕКТИВНОГО КОДА

LUXOFT

LXFT LISTED NYSE

Пара слов обо мне

- работаю в «Люксофте»
- пишу на Java
- люблю разбираться с проблемами производительности

Эпиграф

Большую часть своего времени вы будете тратить на то, чтобы фиксить тупые ошибки, в т. ч. в перформансе.

Алексей Шипилёв
из доклада «Performance Optimization 101»
https://youtu.be/EupF3VNXPPQ?t=651

Что и на чём измеряли?

Код: https://github.com/stsypanov/logeek-night-benchmark.git

Java 8: ветка **master**

Java 9: ветка **master-java-9**

Intel Core i3-3220

Java 8 (Java HotSpot 64-Bit Server VM (build 25.144-b01))

Java 9 (Java HotSpot 64-Bit Server VM (build 9+181))

Что узнаем?

- с каким прищуром смотреть на код, чтобы найти узкие места
- распространённые антипаттерны в коде enterprise-приложений
- неочевидные грабли
- способы поиска и обхода граблей

Разминка

```
class User {
 String name;
 int age;
}
```

Нужно сравнивать объекты

```
@EqualsAndHashCode
class User {
 String name;
 int age;
}
```

equals(): в чём сила, брат?

```
@EqualsAndHashCode
class User {
 String name;
 int age;
}
```

```
@Override
public boolean equals(Object o) {

 // ……
 // return true;

 return false;
}
```

User и [неявный] equals()

```
@EqualsAndHashCode
class User {
 String name;
 int age;
}
```

```
if (name == null && that.name != null)
 return false;
if (name != null && !name.equals(that.name))
 return false;

return age == that.age;
```

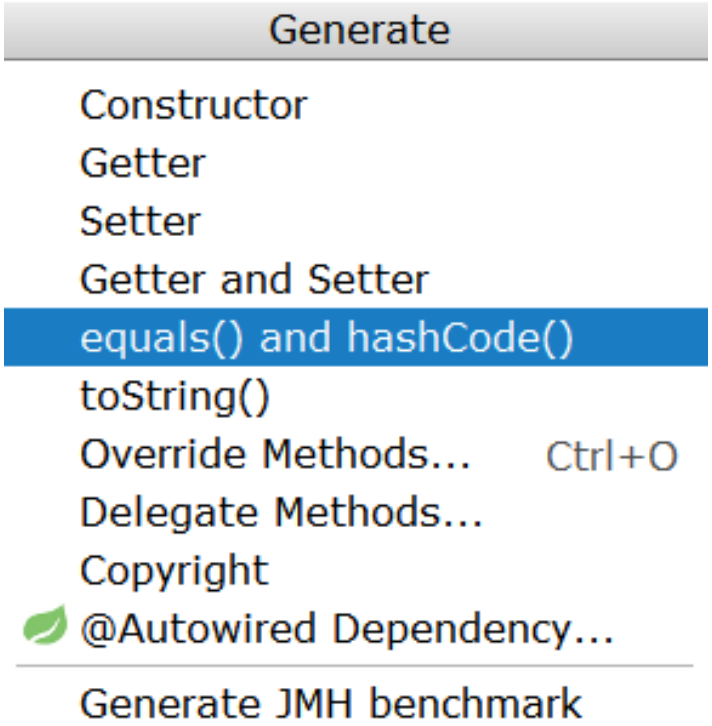
User и [неявный] equals()

```
@EqualsAndHashCode
class User {
 String name;
 int age;
}
```

```
if (name == null && that.name != null)
 return false;
if (name != null && !name.equals(that.name))
 return false;

return age == that.age;
```

User и явный equals()

```
class User {
 String name;
 int age;
}
```



equals(): присмотримся

```
class User {
 String name;
 int age;
}
```

```
@Override
public boolean equals(Object o) {
 if (this == o) return true;

 if (o == null || getClass() != o.getClass())
  return false;

 User that = (User) o;

 return age == that.age &&
  Objects.equals(name, that.name);
}
```

User: расширяемся!

```
class User {
 List<T> props;
 String name;
 int age;
}
```

equals(): меня терзают смутные сомнения

```
class User {
 List<T> props;
 String name;
 int age;
}
```

```
@Override
public boolean equals(Object o) {
 if (this == o) return true;

 if (o == null || getClass() != o.getClass())
  return false;

 User that = (User) o;

 return age == that.age &&
  Objects.equals(props, that.props) &&
  Objects.equals(name, that.name);
}
```

equals(): опаньки!

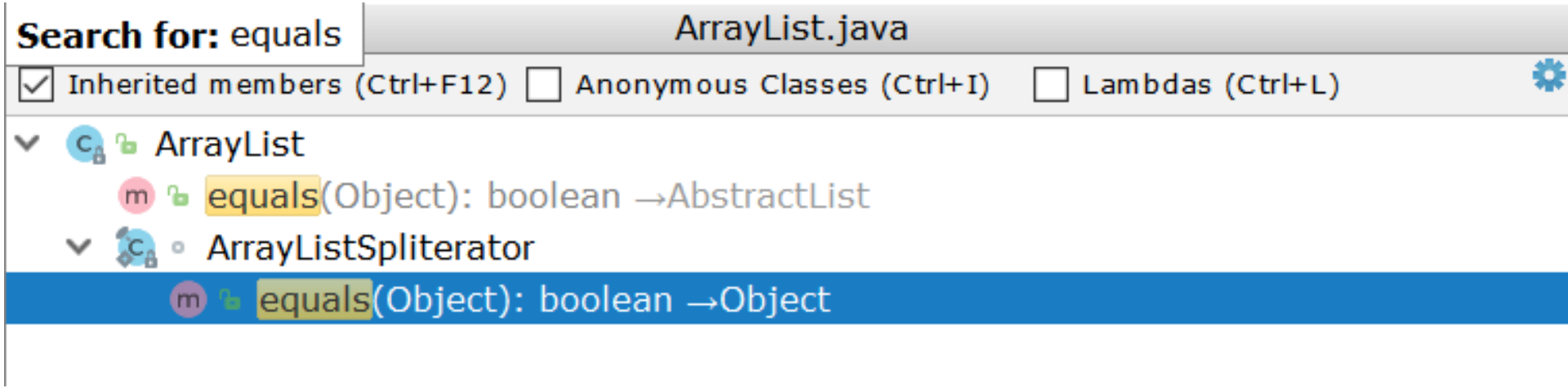
```
class User {
 List<T> props;
 String name;
 int age;
}
```

```
@Override
public boolean equals(Object o) {
 if (this == o) return true;

 if (o == null || getClass() != o.getClass())
  return false;

 User that = (User) o;

 return age == that.age &&
  Objects.equals(props, that.props) &&
  Objects.equals(name, that.name);
}
```

String.equals()

```
public boolean equals(Object o) {
 if (this == o) return true;
 if (o instanceof String) {
  String anotherString = (String) o;
  int n = value.length;
  if (n == anotherString.value.length) {
   //познаковое сравнение
  }
  return false;
 }
 return false;
}
```

ArrayList.equals()



AbstractList.equals()

```
public boolean equals(Object o) {
  if (o == this)                  return true;
  if (!(o instanceof List)) return false;

  ListIterator<E> e1 = listIterator();
  ListIterator<?> e2 = ((List<?>) o).listIterator();
  while (e1.hasNext() && e2.hasNext()) {
    E o1 = e1.next();
    Object o2 = e2.next();
    if (o1 == null ? o2 != null : !o1.equals(o2)) return false;
  }
  return !(e1.hasNext() || e2.hasNext());
}
```

AbstractList.equals(): что важно?

```
public boolean equals(Object o) {
  if (o == this)                  return true;
  if (!(o instanceof List)) return false;

  ListIterator<E> e1 = listIterator();
  ListIterator<?> e2 = ((List<?>) o).listIterator();
  while (e1.hasNext() && e2.hasNext()) {
    E o1 = e1.next();
    Object o2 = e2.next();
    if (o1 == null ? o2 != null : !o1.equals(o2)) return false;
  }
  return !(e1.hasNext() || e2.hasNext());
}
```

Переписанный equals()

```
class User {
 List<T> props;
 String name;
 int age;
}
```

```
@Override
public boolean equals(Object o) {
 if (this == o) return true;

 if (o == null || getClass() != o.getClass())
  return false;

 User that = (User) o;

 return age == that.age &&
  Objects.equals(name, that.name) &&
  Objects.equals(props, that.props);
}
```

Чё там у котлиновцев?

```
data class User(val name: String, val age: Int)
```

```
public boolean equals(Object o) {
 if (this == o) return true;
 if (o instanceof User) {
  User u = (User) o;
  if (Intrinsics.areEqual(name, u.name) && age == u.age)
   return true;
 }
 return false;
}
```

В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-170178
IDEA-170178 Compare collections at the very end of IDEA-generated equals()

https://github.com/rzwitserloot/lombok/issues/1543
[Feature request] Reordering of comparison in @EqualsAndHashCode for better performance #1543

https://youtrack.jetbrains.com/issue/KT-23184
KT-23184 Auto-generated equals() of data class is not optimal

Усложним задачу

```
void check(Dto dto) {
 Entity entity = jpaRepository.findOne(dto.getId());

 boolean valid = dto.isValid();




}
```

Усложним задачу

```
void check(Dto dto) {
 Entity entity = jpaRepository.findOne(dto.getId());

 boolean valid = dto.isValid();

 if (valid && entity.hasGoodRating()) {
  //do smth
 }

}
```

Ничто не предвещало беды

```
void check(Dto dto) {
 Entity entity = jpaRepository.findOne(dto.getId());

 boolean valid = dto.isValid();

 if (valid && entity.hasGoodRating()) {
  //do smth
 }

}
```

«Скрипач не нужен, дядя Вова»

```
void check(Dto dto) {
 Entity entity = jpaRepository.findOne(dto.getId());

 boolean valid = dto.isValid();

 if (valid && entity.hasGoodRating()) {
  //do smth
 }

}
```

Отложенный запрос

```
void check(Dto dto) {
 boolean valid = dto.isValid();

 if (valid && hasGoodRating(dto)) {
  //do smth
 }
}

boolean hasGoodRating(Dto dto) {
  Entity entity = jpaRepository.findOne(dto.getId());
  return entity.hasGoodRating();
};
```

Пример без явного ветвления

```
boolean checkChild(Dto dto) {
 Long id = dto.getId();
 Entity entity = jpaRepository.findOne(id);
 return dto.isValid() && entity.hasChild();
}
```

Fast return

```
boolean checkChild(Dto dto) {
 Long id = dto.getId();
 Entity entity = jpaRepository.findOne(id);
 return dto.isValid() && entity.hasChild();
}
```

```
boolean checkChild(Dto dto) {
 if (!dto.isValid()) return false;

 return jpaRepository.findOne(dto.getId()).hasChild();
}
```

Совет 1

ЕСЛИ МЕДЛЕННОЕ ДЕЙСТВИЕ МОЖНО ОТЛОЖИТЬ – ОТЛОЖИ ЕГО.

Применение: действия стоит выполнять в порядке возрастания их сложности. Возможно, выполнять его не придётся.

Едем дальше

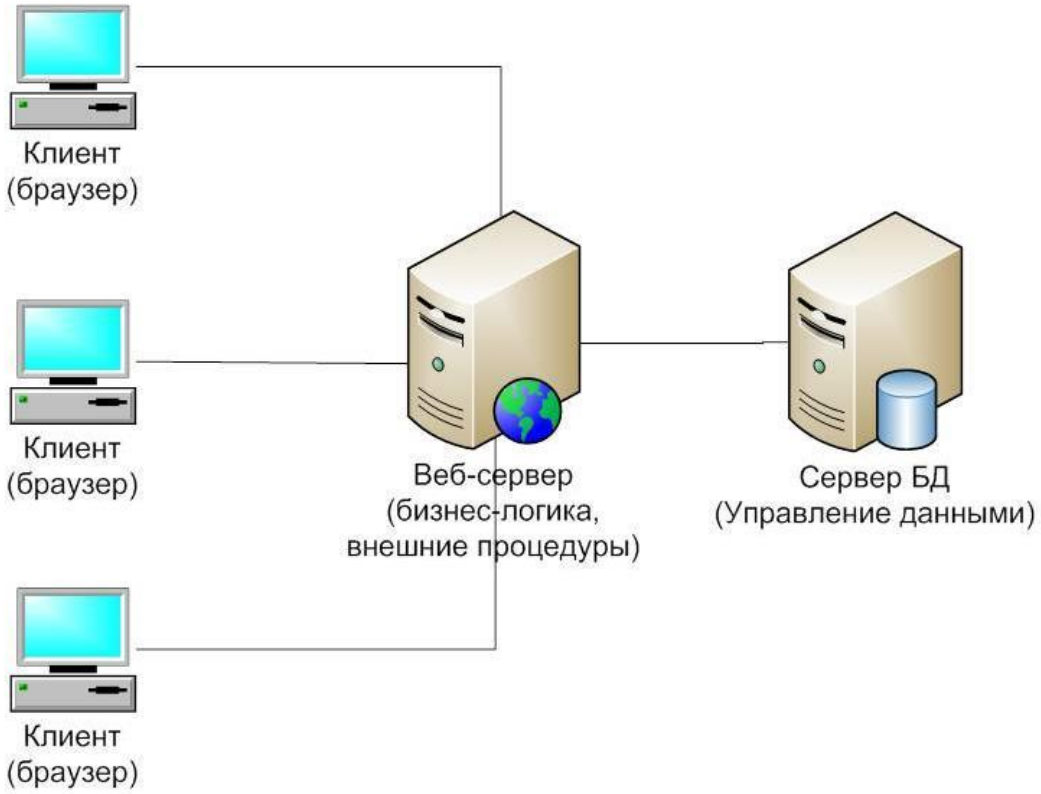
```
void enrollStudents(Set<Long> ids) {
 for (Long id : ids) {
  Student student = jpaRepository.findOne(id);
  enroll(student);
 }
}
```

