Язык программирования Java

Лекция 5

Перевод курса CS11 Java Track

Содержание

Потоки в Java

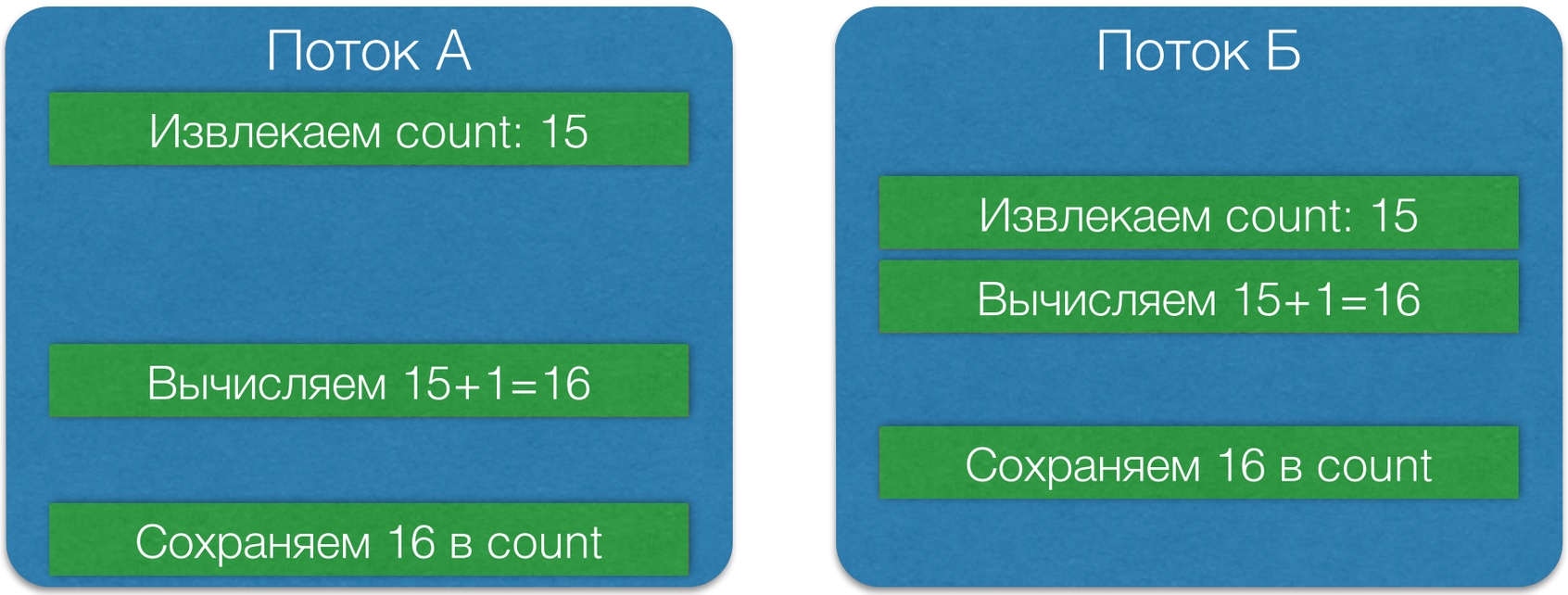
- Поток это последовательно исполняющаяся цепочка инструкций программы
  - Потоки имеют начало и конец
  - Поток может одновременно делать только что то одно
- Каждая программа имеет, по крайней мере, один поток
  - Это поток называется основным потоком программ. В этом потоке исполняется метод main()
- Многозадачные программы имеют несколько потоков
  - Они могут делать разные вещи “одновременно”

Стандартные потоки Java

- Виртуальная машина Java использует для работы несколько потоков
  - Основной поток исполняет вашу программу
  - Отдельный поток может создаваться для “сборки мусора”
  - Java AWT/Swing запускает поток для обработки событий
- Некоторые библиотечные классы Java используют потоки
  - Вы можете создавать и запускать свои собственные потоки. Но (увы!) на этот раз мы этого делать не будем

