Методы класса Object

(C) Класс Animal

```
void voice() {
    System.out.print('Голос');
};
```

extends

extends

extends

(C) Класс Cat

```
@Override
void voice() {
    System.out.print('Мяу!');
};
```

(C) Класс Dog

```
@Override
void voice() {
    System.out.print('Гав!');
};
```

(C) Класс Cow

```
@Override
void voice() {
    System.out.print('Муу!');
};
```

Интерфейс

Когда требуется дополнительное поведение.

Например, некоторые животные умеют кусаться...

методы класса Object

getClass

toString

equals

hashcode

clone

finalize

wait, notify, notifyAll

getClass

Возвращает класс в рантайме

```
System.out.println(animal.getClass());
```

class com.object.intro.Animal (имя пакета + имя класса)

Используется:
- в методе toString,
- можно накрутить дополнительную логику
- используется в рефлексии

toString

Строковое представления объекта

Реализация по умолчанию: **имя класса@хешкод в 16ричном виде**

com.object.intro.Animal@4b67cf4d

Желательно переопределить для логов и анализа проблем.

equals

Проверяет равенство двух объектов

Базовая реализация проверяет **равенство двух ссылок**

Используется в работе коллекций и неправильная работа equals приведет к неправильной работе коллекций

equals

Правильную работу метода обеспечивает выполнение правил:

Рефлексивность
для любого заданного значения `x`, выражение `x.equals(x)` должно возвращать `true`

Симметричность
для любых заданных значений `x` и `y`, `x.equals(y)` должно возвращать `true` только в том случае, когда `y.equals(x)` возвращает `true`

Транзитивность
для любых заданных значений x, y и z, если x.equals(y) возвращает true и y.equals(z) возвращает true, x.equals(z) должно вернуть значение true

Согласованность
для любых заданных значений x и y повторный вызов x.equals(y) будет возвращать значение предыдущего вызова этого метода при условии, что поля, используемые для сравнения этих двух объектов, не изменялись между вызовами.

Сравнение null
для любого заданного значения x вызов x.equals(null) должен возвращать false

equals

Общий алгоритм определения equals

1. Проверить на **равенство ссылки объектов `this`** и параметра метода `o`.
   `if (this == o) return true;`
2. Проверить, определена ли ссылка `o`, **т. е. является ли она `null`**.
   Если в дальнейшем при сравнении типов объектов будет использоваться оператор `instanceof`, этот пункт можно пропустить, т. к. этот параметр возвращает `false` в данном случае `null instanceof Object`.
3. Сравнить типы объектов `this` и `o` с помощью оператора **`instanceof`** или метода **`getClass()`**, руководствуясь описанием выше и собственным чутьем.
4. Если метод `equals` переопределяется в подклассе, не забудьте сделать вызов `super.equals(o)`
5. Выполнить преобразование типа параметра `o` к требуемому классу.
6. Выполнить сравнение всех значимых полей объектов:
   - для примитивных типов (кроме `float` и `double`), используя оператор `==`
   - для ссылочных полей необходимо вызвать их метод `equals`
   - для массивов можно воспользоваться перебором по циклу, либо методом `Arrays.equals()`
   - для типов `float` и `double` необходимо использовать методы сравнения соответствующих оберточных классов `Float.compare()` и `Double.compare()`
7. И, наконец, ответить на три вопроса: является ли реализованный метод симметричным? Транзитивным? Согласованным? Два других принципа (рефлексивность и определенность), как правило, выполняются автоматически.

hashcode

Уникальный идентификатор объекта в системе

Базовая реализация возвращает **случайный int**

Используется в работе коллекций и неправильная работа hashcode приведет к неправильной работе хеш таблиц

hashcode

Правильно определенный метод

- Повторный вызов hashcode возвращает одно и то же значение
- Два объекта с разными хеш-кодами гарантированно разные
- Два объекта с одинаковыми хеш-кодами не гарантированно равны

equals и hashcode

Необходимо переопределять оба метода одновременно

1. **Объекты равны**
   x.equals(y) - **true**
   x.hashcode **==** y.hashcode

2. **Объекты разные**
   x.equals(y) - **false**
   x.hashcode **!=** y.hashcode

3. **Объекты не равны**
   x.equals(y) - **false**
   x.hashcode **==** y.hashcode **- Коллизия!**