Внезапно

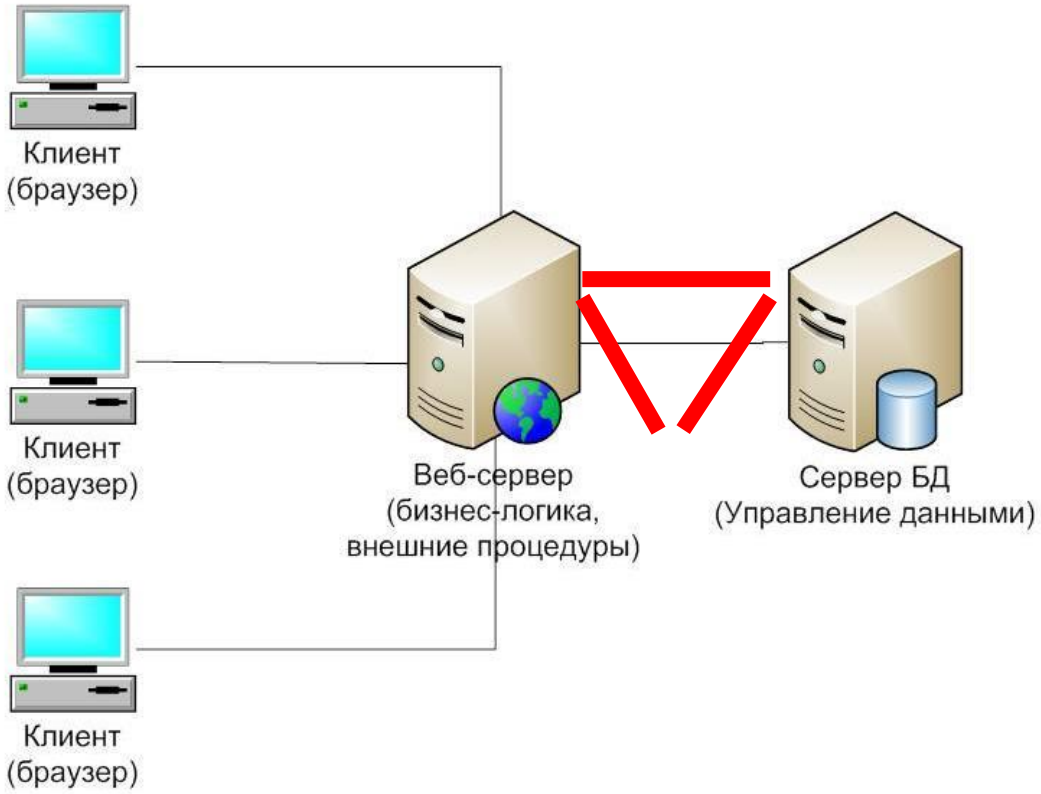
O(n) 

```
void enrollStudents(Set<Long> ids) {
 for (Long id : ids) {
  Student student = jpaRepository.findOne(id);
  enroll(student);
 }
}
```

План кровавого Ынтерпрайза



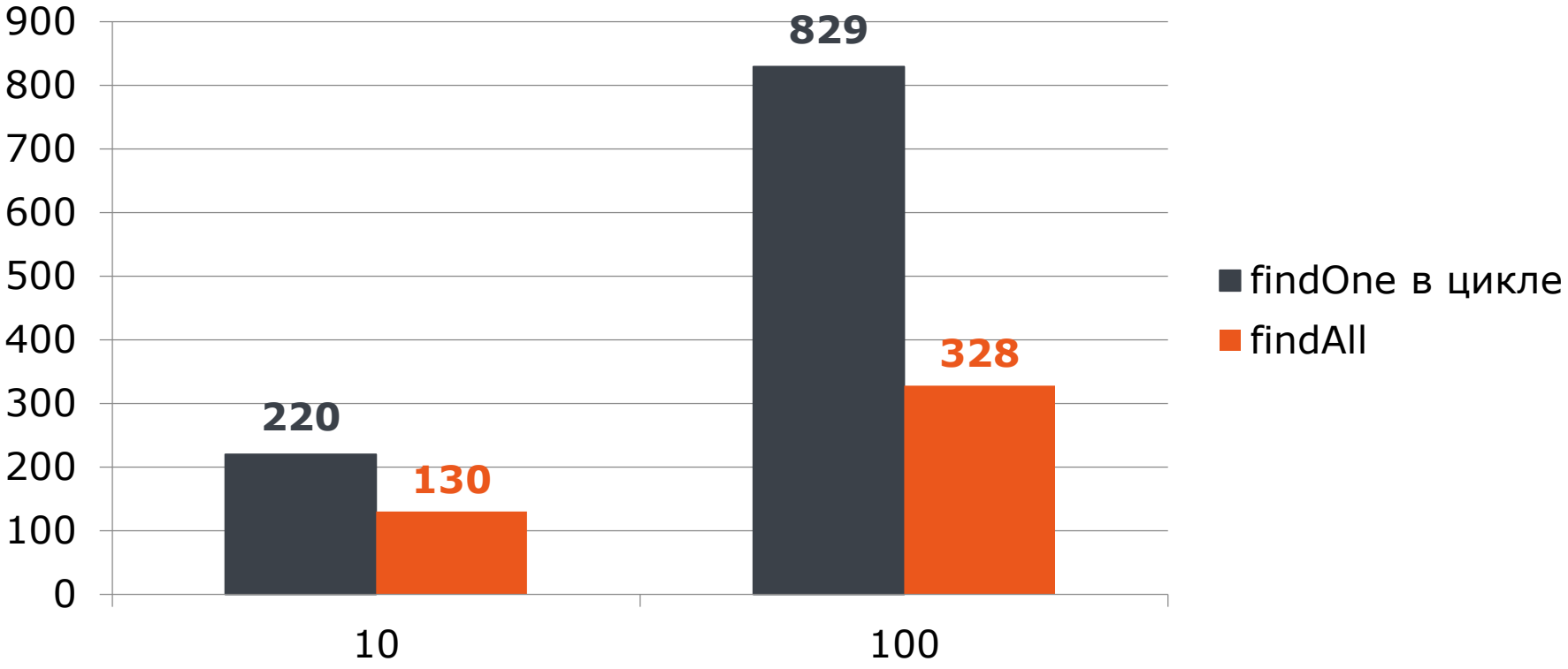
Потери на стыке



«Грузите апельсины бочках»

```
void enrollStudents(Set<Long> ids) {
 if (ids.isEmpty()) return;

 for (Student student : jpaRepository.findAll(ids)) {
  enroll(student);
 }
}
```

Время выполнения, мкс



Более сложный пример

```
for (Long id : ids) {
 Region region = jpaRepository.findOne(id);

 if (region == null) {
  region = new Region();
  region.setId(id);
 }

 use(region);
}
```

Более сложный пример: затруднение

```
for (Long id : ids) {
 Region region = jpaRepository.findOne(id);

 if (region == null) {
  region = new Region();
  region.setId(id);
 }

 use(region);
}
```

Что такое findOne?

```
for (Long id : ids) {
 Region region = jpaRepository.findOne(id);

 if (region == null) {
  region = new Region();
  region.setId(id);
 }

 use(region);
}
```

Map подменяет БД

```
Map<Long, Region> regionMap = jpaRepository.findAll(regionIds)
 .stream().collect(toMap(Region::getId, identity()));

for (Long id : regionIds) {
 Region region = map.get(id);

 if (region == null) {
  region = new Region();
  region.setId(id);
 }

 use(region);
}
```

Пример в обратную сторону

```
@Transactional
public void audit(List<AuditDto> inserts) {
 inserts.map(this::toEntities).forEach(saver::save);
}
```

Пример в обратную сторону

```
@Transactional
public void audit(List<AuditDto> inserts) {
 inserts.map(this::toEntities).forEach(saver::save);
}

// class Saver

@Transactional
public void save(List<AuditEntity> entities) {
 jpaRepository.save(entities);
}
```

Отдельная транзакция

```
@Transactional
public void audit(List<AuditDto> inserts) {
 inserts.map(this::toEntities).forEach(saver::save);
}

// class Saver

@Transactional(propagation = Propagation.REQUIRES_NEW)
public void save(List<AuditEntity> entities) {
 jpaRepository.save(entities);
}
```

Управление транзакциями

```
@Transactional
public void audit(List<AuditDto> inserts) {
 inserts.map(this::toEntities).forEach(saver::save);
}

// class Saver

@Transactional(propagation = Propagation.REQUIRES_NEW)
public void save(List<AuditEntity> entities) {
 jpaRepository.save(entities);
}
```

Почему плохо?

```
Transaction tx = null;
try (Session session = sessionFactory.openSession()) {
 tx = session.beginTransaction();

 //работаем

 tx.commit();
} catch (Exception e) {
 if (tx != null) {
  tx.rollback();
 }
 throw e;
}
```

Почему плохо: накладные расходы

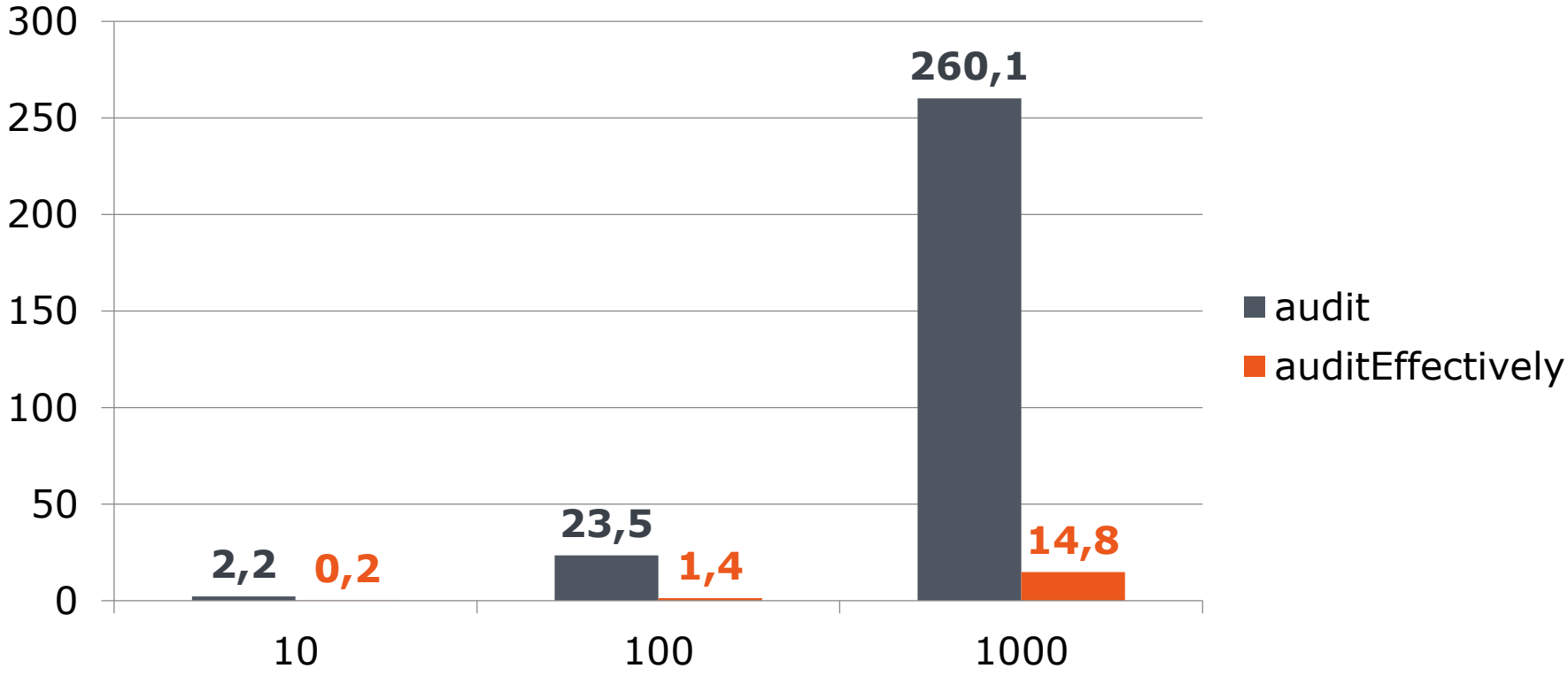
```
Transaction tx = null;
try (Session session = sessionFactory.openSession()) {
 tx = session.beginTransaction();

 //работаем

 tx.commit();
} catch (Exception e) {
 if (tx != null) {
  tx.rollback();
 }
 throw e;
}
```

Исправим

```
@Transactional
public void audit(List<AuditDto> inserts) {
 inserts.map(this::toEntities).forEach(saver::save);
}
```

```
@Transactional
public void audit(List<AuditDto> inserts) {
 List<AuditEntity> items = inserts.stream().map(this::toEntities)
  .collect(toList());

 saver.save(items);
}
```

Время выполнения, мкс



В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-165730

IDEA-165730 Warn about inefficient usage of JpaRepository

В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-165730
IDEA-165730 Warn about inefficient usage of JpaRepository

Правило применимо не только к БД и Тагир Валеев пошел дальше:

https://youtrack.jetbrains.com/issue/IDEA-165942
~~IDEA-165942~~ Inspection to replace method call in a loop with bulk operation

Fixed in: **2017.1**

И если раньше было так

```
List<Long> list = new ArrayList<>();
for (Long id : items) {
 list.add(id);
}
```

То теперь стало так

```
List<Long> list = new ArrayList<>();
for (Long id : items) {
 list.add(id);
}
```

```
List<Long> list = new ArrayList<>();
list.addAll(items);
```