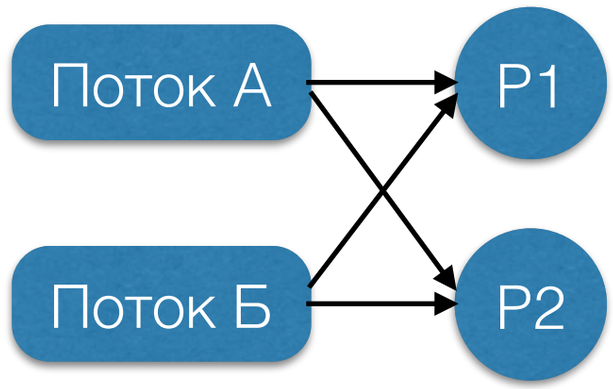
Потоки и ресурсы

- Поток может иметь локальные ресурсы, которые используются только самим потоком
- Потоки также могут иметь общие с другими потоками ресурсы
  - Это порождает много проблем
- Одна из главных проблем несгласованный доступ к ресурсам
  - Пусть, например, count это общая переменная. Предположим, count = 15
  - Два потока выполняют операцию count = count + 1;



Захват общих ресурсов

- Общие ресурсы должны управляться автоматически
  - Одновременно только один поток должен имеет доступ к ресурсу
  - Общий ресурс, может быть захвачен потоком
- Если поток имеет возможность захватить ресурс несколько раз, может возникнуть, так называемая, тупиковая ситуация
  - Поток А захватывает ресурс Р1
  - Поток Б захватывает ресурс Р2
  - Поток А пытается захватить ресурс Р2 …
  - Поток Б пытается захватить ресурс Р1 …
  - Здесь проблема в очередности захвата ресурсов



Swing и потоки

- Swing имеет отдельный поток для обработки событий
  - Поток диспетчер событий
- С компонентами Swing, после того как они становятся видимыми, можно работать только из этого потока диспетчера
- Компоненты Swing можно инициализировать из других потоков (пока компоненты невидимы)
  - Обычно это делают в основном потоке приложения

Длительные задачи

- Очень часто пользовательский интерфейс должен выполнять длительные по времени задачи
  - Например, веб браузер может загружать большой файл, одновременно отображая содержимое веб страницы
- Проблема:
  - Во время исполнения длительной операции поток диспетчер событий не может обрабатывать события!
  - Имеется только один поток диспетчер. Если он перестает работать, пользовательский интерфейс “зависает” до тех пор пока не завершится исполнение задачи

Длительные задачи (2)

- Swing имеет решение этой проблемы:
  - javax.swing.SwingWorker
- Этот класс может выполнить длительную задачу в рабочем потоке в фоновом режиме
  - Поток не влияет на работу диспетчера событий
  - Программа продолжает взаимодействовать с пользователем во время исполнения задачи
- Когда задача завершена, результаты работы SwingWorker становятся доступны в потоке диспетчере
  - Есть возможность вывести их на экран

Детали реализации SwingWorker

- SwingWorker это абстрактный класс
  - Для выполнения задачи надо сделать наследника этого класса
- И реализовать в нем несколько важных методов:
  - protected Object doInBackground()
    - В этом методе реализуется длительная задача
    - Он никогда не вызывается из потока диспетчера событий
    - Вызывается в рабочем потоке из небольшого пула потоков
- protected void done()
  - Вызывается всегда в контексте потока диспетчера событий
  - Этот метод нужен для передачи графическому пользовательскому интерфейсу Swing результатов работы задачи

Детали SwingWorker<T,V>

- SwingWorker <T, V>, это класс созданный по правилам обобщенного программирования
  - Он может (и должен) иметь параметры
- T задает тип значения, которое возвращает doInBackGround()
  - protected T doInBackground()
- Если ваша реализация doInBackground ничего не возвращает:
  - Просто укажите для T тип Object и возвращайте null

Детали SwingWorker<T,V> (2)

- V это промежуточное состояние
  - Некоторые задачи генерируют промежуточные результаты, которые должны отображаться в пользовательском интерфейсе
    - (это нужно далеко не всегда, поэтому не каждый класс использует этот функционал)
  - Если он нужен, метод doInBackGround() должен вызывать метод:
    - protected void publish(V[] chunks)
    - Когда требуется вывести промежуточные результаты
  - При этом в потоке диспетчере событий будет вызван метод:
    - protected void process(List<V> chunks)

Детали SwingWorker<T,V> (3)

- Как и раньше, если ваш SwingWorker не сообщает о промежуточных состояниях:
  - Просто укажите Object для параметра V и не используйте метод publish().

