Параллельное программирование: Лекция/Практика 2

Учебный курс

Кензин Максим Юрьевич
Кафедра Информационных технологий
ИМИТ ИГУ / Институт динамики
систем и теории управления СО РАН

ИМИТ ИГУ, 2022

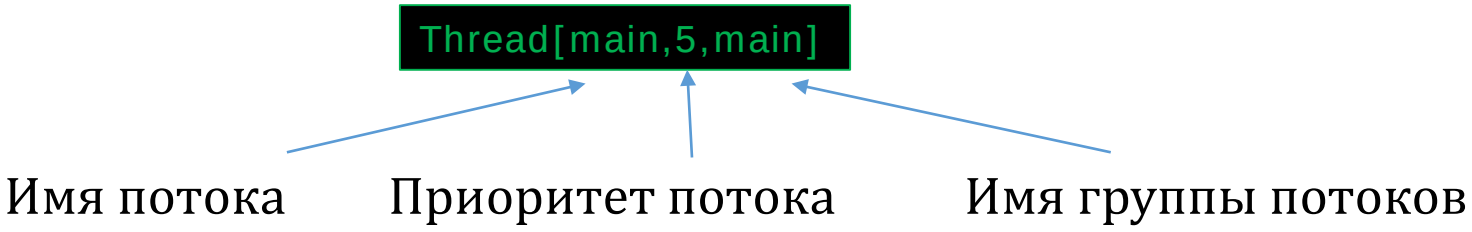
Потоки в JAVA

Класс **Thread**:

- Главный поток (создается по умолчанию);
- дочерние потоки (может создавать сами).

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

   Thread t = Thread.currentThread(); // получаем главный поток
   System.out.println(t); // main
}
```



Методы Thread

- **getName()**: возвращает имя потока
- **setName(String name)**: устанавливает имя потока
- **getPriority()**: возвращает приоритет потока
- **setPriority(int proirity)**: устанавливает приоритет потока. Приоритет является одним из ключевых факторов для выбора системой потока из кучи потоков для выполнения. В этот метод в качестве параметра передается числовое значение приоритета - от 1 до 10. По умолчанию главному потоку выставляется средний приоритет - 5.
- **isAlive()**: возвращает true, если поток активен
- **isInterrupted()**: возвращает true, если поток был прерван
- **join()**: ожидает завершение потока
- **run()**: определяет точку входа в поток
- **sleep()**: приостанавливает поток на заданное количество миллисекунд
- **start()**: запускает поток, вызывая его метод run()

Создание нового потока (-ов)

1. **Поток должен наследовать класс Thread**

```
class NewThread extends Thread {
…
}
```

Или интерфейс Runnable (единственное наследние)

```
class NewThread implements Runnable {
…
}
```

2. **Должен быть прописан метод начала работы потока**

```
public void run(){
…
}
```

 # 1 (Thread), 2 (Runnable)

Вступаем в синхронизацию

1. **Ожидаем завершения работы потока - join()**

```
t.start();
    try{
       t.join();
    }
```

 # 3 (Join [])

2. **Ловим ошибку прерывания потока.** sleep(), wait() и join() — если во время их выполнения будет вызван метод interrupt() потока, они сгенерируют исключение InterruptedException.

```
catch(InterruptedException e){
     System.out.printf("%s has been interrupted",
t.getName());
    }
```

 # 4 (Catch, Finally)

Завершение/прерывание потоков

1. **Используем булеву переменную, внутри которой работает цикл потока**

```
                Private Boolean isActive;
    MyThread()     {        isActive = true;     }
    void disable() {        isActive = false;    }
    public void run(){
           while(isActive)        { …
```

2. **Метод interrupt() (Thread.currentThread().isInterrupted()** для Runnable**)**. Сообщает потоку, что он теперь считается прерванным (Не прерывает сам!). Но зато метод **isInterrupted()** теперь возвращает *true*.

```
public void run(){
while(!isInterrupted())       { …

Thread.interrupt();
```

 # 5 (Interrupt)

Прерывание и исключения

Если срабатывает исключение, статус потока сбрасывается.

1. **Либо еще раз выставлять статус потока:**

```
catch(InterruptedException e){ interrupt(); }
```

2. **Либо весь цикл while засунуть внутрь try {}.**

```
try    {
while(!isInterrupted())      { … }
       }
catch  { … }
```