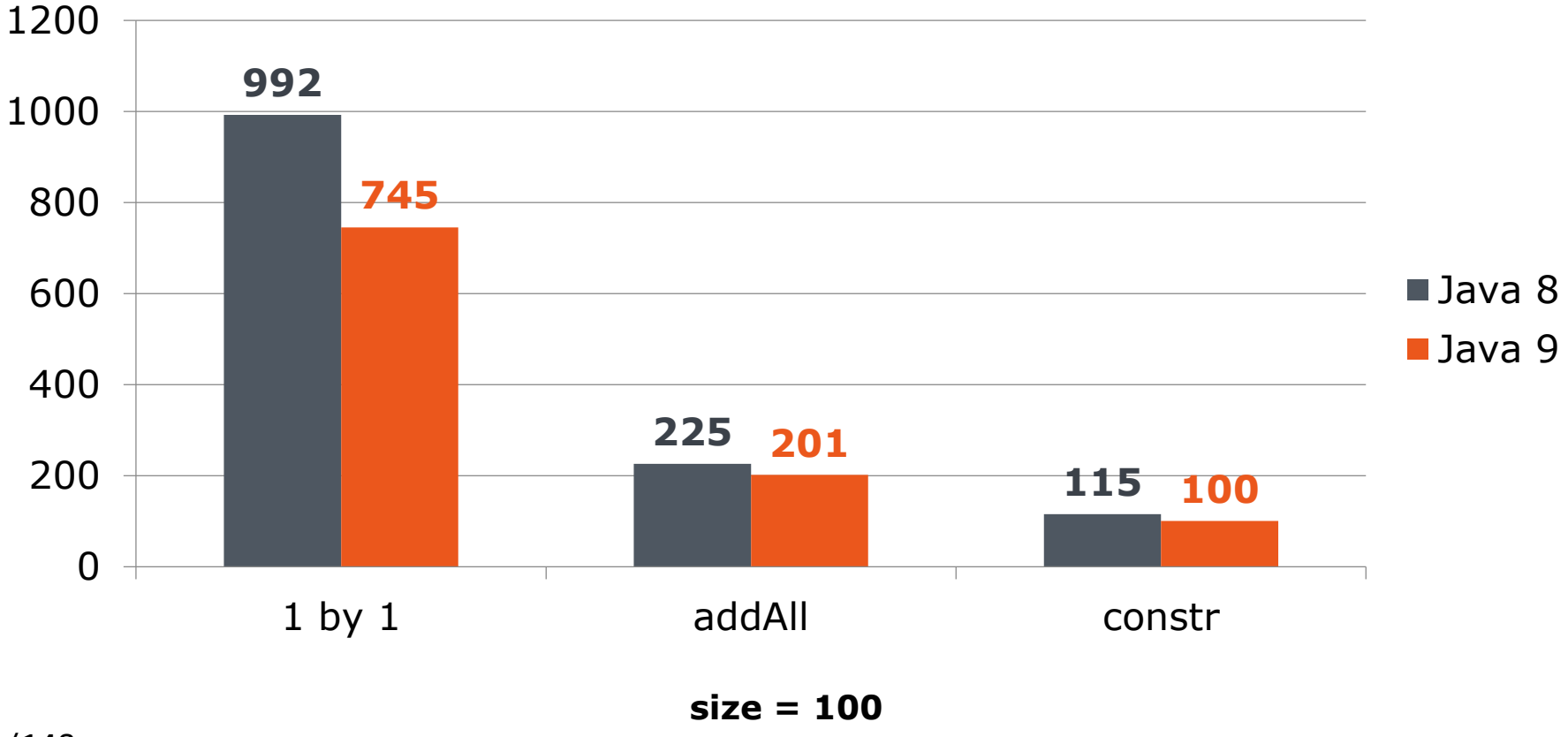
Или даже так

```
List<Long> list = new ArrayList<>();
for (Long id : items) {
 list.add(id);
}
```

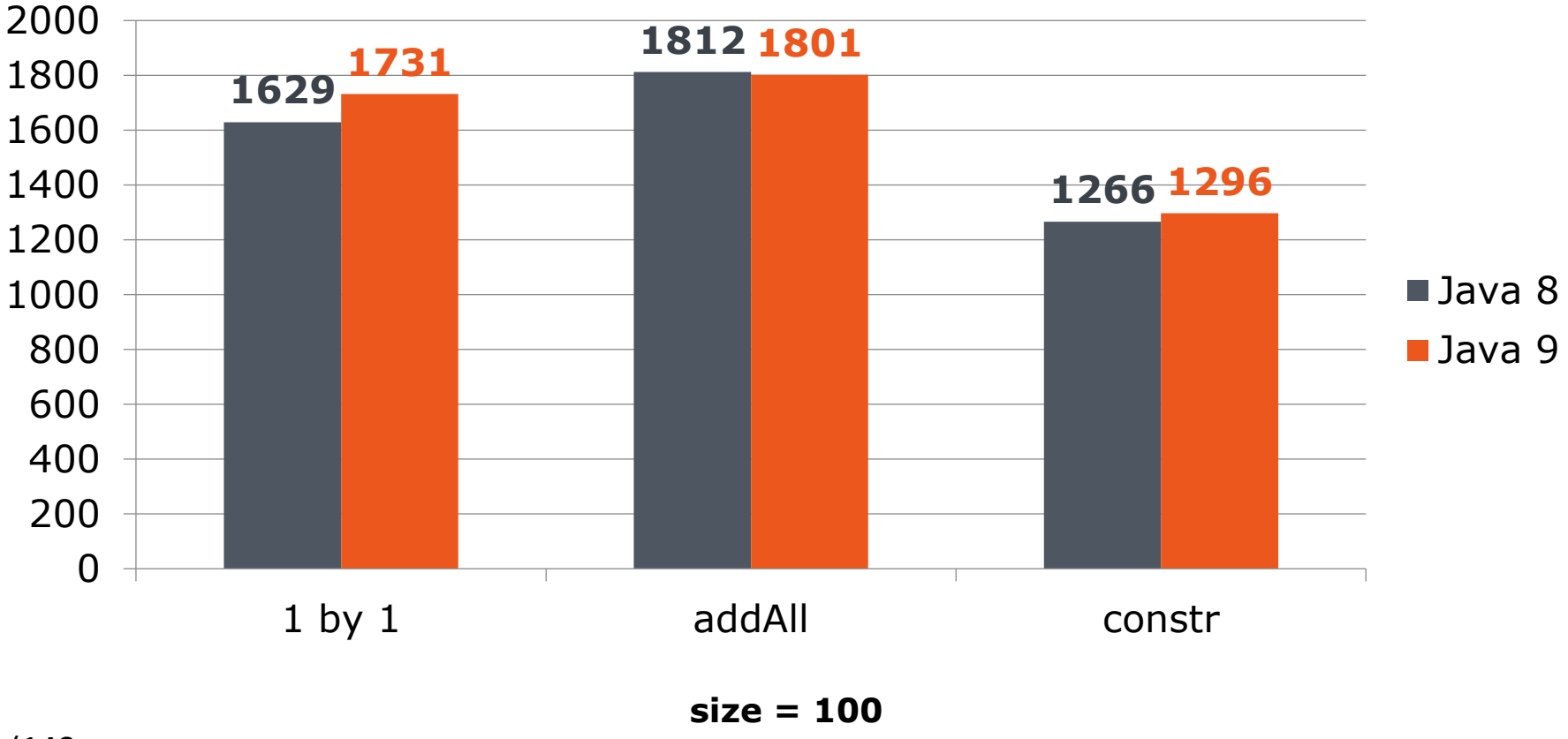
```
List<Long> list = new ArrayList<>();
list.addAll(items);
```

```
List<Long> list = new ArrayList<>(items);
```

ArrayList, время выполнения, нс



HashSet, время выполнения, нс



В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-138456

~~IDEA-138456~~ Add new Performance inspection: 'Collection.addAll() can be replaced with parametrized constructor'

Fixed in: **142.1217**

```
List<Long> list = new ArrayList<>();
list.addAll(items);
```

```
List<Long> list = new ArrayList<>(items);
```

В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-138456
~~IDEA-138456~~ Add new Performance inspection: 'Collection.addAll() can be replaced with parametrized constructor'

Fixed in: **142.1217**

https://youtrack.jetbrains.com/issue/IDEA-178761
~~IDEA-178761~~ Inspection 'Collection.addAll() can be replaced with parametrized constructor' should be turned on by default

Fixed in: **2017.3**

ArrayList.remove(index)

```
for (int i = from; i < to; i++) {
 list.remove(from);
}
```

ArrayList.remove(index)

```
public E remove(int index) {
 Objects.checkIndex(index, size);

 modCount++;
 E oldValue = elementData(index);

 int numMoved = size - index - 1;
 if (numMoved > 0)
  System.arraycopy(array, index+1, array, index, numMoved);
 array[--size] = null; // clear to let GC do its work

 return oldValue;
}
```

ArrayList.remove(index)

```
public E remove(int index) {
 Objects.checkIndex(index, size);

 modCount++;
 E oldValue = elementData(index);

 int numMoved = size - index - 1;
 if (numMoved > 0)
  System.arraycopy(array, index+1, array, index, numMoved);
 array[--size] = null; // clear to let GC do its work

 return oldValue;
}
```

Решение: ArrayList.subList().clear()

```
list.subList(from, to).clear();
```

Решение: ArrayList.subList().clear()

```
list.subList(from, to).clear();

public void clear() {
 removeRange(0, size());
}

void removeRange(int from, int to) {
 checkForComodification();
 parent.removeRange(parentOffset + from, parentOffset + to);
 this.modCount = parent.modCount;
 this.size -= toIndex - fromIndex;
}
```

Если удалённое значение используется?

```
for (int i = from; i < to; i++) {
 E removed = list.remove(from);
 use(removed)
}
```

Если удалённое значение используется?

```
List<String> removed = list.subList(from, to);
removed.forEach(this::use);
removed.clear();
```

ArrayList.remove(i): обратный проход

```
//прямой проход
for (int i = from; i < to; i++) {
 E removed = list.remove(from);
 use(removed)
}
```

```
//обратный проход
for (int i = to - 1; i >= from; i--) {
 E removed = list.remove(i);
 use(removed)
}
```

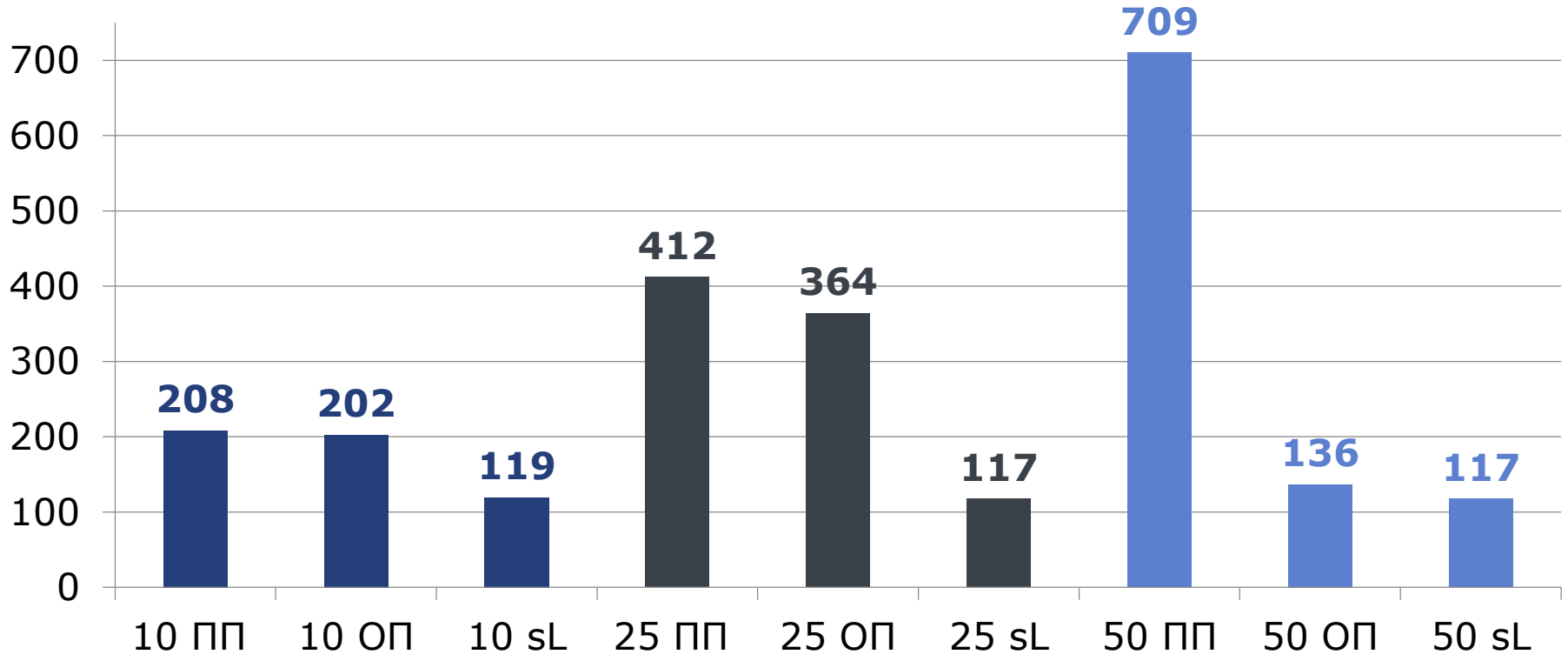
Сравним удаление из середины списка

```
for (int i = from; i < to; i++)          // прямой проход
 list.remove(from);
```

```
for (int i = to - 1; i >= from; i--)     // обратный проход
 list.remove(i);
```

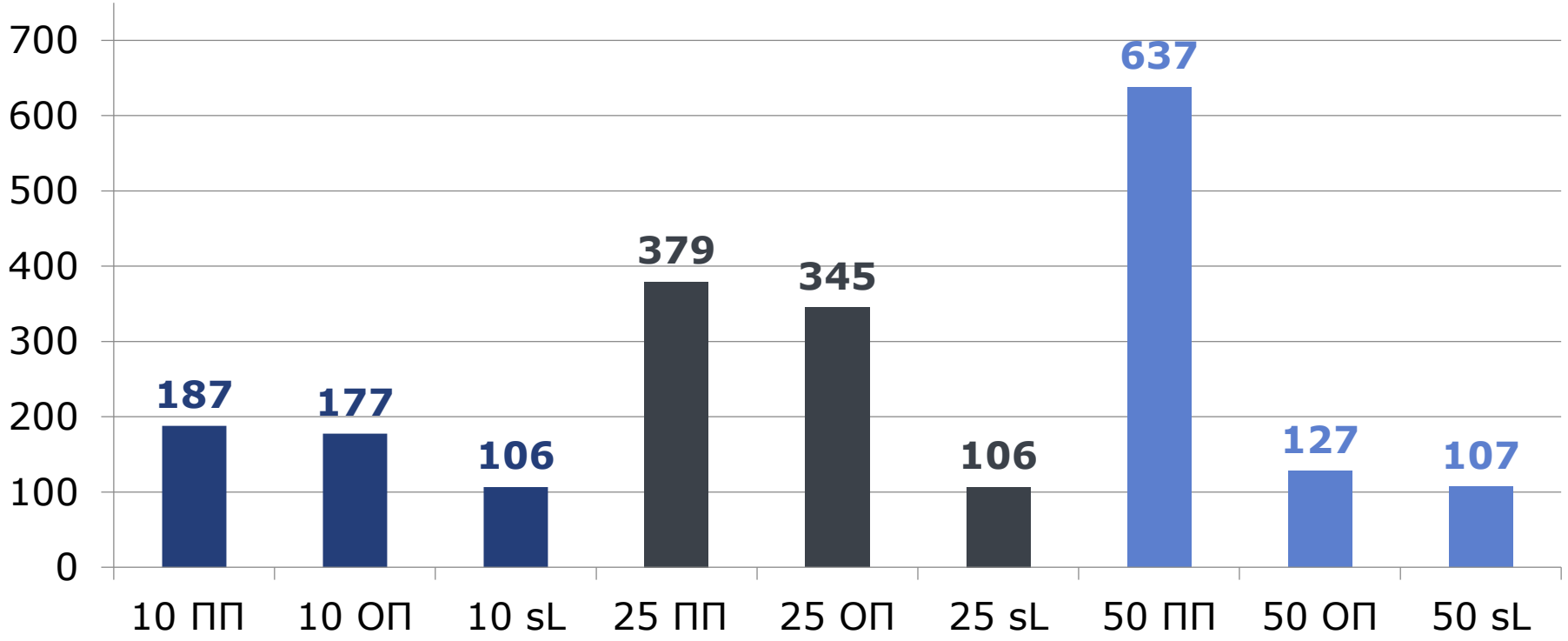
```
list.subList(from, to).clear();          // subList
```