Нарушение связанности:

- Если не переопределить equals - поломаем List
- Если не переопределить hashcode - поломаем hash-table (Set и Map)

clone

Клонирует объект

Базовая реализация - **поверхностное копирование**

- для параметров-объектов копируется только ссылка
- требуется имплементация интерфейса-маркера Cloneable

Лучше переопределить метод с реализацией **глубокого копирования**

- создание новых объектов
- требуется имплементация интерфейса-маркера Cloneable

А еще лучше **сериализация**

- сохранение объекта в байтовый поток
- требуется имплементация интерфейса-маркера Serializable

finalize

Вызывается перед удалением объекта **Сборщиком Мусора**

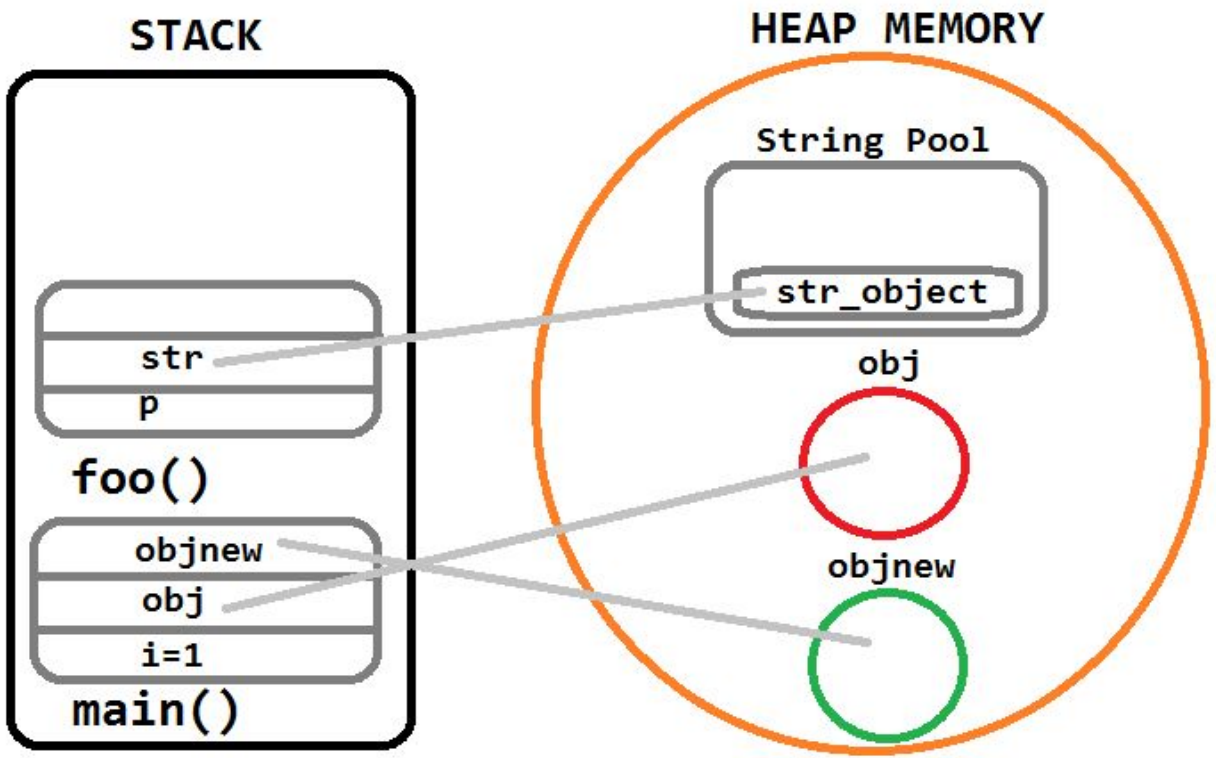
Базовая имплементация - пустой

Служит для освобождения ресурсов, которые используются объектом (например, подключение в Базе Данных)