Завершение работы приложения Java

- В Java AWT закрытие главного окна приложения просто делает его невидимым
  - Если вы не позаботитесь о том чтобы закрыть приложение, оно продолжит работу
  - Для того чтобы завершить работу, при закрытии окна надо зарегистрировать реализацию WindowListener
- В Swing это можно сделать в JFrame:

```
JFrame f = new JFrame("Мое приложение!");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

  - По умолчанию используется HIDE_ON_CLOSE, так же, как в окнах AWT

Массивы в Java

- В Java массивы это объекты
  - Хотя для них разработан особенный синтаксис
- Пример:

```
int[] myInts = new int[10]; // Выделить память под массив.
for (int i = 0; i < myInts.length; i++) {
    myInts[i] = 100 * i; // Сохранить в нем данные.
}
```

- В Java все массивы размещаются в памяти динамически
- Обращение к элементам массива с помощью скобок (как в C / C++)
- Массивы имеют поле length хранящее количество элементов
- Поле length (конечно) доступно только для чтения.

Переменные массивов

- При объявлении переменных массивов скобки ставятся после типа , а не после имени переменной:
  - String[] names; а не String names[];
  - Второй способ тоже допускается, но не приветствуется
- Переменные массивы можно объявлять без инициализации:

```
boolean[] flags; // Массив значений логического типа
float[] weights; // массив чисел с плавающей точкой
```

- Но их надо проинициализировать перед использованием
  - Массив можно создать оператором new тип[размер];
    - Размер может быть равен 0! Такие массивы называются пустыми
  - Переменной массива можно присвоить другой массив
    - (Фактически массивы, это объекты в работе с которыми используется дополнительный синтаксис)
  - Массиву можно также присвоить значение null

Еще о инициализации массивов

- Значения элементов массива можно задать:

```
String[] colorNames = {
        "киноварь", "терракота", "фуксия", "шартрез", "умбра"
};
// colorNames.length == 5
```

  - Этот упрощенный синтаксис удобно использовать для инициализации массива
  - colorNames это ссылка на массив строк
    - Значения массива и его размер можно впоследствии изменить

Массивы объектов

- Массивы объектов первоначально хранят значения null
  - йИнициализация массива не инициализирует автоматически его элементы
  - Это надо делать отдельно.
- Пример:

```
//Резервируем массив для 20 ссылок
Point2d[] points = new Point2d[20];
//создаем новый объект Point2d для каждого элемента
for (int i = 0; i < points.length; i++)
    points[i] = new Point2d();
```

Многомерные массивы

- Элементом массива может быть массив. Каждый элемент nums2d имеет тип int[]

```
int[][] nums2d; // двухмерный массив целых чисел.
```

- Сначала выделяется память под массив массивов

```
nums2d = new int[20][];
```

- Затем выделяется память под каждый внутренний массив.

```
for (int i = 0; i < nums2d.length; i++)
    nums2d[i] = new int[50];
```

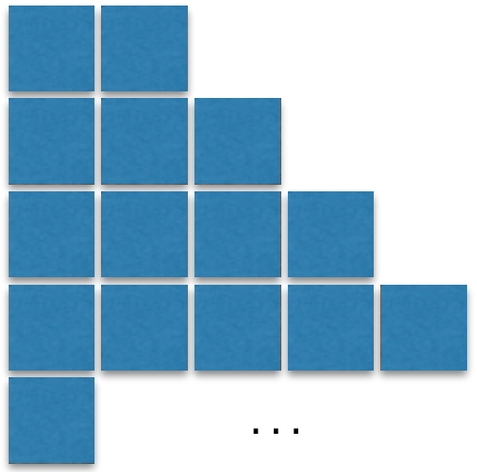
- Если размеры массивов одинаковые можно написать проще:

```
int[][] nums2d = new int[20][50]; // то же самое!
```

Еще о многомерных массивах

- Внутренние массивы могут иметь различные размеры

```
int[][] reducedMatrix;
reducedMatrix = new int[20][];
for (int i = 1; i <= 20; i++)
    reducedMatrix[i] = new int[i];
```



- Значения элементов можно указать и при инициализации многомерных массивов

```
double[][] weights = {
        {3.1,2.6}, {1.5,4.4,-3.6},
null, {6.2}
};
```

Копирование массивов

- Для того чтобы сскопировать один массив в другой используйте system.arraycopy()
- Для того чтобы сделать копию массива используйте метод clone()
  - Метод возвращает тип Object, поэтому надо при вызове сделать преобразование типов:

```
int[] nums = new int[35];
...
int[] numsCopy = (int[]) nums.clone();
```

- Это поверхностное копирование – копируются только объекты верхнего уровня
  - Если массив содержит объекты, они не клонируются
  - Если элементы массива – массивы, они тоже не клонируются.