Java 8, время выполнения, нс, size = 100



ПП – прямой порядок, ОП – обратный порядок, sl – subList

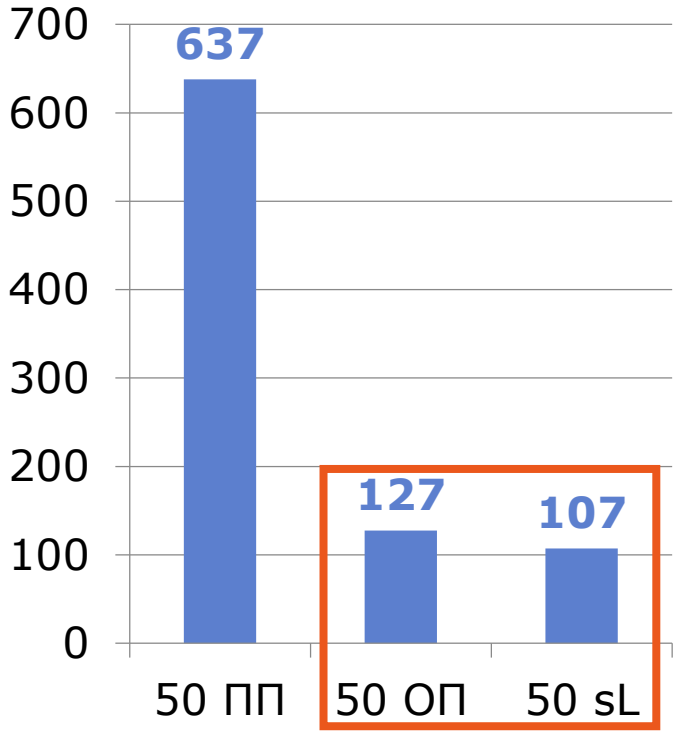
Java 9, время выполнения, нс, size = 100



ПП – прямой порядок, ОП – обратный порядок, sl – subList

Всегда удаляем из последней ячейки

```
public E remove(int i) {
 Objects.checkIndex(i, size);

 modCount++;
 E oldValue = elementData(i);

 int moved = size - i - 1;
 if (moved > 0)
  arraycopy(array, i+1, array, i, moved);
 array[--size] = null;

 return oldValue;
}
```



В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-177466

IDEA-177466 Detect List.remove(index) called in a loop (Open)

Совет 2

СОБИРАЙ И ВЛАСТВУЙ.

Применение: повторяющиеся действия, выполняемые в цикле, часто можно свести к одной массовой операции, особенно когда речь идёт об обращении к БД или работе с коллекциями.

Что не так?

```
void leaveForTheSecondYear() {
 List<Student> naughty = repository.findNaughty();

 leaveForTheSecondYear(naughty);
}
```

Что не так?

```
void leaveForTheSecondYear() {
 List<Student> naughty = repository.findNaughty();

 List<Student> underAchieving = repository.findUnderAchieving();

 if (Settings.leaveBothCategories()) {
  leaveForTheSecondYear(naughty, underAchieving);
  return;
 }

 leaveForTheSecondYear(naughty);
}
```

Что не так: область видимости

```
void leaveForTheSecondYear() {
 List<Student> naughty = repository.findNaughty();

 List<Student> underAchieving = repository.findUnderAchieving();

 if (Settings.leaveBothCategories()) {
  leaveForTheSecondYear(naughty, underAchieving);
  return;
 }

 leaveForTheSecondYear(naughty);
}
```

Сужаем: -1 запрос в БД

```
void leaveForTheSecondYear() {
 List<Student> naughty = repository.findNaughty();


 if (Settings.leaveBothCategories()) {
  List<Student> underAchieving = repository.findUnderAchieving();
  leaveForTheSecondYear(naughty, underAchieving);
  return;
 }

 leaveForTheSecondYear(naughty);
}
```

Статический анализ: «ну не шмогла я!»

```
void leaveForTheSecondYear() {
 List<Student> naughty = repository.findNaughty();

 List<Student> underAchieving = repository.findUnderAchieving();

 if (Settings.leaveBothCategories()) {
  leaveForTheSecondYear(naughty, underAchieving);
  return;
 }

 leaveForTheSecondYear(naughty);
}
```

Эрзац-анализ: подсветка переменных

```
void leaveForTheSecondYear() {
 List<Student> naughty = repository.findNaughty();

 List<Student> underAchieving = repository.findUnderAchieving();

 if (Settings.leaveBothCategories()) {
  leaveForTheSecondYear(naughty, underAchieving);
  return;
 }

 leaveForTheSecondYear(naughty);
}
```

Совет 3

НЕ ДЕЛАЙ ЛИШНЮЮ РАБОТУ ТАМ, ГДЕ ЕЁ МОЖНО НЕ ДЕЛАТЬ.

Применение: обращай внимание на область видимости переменных.

Самый необычный пример

```
@Service
public class RemoteService {

 @Transactional(readOnly = true)
 public int countContracts(Dto dto) {
  if (dto.isInvalid()) {
   return -1;
  }
  return contractCounter.countContracts(dto);
 }

}
```

Внезапно

```
@Service
public class RemoteService {

 @Transactional(readOnly = true)
 public int countContracts(Dto dto) {
  if (dto.isInvalid()) {
   return -1;
  }
  return contractCounter.countContracts(dto);
 }

}
```

Прячем транзакционную логику внутрь

```
@Service
public class RemoteService {

// @Transactional(readOnly = true)
 public int countContracts(Dto dto) {
  if (dto.isInvalid()) {
   return -1;
  }
  return contractCounter.countContracts(dto);
 }

}
```

Время выполнения, нс

Потребление памяти, байт

Совет 4

РАЗДЕЛЯЙ И ВЛАСТВУЙ.

Применение: иногда грамотное разделение слоёв делает приложение более «отзывчивым».

До Java 8

```
SimpleDateFormat formatter = new SimpleDateFormat("dd.MM.yyyy");

String dateAsStr = formatter.format(date);
```

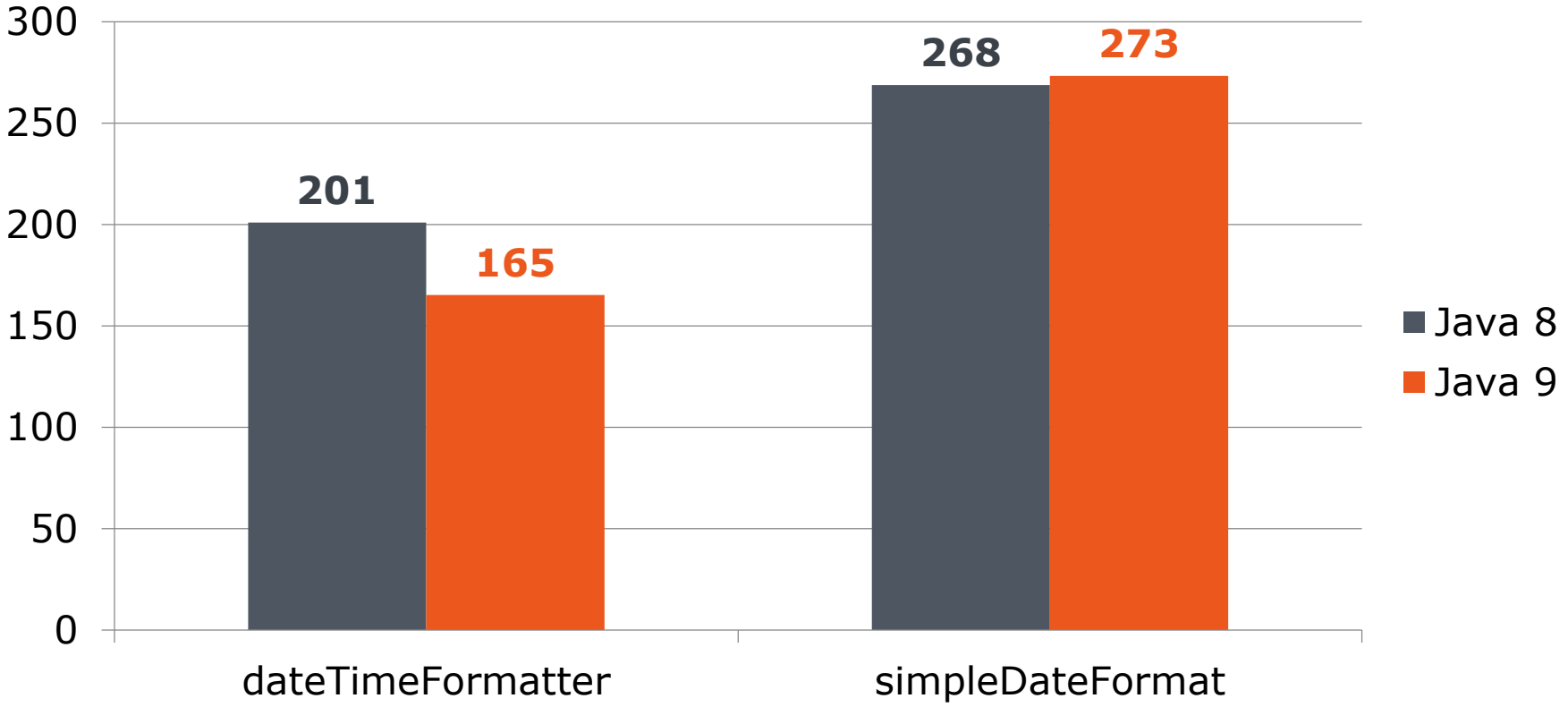
В Java 8 появился DateTimeFormatter

```
DateTimeFormatter formatter = ofPattern("dd.MM.yyyy");

String dateAsStr = formatter.format(localDate);
```

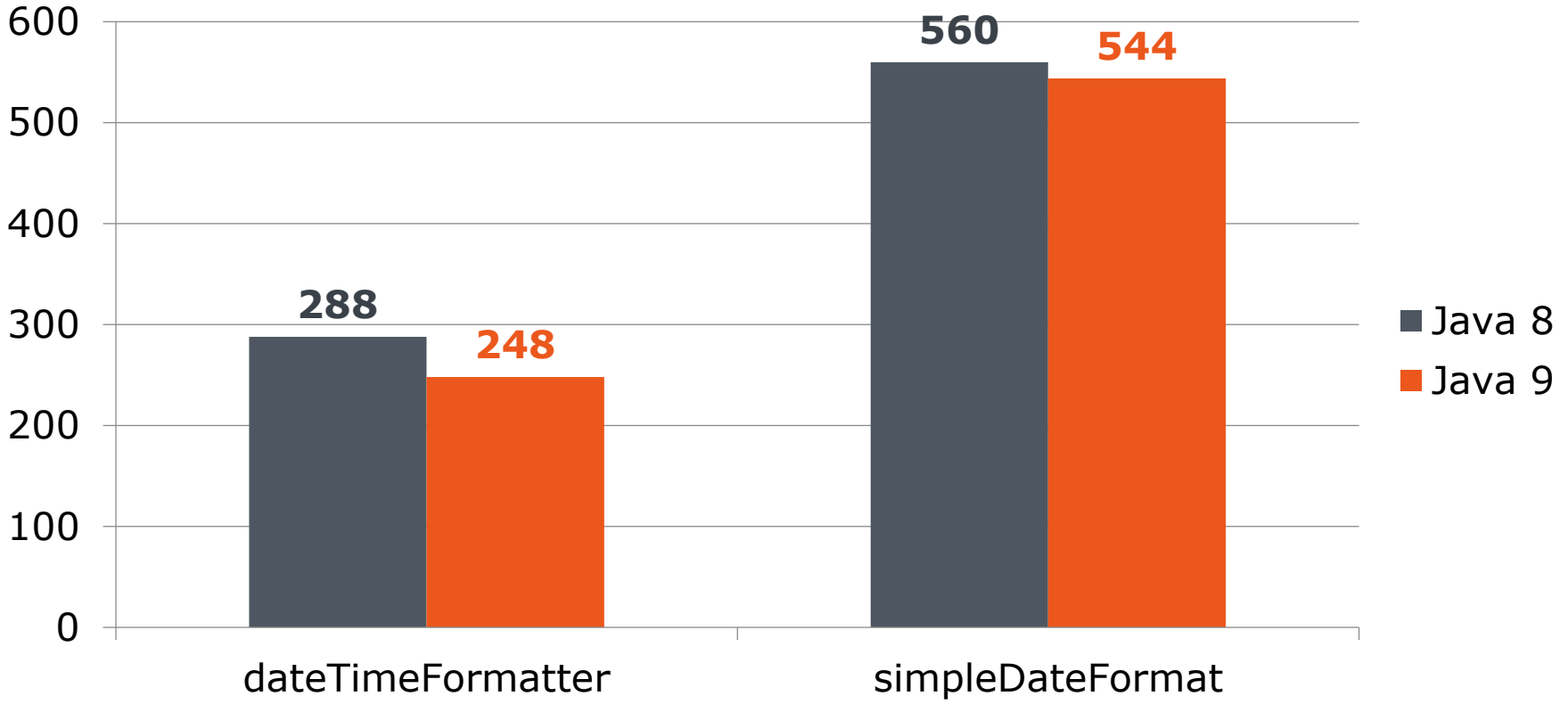
Измерим

```
@Benchmark
public String simpleDateFormat() {
 return sdf.format(date);
}

@Benchmark
public String dateTimeFormatter() {
 return dtf.format(localDate);
}

Date date = new Date();
LocalDate localDate = LocalDate.now();
SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd.MM.yyyy");
```