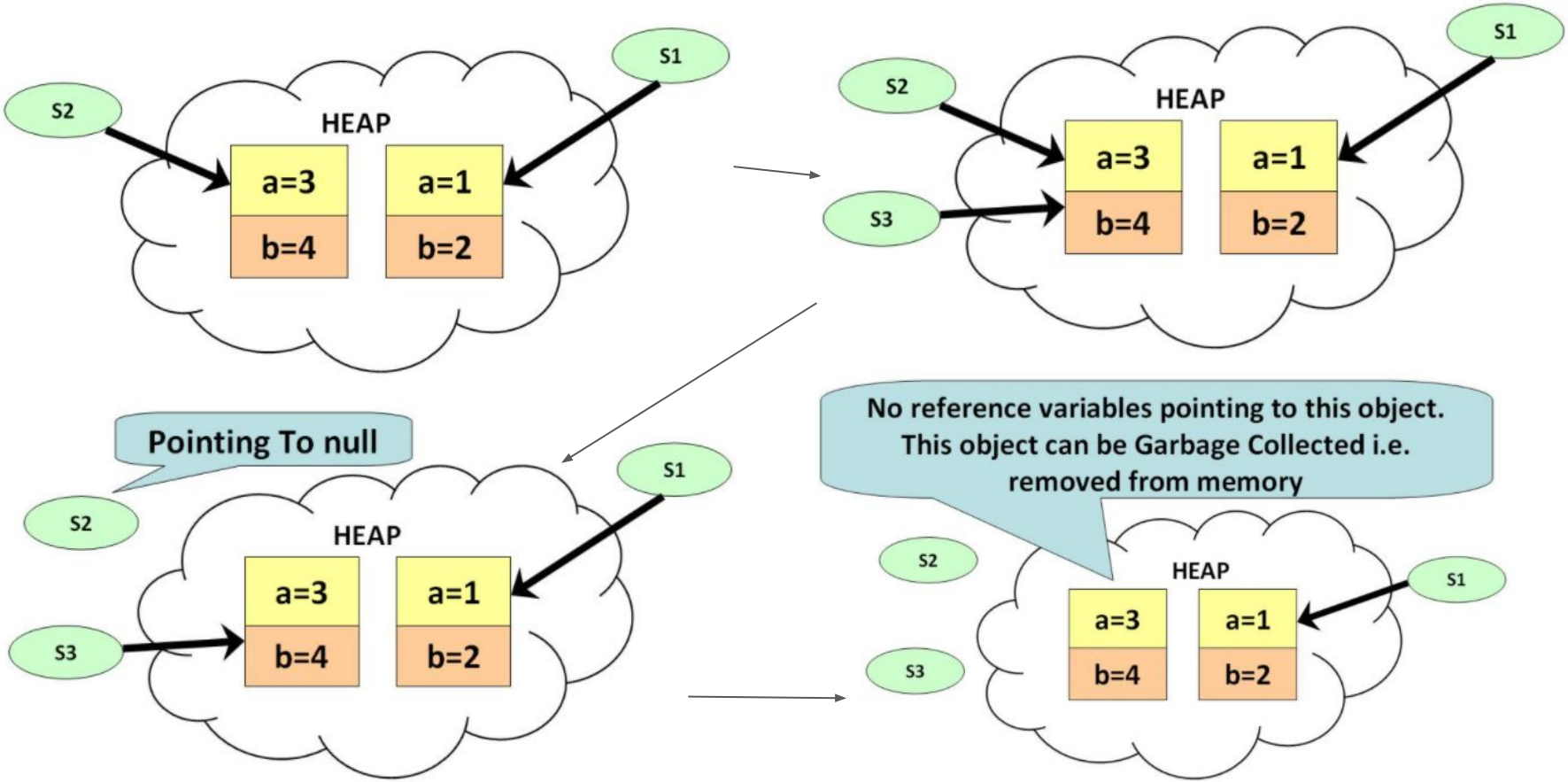
Рекомендации по использованию - **не использовать**

- Нет гарантий что сборщик мусора уничтожит объект сразу после вызова
- Deprecated в Java 9+