Время выполнения, нс



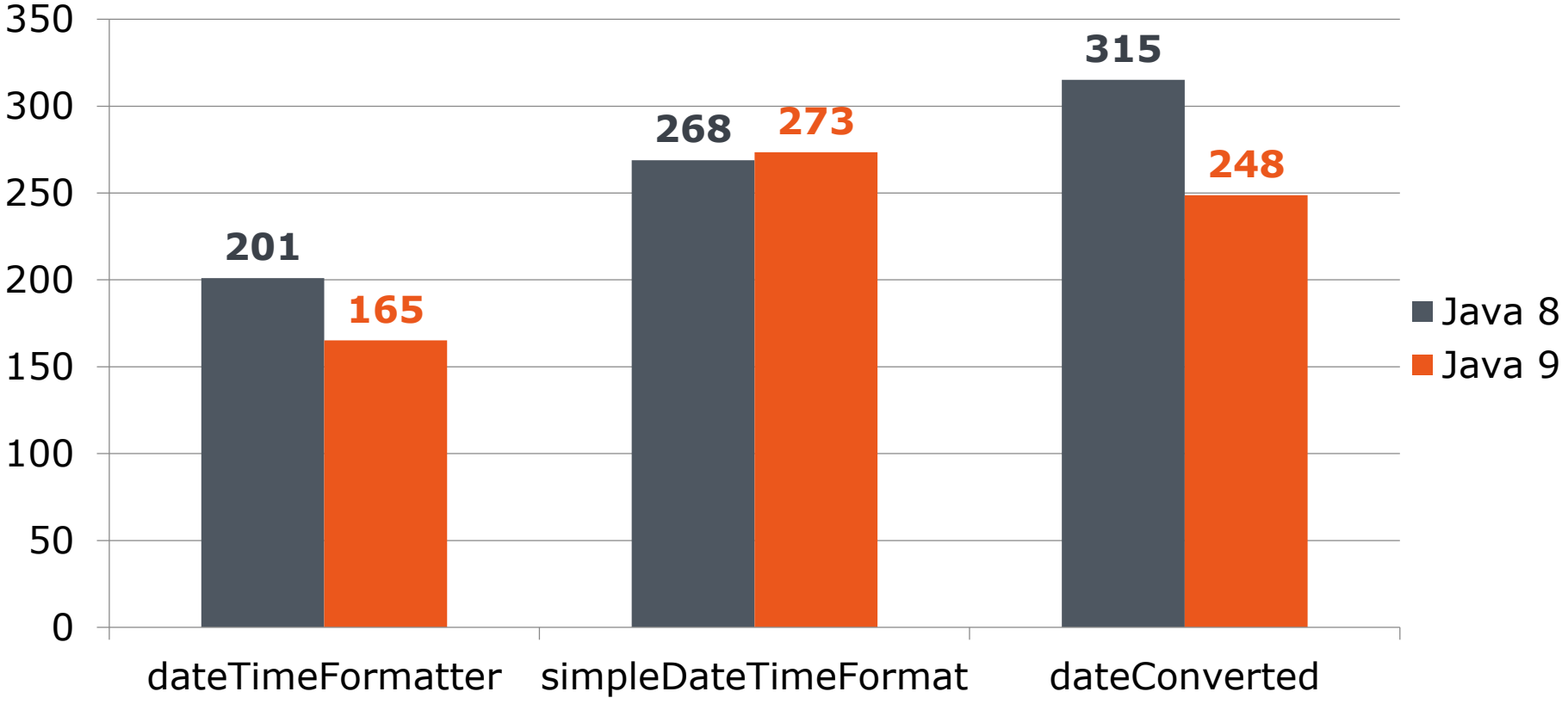
Потребление памяти, байт



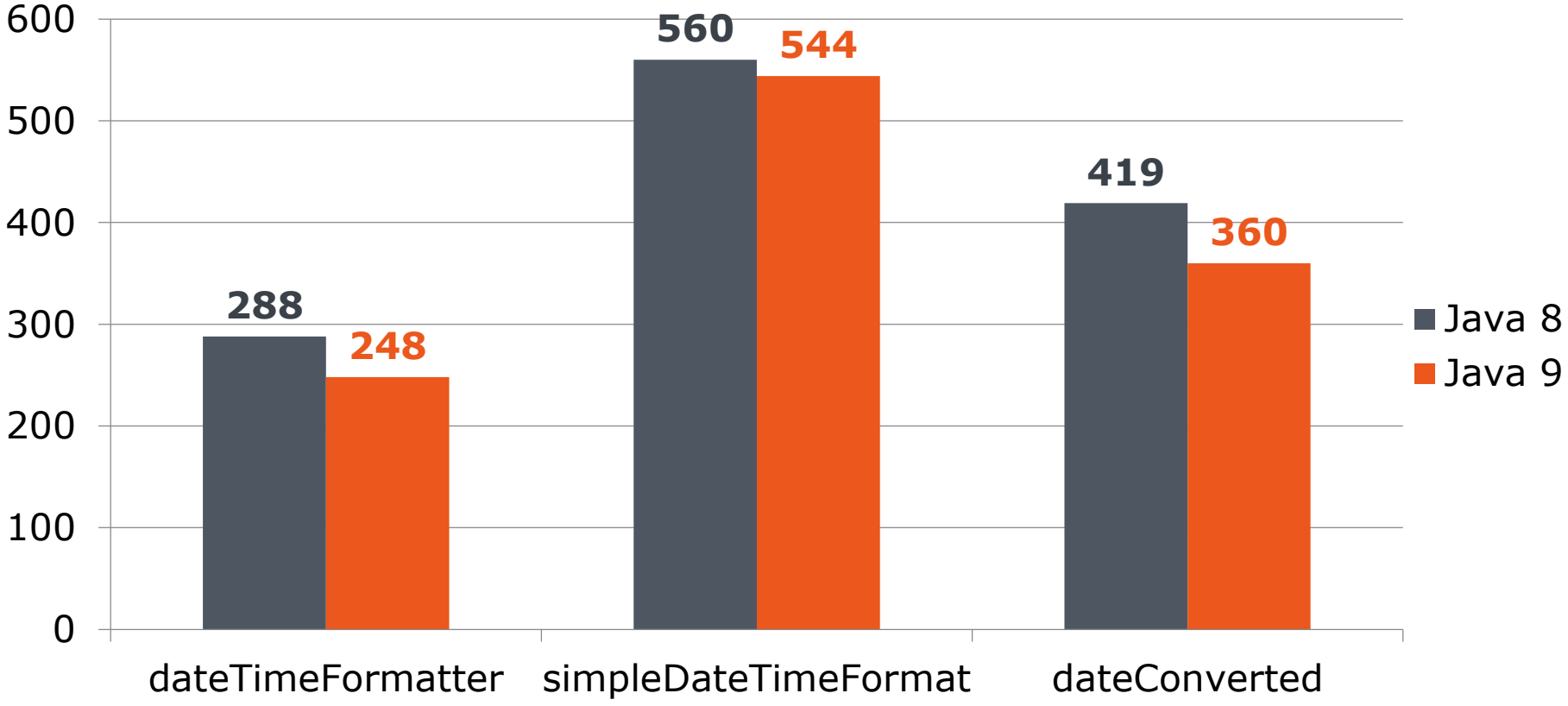
Преобразование java.util.Date

```
@Benchmark
public String measureDateConverted(Data data) {
 LocalDate localDate = toLocalDate(data.date);
 return data.dateTimeFormatter.format(localDate);
}

private LocalDate toLocalDate(Date date) {
 return date
   .toInstant()
   .atZone(ZoneId.systemDefault())
   .toLocalDate();
}
```

Преобразование j.u.Date (время, нс)



Преобразование j.u.Date (память, байт)



Совет 5

СМЕЛЕЕ ПРОБУЙТЕ.

Применение: читай release notes и не бойся пробовать что-то новое.

Ещё восьмёрка

```
Iterator<Long> iterator = items
 .stream()
 .map(func)
 .collect(toList())
 .iterator();

while (iterator.hasNext())
 bh.consume(iterator.next());
```

Избыточность

```
Iterator<Long> iterator = items
 .stream()
 .map(func)
 .collect(toList())
 .iterator();

while (iterator.hasNext())
 bh.consume(iterator.next());
```

Убираем избыточность

```
Iterator<Long> iterator = items
 .stream()
 .map(func)
 .collect(toList())
 .iterator();

while (iterator.hasNext())
 bh.consume(iterator.next());
```

```
Iterator<Long> iterator = items
 .stream()
 .map(func)
 .iterator();

while (iterator.hasNext())
 bh.consume(iterator.next());
```

Время выполнения, нс

```
items
  .stream()
  .map(func)
  .collect(toList())
  .iterator();
```

```
items
  .stream()
  .map(func)
  .iterator();
```

size	Score	Error	Score	Error	Units
10	628,2	± 11,5	802,8	± 9,6	ns/op
100	5570,1	± 36,0	7121,4	± 131,6	ns/op
1000	62056,4	± 2977,8	76045,7	± 1458,1	ns/op

Это как?

```
items
  .stream()
  .map(func)
  .collect(toList())
  .iterator();
```

```
items
  .stream()
  .map(func)
  .iterator();
```

size	Score	Error	Score	Error	Units
10	628,2	± 11,5	802,8	± 9,6	ns/op
100	5570,1	± 36,0	7121,4	± 131,6	ns/op
1000	**62056,4**	**± 2977,8**	**76045,7**	**± 1458,1**	ns/op

Сравним с Java 9, нс

```
items
  .stream()
  .map(func)
  .collect(toList())
  .iterator();
```

```
items
  .stream()
  .map(func)
  .iterator();
```

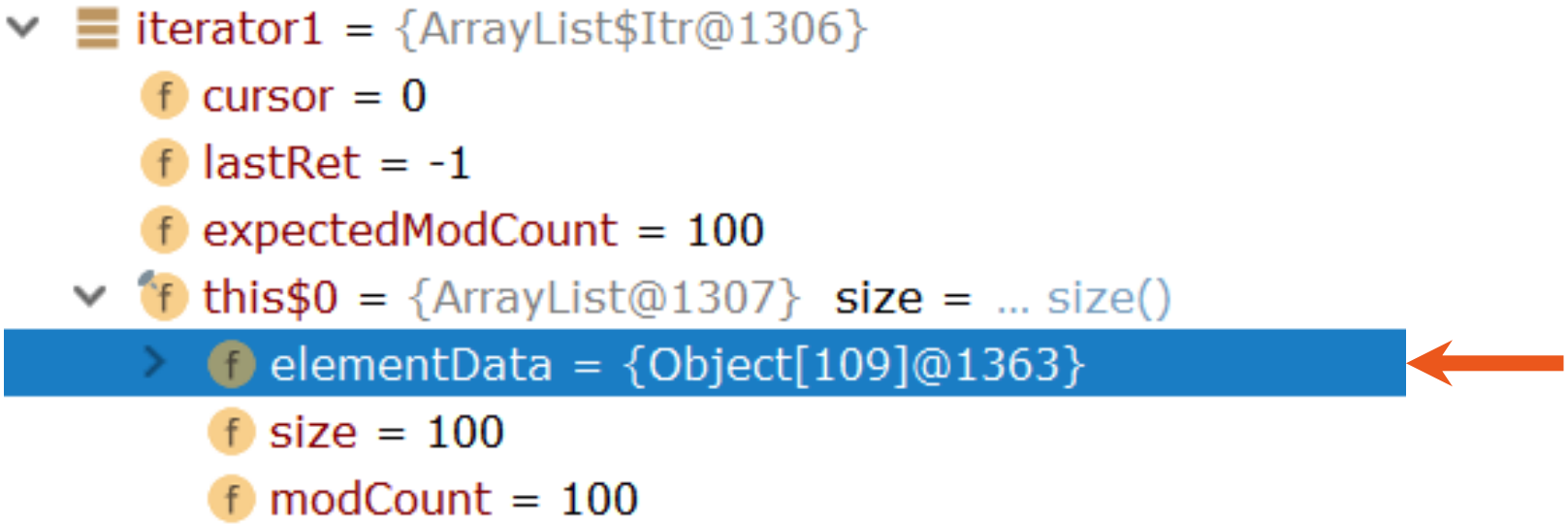
	size	Score	Score	Diff
Java 8	10	628,2	802,8	**+ 21,75 %**
	100	5570,1	7121,4	**+ 21,78 %**
	1000	62056,4	76045,7	**+ 18,40 %**
Java 9	10	485,2	599,2	**+ 19,03 %**
	100	3993,7	5220,3	**+ 23,50 %**
	1000	43252,5	49409,6	**+ 12,46 %**

Возьмём два итератора

```
Iterator iterator1 = items.stream().collect(toList()).iterator();

Iterator iterator2 = items.stream().iterator();
```

И посмотрим на них в отладчике

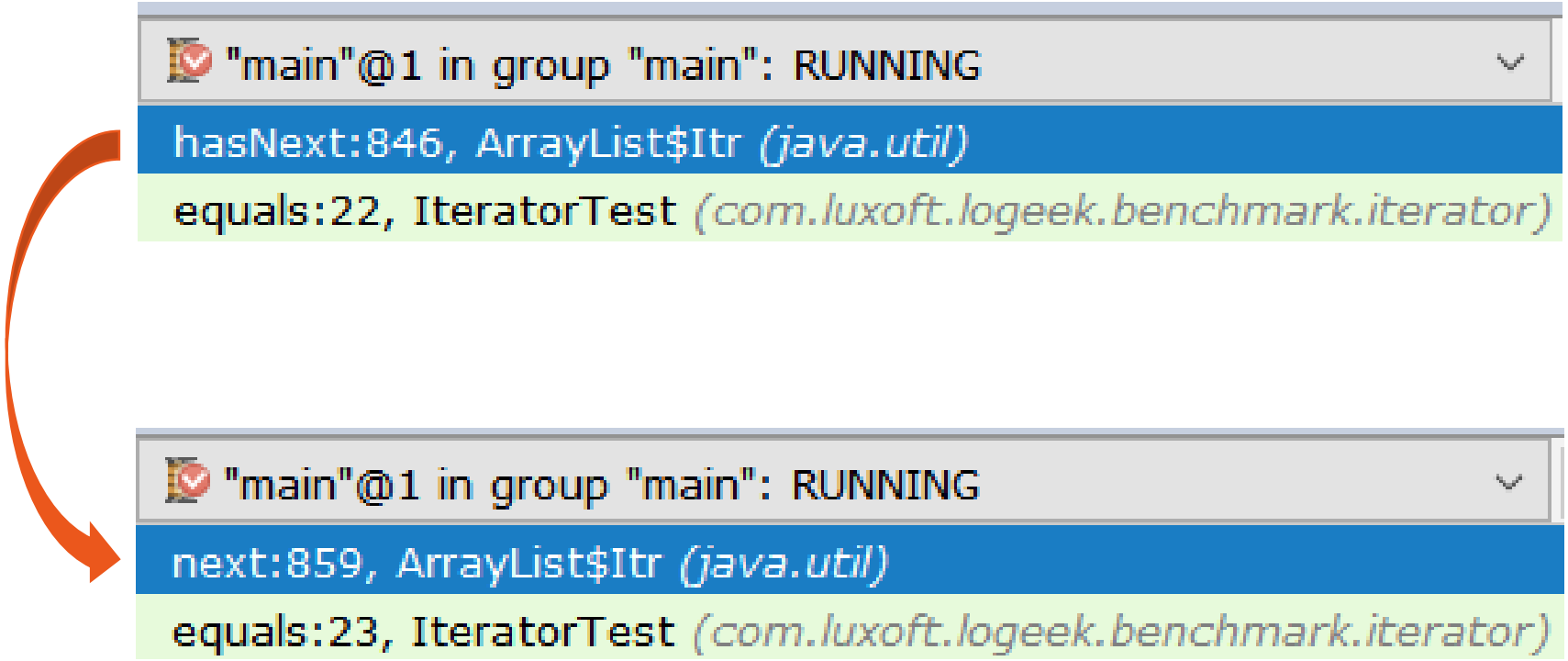
Итератор 1 – java.util.ArrayList$Itr



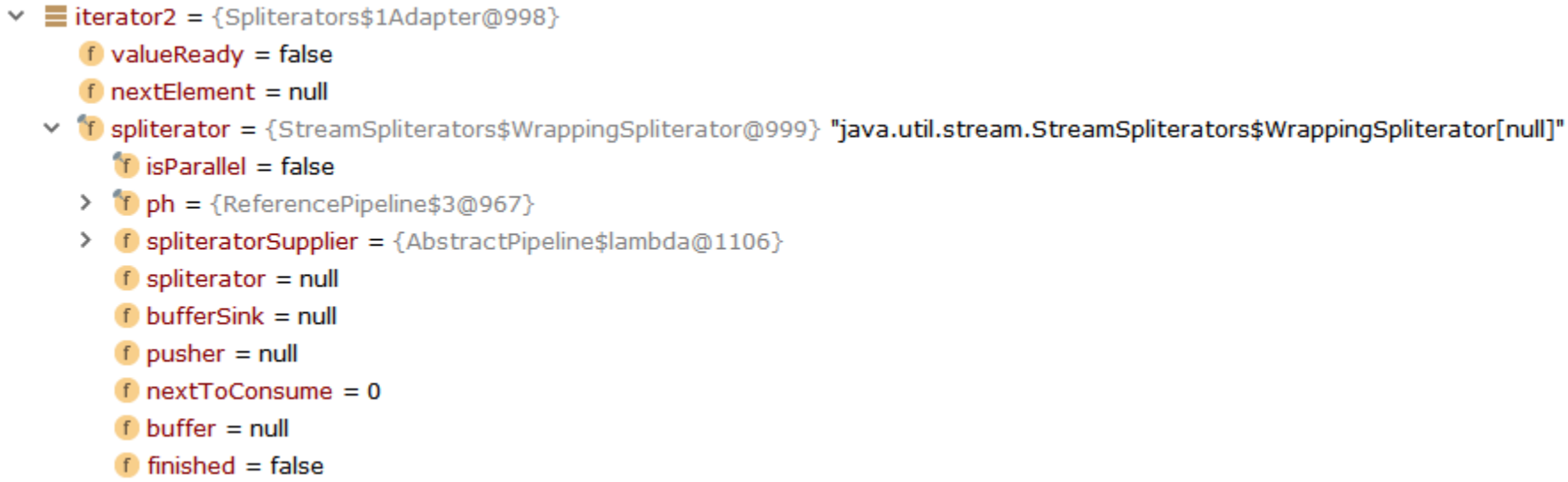
Проход по нему несложный

```
public boolean hasNext() {
 return cursor != size;
}

public E next() {
 checkForComodification();
 int i = cursor;
 if (i >= size) throw new NoSuchElementException();
 Object[] elementData = ArrayList.this.elementData;
 if (i >= elementData.length)
  throw new ConcurrentModificationException();
 cursor = i + 1;
 return (E) elementData[lastRet = i];
}
```

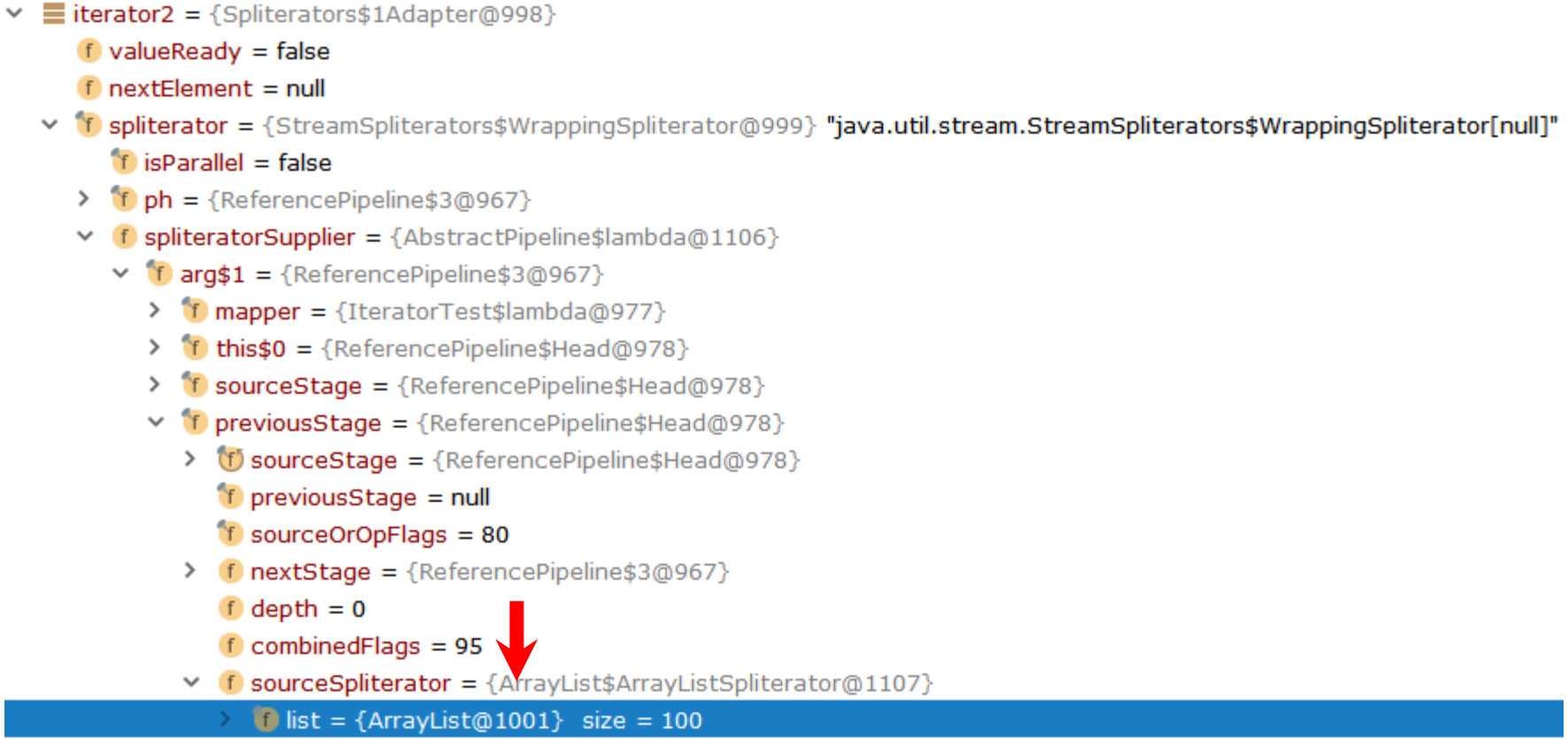
Проход по нему тоже несложный



Итератор 2 интереснее



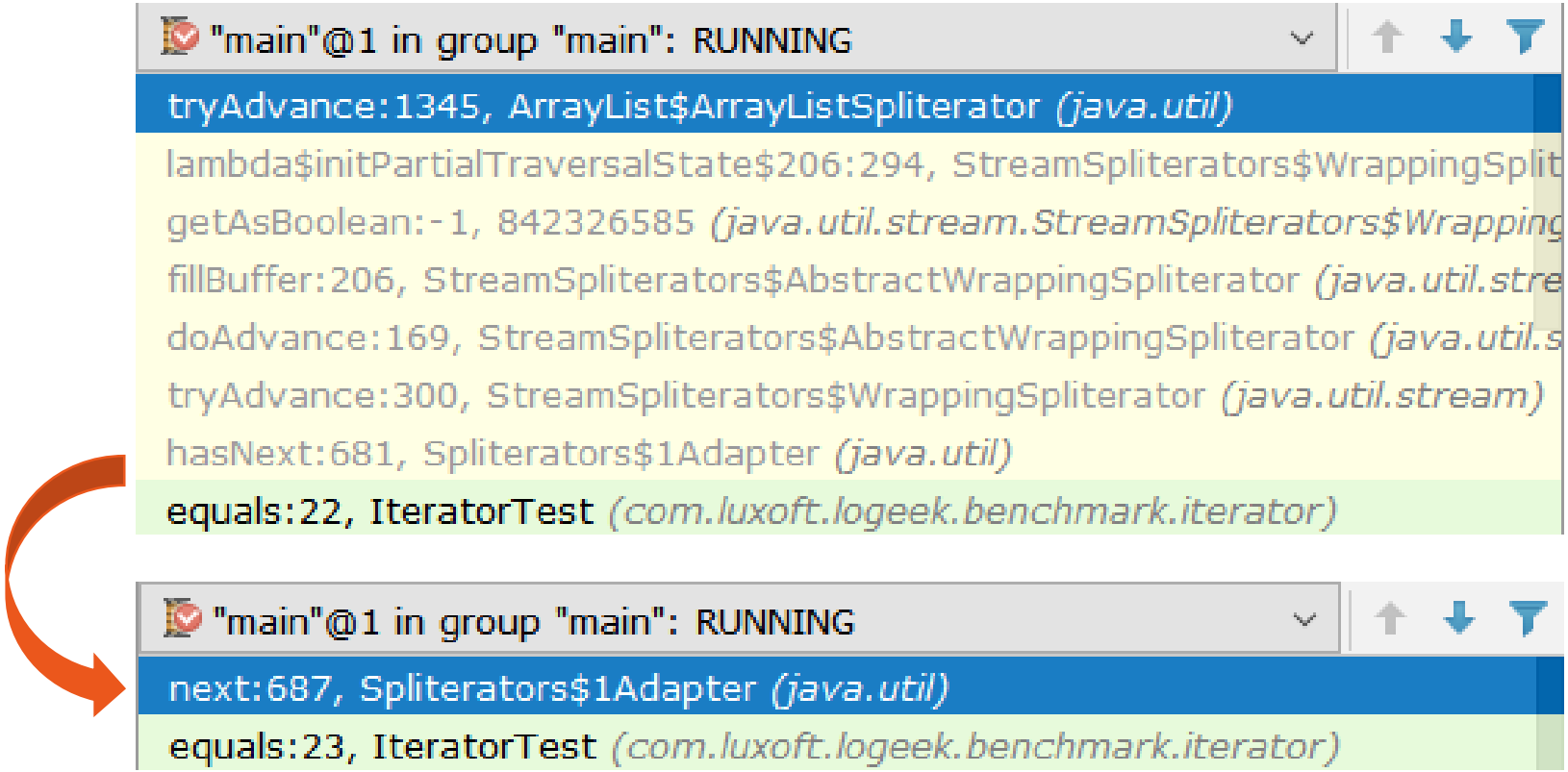
В его основе ArrayList$ArrayListSpliterator



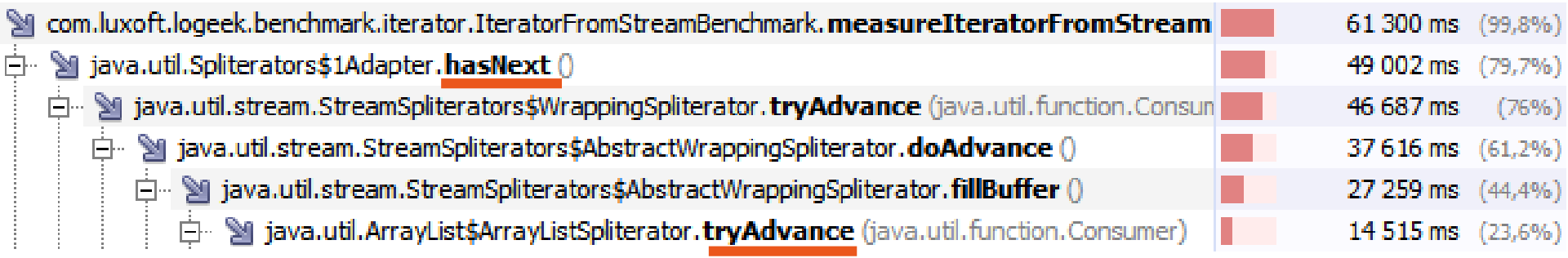
Проход по нему сложнее

```
public boolean hasNext() {
 if (!valueReady)
  spliterator.tryAdvance(this);
 return valueReady;
}

public T next() {
 if (!valueReady && !hasNext()) throw new NoSuchElementException();
 else {
  valueReady = false;
  return nextElement;
 }
}
```