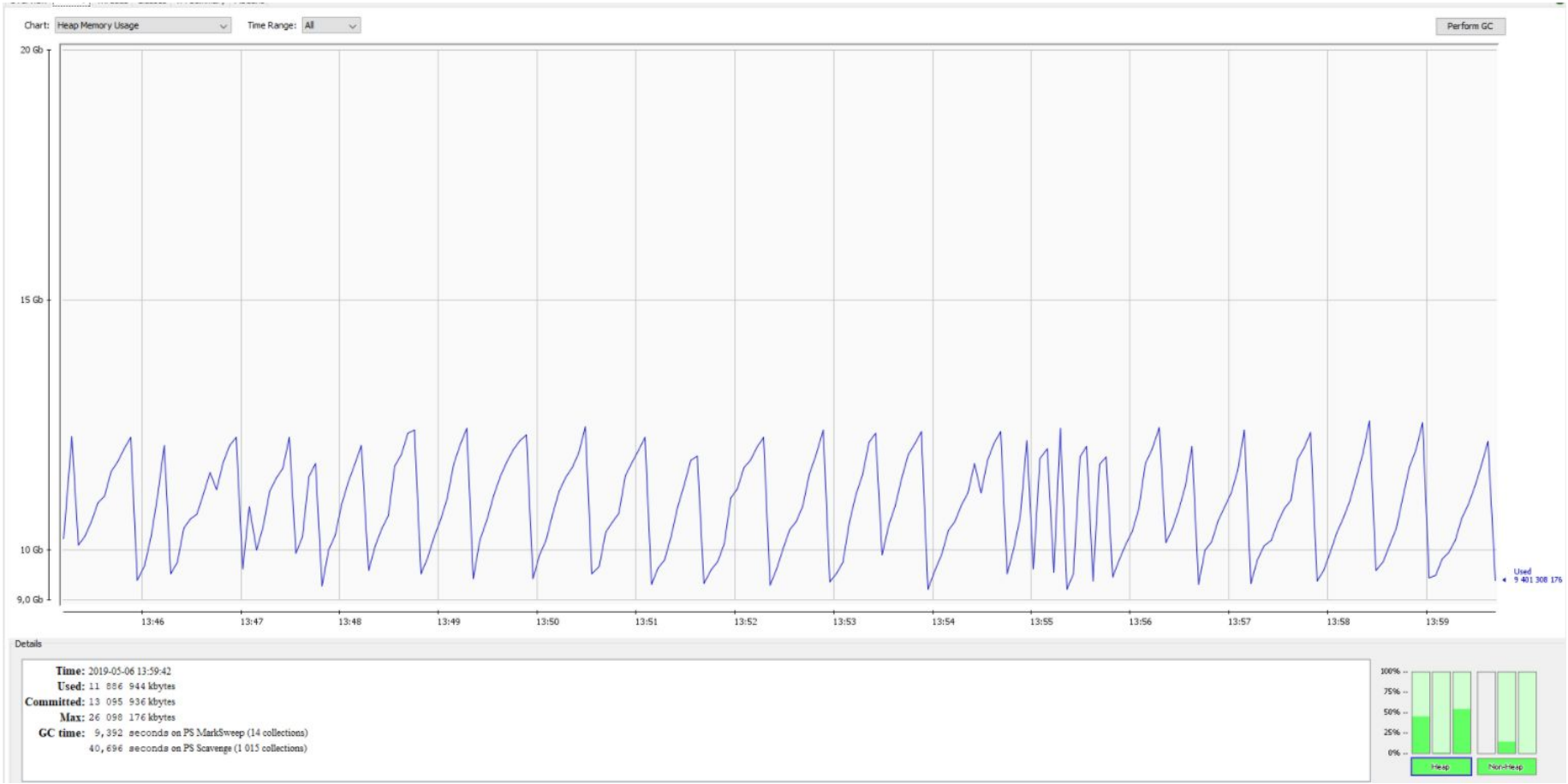
Память в Java



Сборщик мусора



Сборщик мусора



wait, notify, notifyAll

Методы для синхронизации потоков

wait - перевести поток в режим ожидания

notify - “разбудить” последний “уснувший” поток

notifyAll - “разбудить” все “спящие” потоки