Проход по нему сложнее



Проход по нему сложнее и дороже



Попробуем без итератора

```
Iterator itr =
 items
 .stream()
 .map(func)
 .collect(toList())
 .iterator();
```

```
Iterator itr =
 items
 .stream()
 .map(func)
 .iterator();
```

```
items
 .stream()
 .map(func)
 .forEach(bh::consume);
```

Попробуем без итератора

```
stream()
  .map(func)
  .collect(toList())
  .iterator();
```

	Java 8	Java 9	
size	Score	Score	Units
10	628,2	485,2	ns/op
100	5570,1	3993,7	ns/op
1000	62056,4	43252,5	ns/op

```
stream()
  .map(func)
  .iterator();
```

10	802,8	599,2	ns/op
100	7121,4	5220,3	ns/op
1000	76045,7	49409,6	ns/op

```
stream()
  .map(func)
  .forEach(bh::consume);
```

10	**539,6**	**416,4**	ns/op
100	**5009,9**	**3761,8**	ns/op
1000	**52946,0**	**34939,0**	ns/op

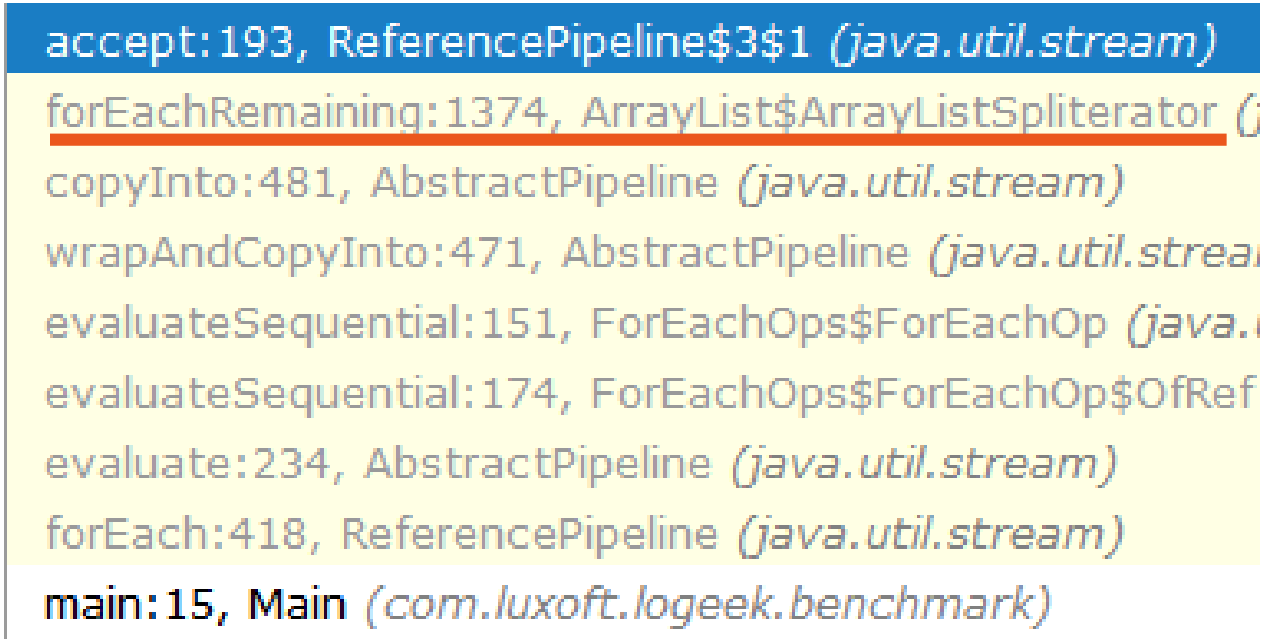
Почему стало быстрее?

Вроде бы, быстрее быть не должно, т. к. в основе тот же ArrayListSpliterator.

Почему стало быстрее?

Вроде бы, быстрее быть не должно, т. к. в основе тот же ArrayListSpliterator.

Но есть отличие:



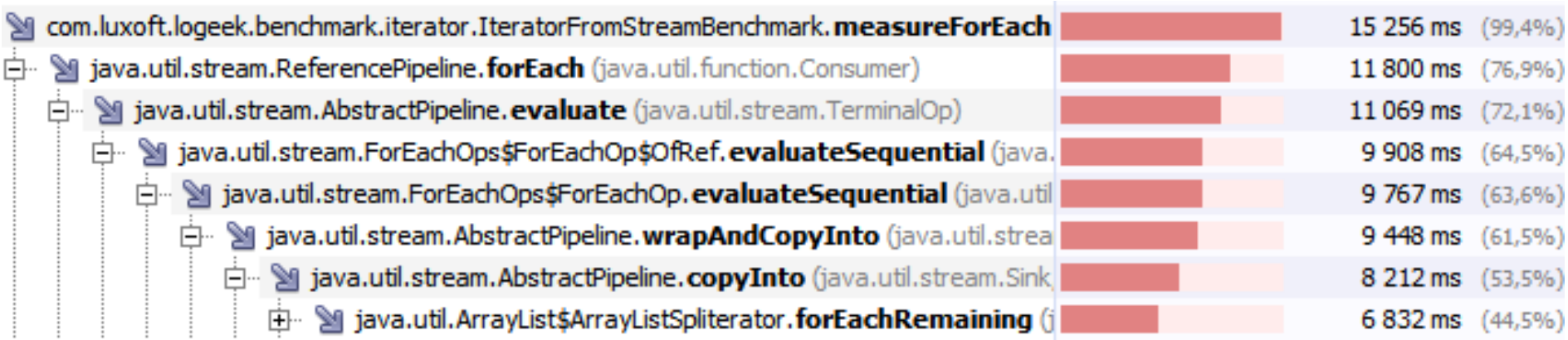
ArrayListSpliterator.forEachRemaining

```
void forEachRemaining(Consumer<? super E> action) {
 // …
 Object[] a = lst.elementData;
 if ((i = index) >= 0 && (index = hi) <= a.length) {
  for (; i < hi; ++i) {
   E e = (E) a[i];
   action.accept(e);
  }
  if (modCount == mc) return;
 }
 // …
}
```

Почему стало быстрее?

```
void forEachRemaining(Consumer<? super E> action) {
 // …
 Object[] a = lst.elementData;
 if ((i = index) >= 0 && (index = hi) <= a.length) {
  for (; i < hi; ++i) {
   E e = (E) a[i];
   action.accept(e);
  }
  if (modCount == mc) return;
 }
 // …
}
```

tryAdvance? Давай, до свидания!



Совет 6

ПРОВЕРЯЙ. ПРОВЕРЯЙ. ПРОВЕРЯЙ.

Применение: не гадай о том, как код может работать, а смотри внутрь и измеряй.

2 подвоха

```
StackTraceElement[] trace = th.getStackTrace();

StackTraceElement[] newTrace = Arrays
 .asList(trace)
 .subList(0, depth)
 .toArray(new StackTraceElement[newDepth]);
```

Подвох 1: протухшее «улучшение»

```
StackTraceElement[] trace = th.getStackTrace();

StackTraceElement[] newTrace = Arrays
 .asList(trace)
 .subList(0, depth)
 .toArray(new StackTraceElement[newDepth]);
```

Arrays of Wisdom of the Ancients

https://shipilev.net/blog/2016/arrays-wisdom-ancients/

Подвох 2: кто видит?

```
StackTraceElement[] trace = th.getStackTrace();

StackTraceElement[] newTrace = Arrays
  .asList(trace)
  .subList(0, depth)
  .toArray(new StackTraceElement[0]);
```

Подвох 2: избыточность

```
StackTraceElement[] trace = th.getStackTrace();

StackTraceElement[] newTrace = Arrays
  .asList(trace)
  .subList(0, depth)
  .toArray(new StackTraceElement[0]);
```

Что сделали?

```
asList(trace).subList(0, depth).toArray(new StackTraceElement[0]);
```

Завернули массив в список

Обрезали список

Развернули список в массив

Что нужно было сделать

```
asList(trace).subList(0, depth).toArray(new StackTraceElement[0]);
```

=

```
Arrays.copyOf(trace, depth);
```

Что нужно было сделать

```
asList(trace).subList(0, depth).toArray(new StackTraceElement[0]);
```

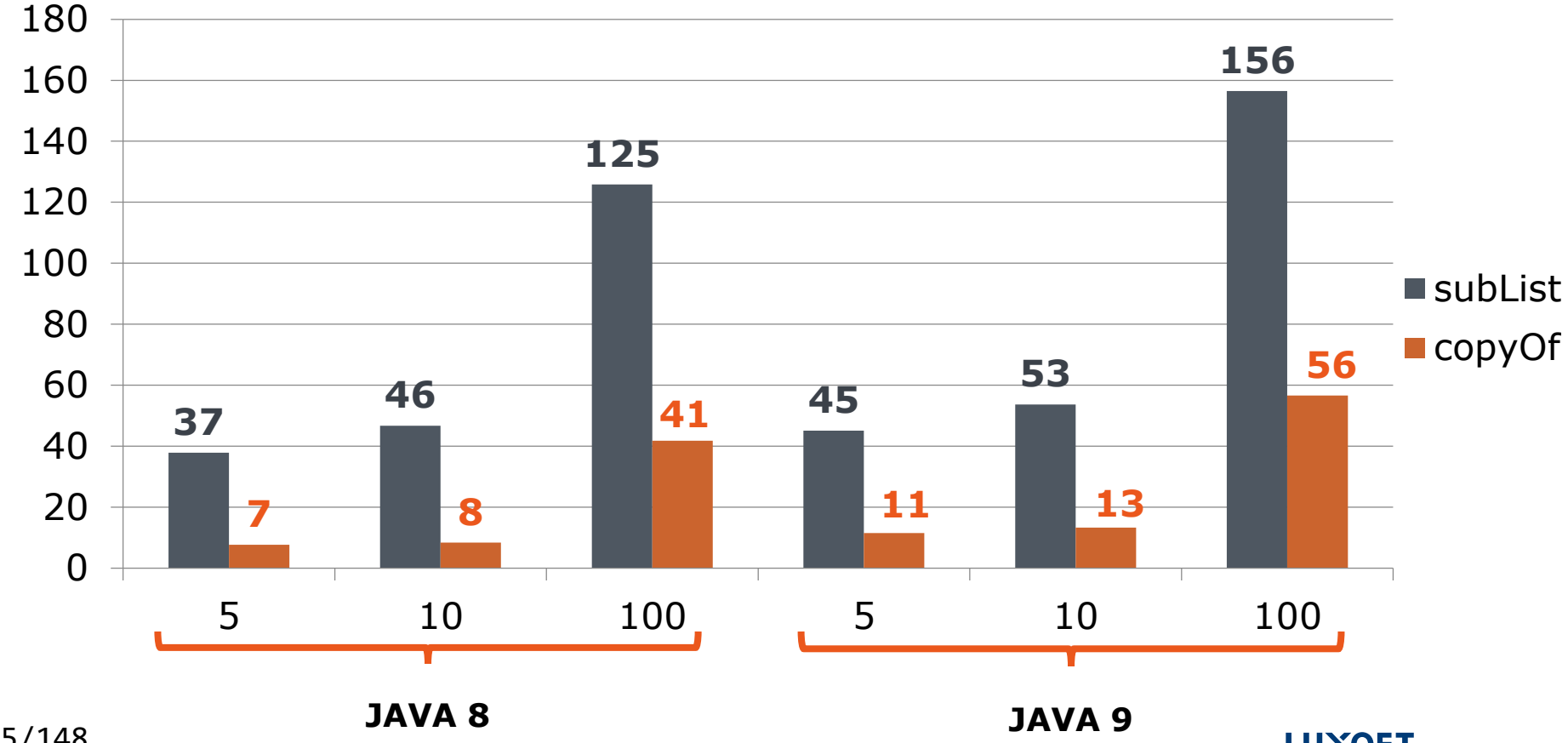
=

```
Arrays.copyOf(trace, depth);                    // отсчёт с 0
```

или

```
Arrays.copyOfRange(trace, 0, depth);
```

Время выполнения, нс



В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-182206
~~IDEA-182206~~ Simplification for Arrays.asList().sublist().toArray()

Fixed in: **2018.1**

https://youtrack.jetbrains.com/issue/IDEA-180847
~~IDEA-180847~~ Inspection 'Call to Collection.toArray with zero-length array argument' brings pessimization

Fixed in: **2018.1**

Что мы проверяем?

```
return list.stream().allMatch(set::contains);
```

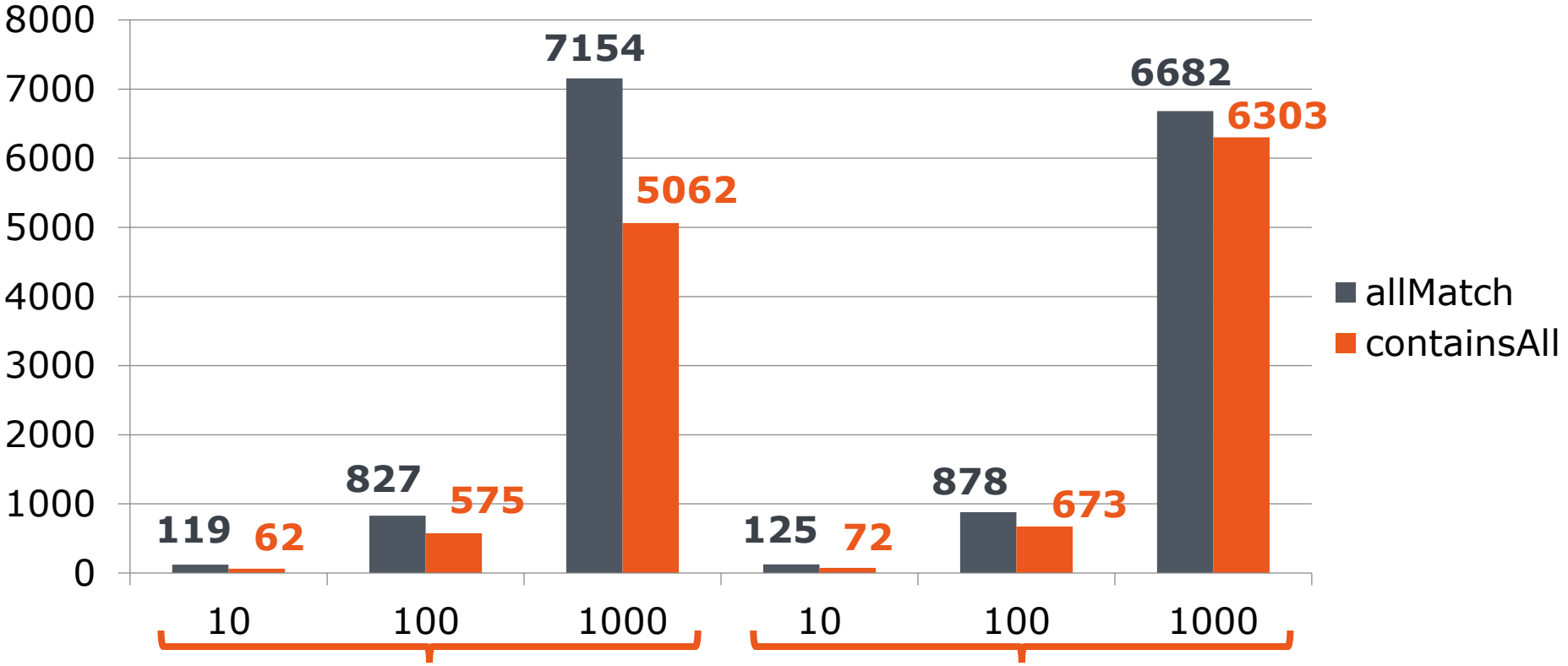
Острый стримоз головного мозга

```
return list.stream().allMatch(set::contains);
```

=

```
return set.containsAll(list);
```

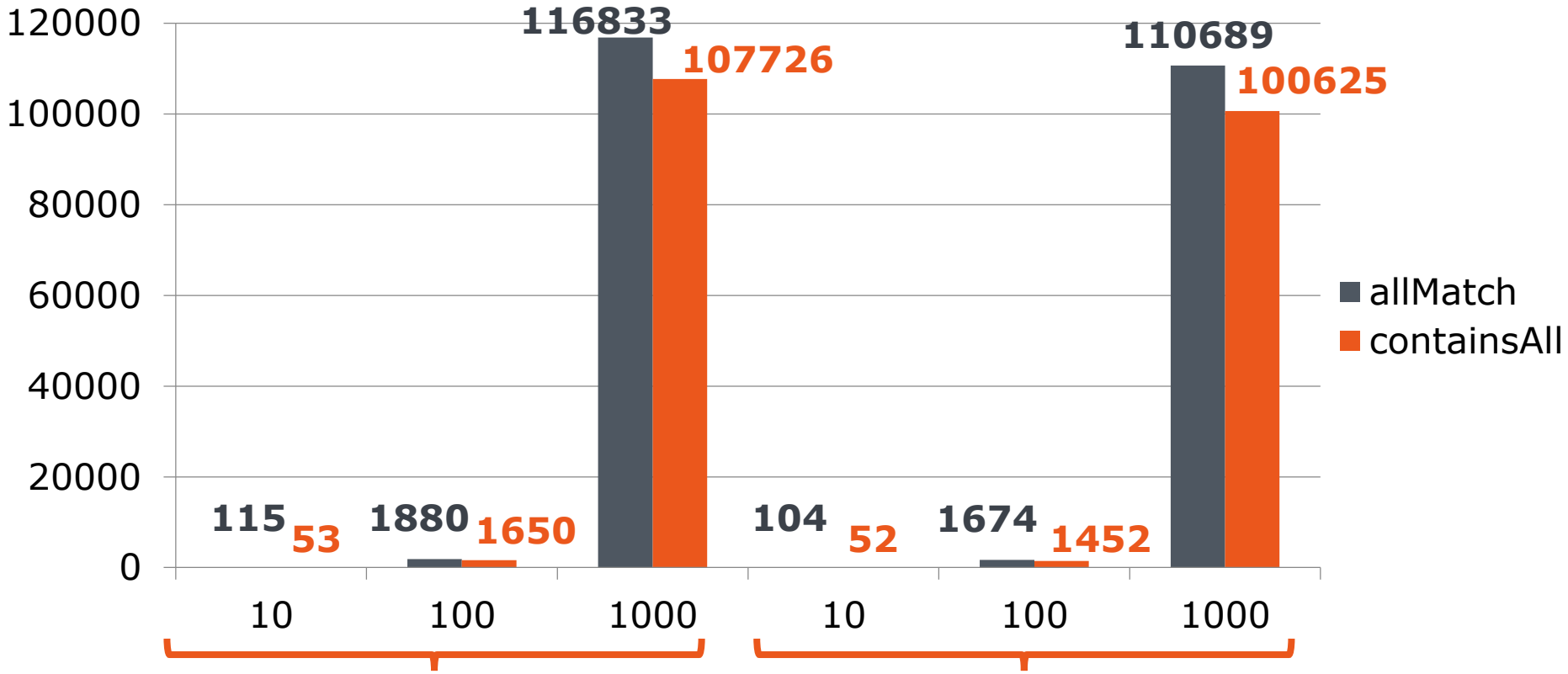
HashSet, время выполнения, нс



HashSet, память

		Java 8	Java 9	
streamAllMatch:·gc.alloc.rate.norm	1	160	184	B/op
containsAll:·gc.alloc.rate.norm	1	≈ 0	≈ 0	B/op
streamAllMatch:·gc.alloc.rate.norm	10	160	184	B/op
containsAll:·gc.alloc.rate.norm	10	≈ 0	≈ 0	B/op
streamAllMatch:·gc.alloc.rate.norm	100	160	184	B/op
containsAll:·gc.alloc.rate.norm	100	≈ 0	≈ 0	B/op
streamAllMatch:·gc.alloc.rate.norm	1000	160	184	B/op
containsAll:·gc.alloc.rate.norm	1000	32	≈ 0	B/op

ArrayList, время выполнения, нс



ArrayList, память

		Java 8	Java 9	
streamAllMatch:·gc.alloc.rate.norm	1	160	184	B/op
containsAll:·gc.alloc.rate.norm	1	≈ 0	≈ 0	B/op
streamAllMatch:·gc.alloc.rate.norm	10	160	184	B/op
containsAll:·gc.alloc.rate.norm	10	≈ 0	≈ 0	B/op
streamAllMatch:·gc.alloc.rate.norm	100	160	184	B/op
containsAll:·gc.alloc.rate.norm	100	≈ 0	≈ 0	B/op
streamAllMatch:·gc.alloc.rate.norm	1000	162	175	B/op
containsAll:·gc.alloc.rate.norm	1000	17	40	B/op

В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-181928

~~IDEA-181928~~ Stream.allMatch(Collection::contains) can be simplified to Collection.containsAll()

Fixed in: **2018.1**

Есть такая штука

```
interface FileNameLoader {
 String[] loadFileNames();
}
```

И её использование

```
interface FileNameLoader {
 String[] loadFileNames();
}

private FileNameLoader loader;

void load() {
 for (String str : asList(loader.loadFileNames())) {
  use(str);
 }
}
```

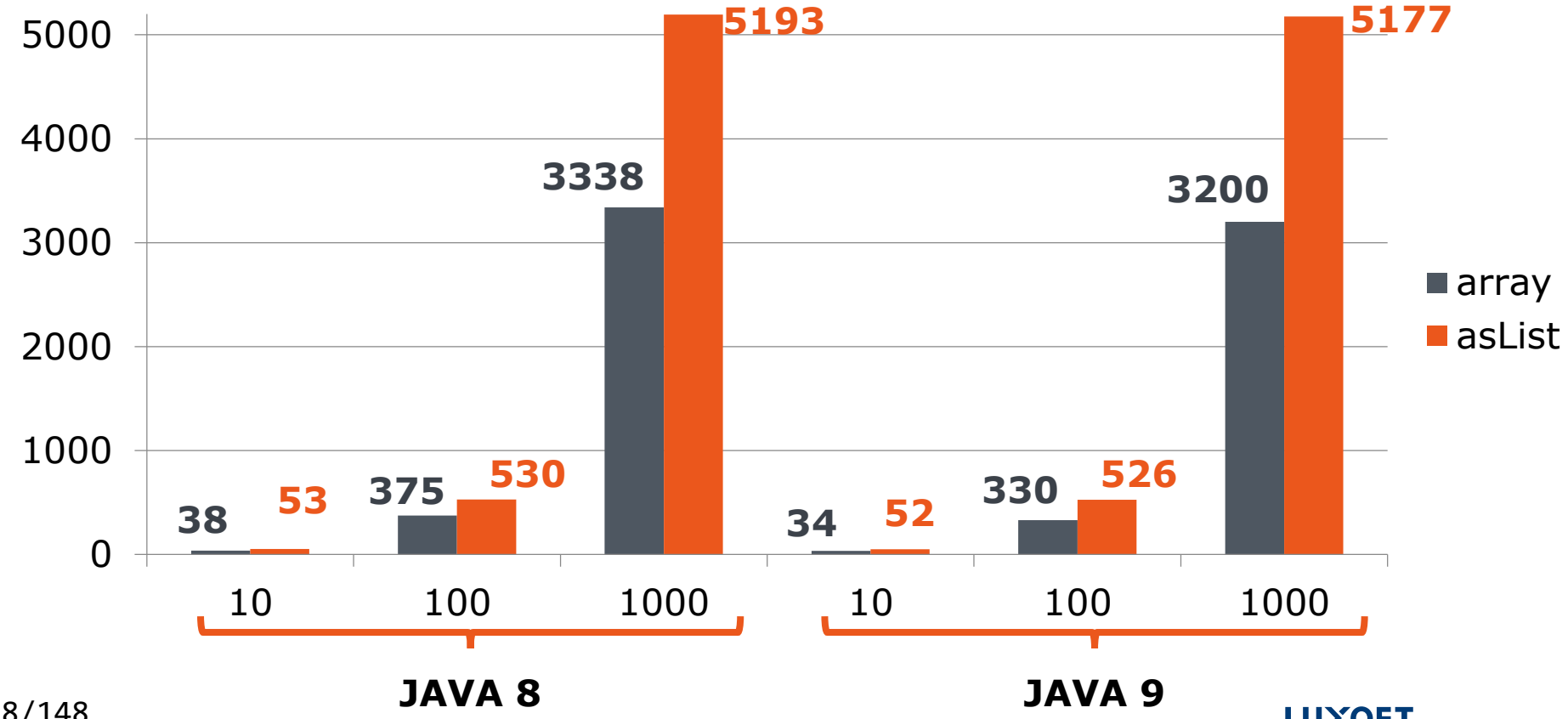
Избыточность

```
interface FileNameLoader {
 String[] loadFileNames();
}

private FileNameLoader loader;

void load() {
 for (String str : asList(loader.loadFileNames())) {
  use(str);
 }
}
```

Ничего лишнего

```
interface FileNameLoader {
 String[] loadFileNames();
}

private FileNameLoader loader;

void load() {
 for (String str : loader.loadFileNames()) {
  use(str);
 }
}
```

Время выполнения, нс



В сухом остатке

https://youtrack.jetbrains.com/issue/IDEA-184240
~~IDEA-184240~~ Unnecessary array-to-collection wrapping should be detected

Fixed in: **2018.1**

Совет 7

НЕ УСЛОЖНЯЙ.

Применение: чем сложнее код, тем вероятнее ошибка [и низкая производительность].

ВЫВОДЫ

Более производительный код как правило:

- более краткий
- более надёжный
- более простой для понимания

ПРИМЕРЫ

```
List<T> list = new ArrayList<>();
for (T item : set) list.add(item);
```

```
List<T> list =
 new ArrayList<>(set);
```

ПРИМЕРЫ

```
List<T> list = new ArrayList<>();
for (T item : set) list.add(item);

Iterator<T> itr = list.stream()
 .iterator();
while (itr.hasNext())
 consumer.consume(itr.next());
```

```
List<T> list =
 new ArrayList<>(set);

list.stream()
 .forEach(consumer);
```

ПРИМЕРЫ

```
List<T> list = new ArrayList<>();
for (T item : set) list.add(item);
```

```
List<T> list =
 new ArrayList<>(set);
```

```
Iterator<T> itr = list.stream()
 .iterator();
while (itr.hasNext())
 consumer.consume(itr.next());
```

```
list.stream()
 .forEach(consumer);
```

```
list.stream.allMatch(set::contains);
```

```
set.containsAll(list);
```

ПРИМЕРЫ

```
List<T> list = new ArrayList<>();
for (T item : set) list.add(item);

Iterator<T> itr = list.stream()
 .iterator();
while (itr.hasNext())
 consumer.consume(itr.next());

list.stream.allMatch(set::contains);

Arrays.asList(arr)
 .subList(0, 9)
 .toArray(new T[0]);
```

```
List<T> list =
 new ArrayList<>(set);

list.stream()
 .forEach(consumer);

set.containsAll(list);

Arrays.copyOf(arr, 9);
```

В сухом остатке (уже работает)

~~IDEA-138456~~ 'Collection.addAll() -> parametrised constructor' (**142.1217**)

~~IDEA-165942~~ Replace method call in a loop with bulk operation (**2017.1**)

~~IDEA-178761~~ 'Collection.addAll() can be replaced with parametrized constructor' should be active by default (**2017.3**)

~~IDEA-182206~~ Simplifiable Arrays.asList().sublist().toArray() (**2018.1**)

~~IDEA-181928~~ Stream.allMatch(Coll::contains) -> Coll.containsAll() (**2018.1**)

~~IDEA-184240~~ Detect pointless array-to-collection wrapping (**2018.1**)

~~IDEA-180847~~ Wrong 'Call to Collection.toArray with empty array' (**2018.1**)

В сухом остатке (ещё пилится)

IDEA-170178 Compare collections at the very end of IDEA-generated equals()

IDEA-177466 Detect List.remove(int index) called in a loop

IDEA-165730 Warn about inefficient usages of JpaRepository

Lombok-1543 [Feature request] Reordering of comparison in @EqualsAndHashCode for better performance #1543

KT-23184 Auto-generated equals() of data class is not optimal

БЛАГОДАРЮ ЗА ВНИМАНИЕ

sergei.tsypanov@yandex.ru