**Создание и запуск новых нитей (трэдов)**

[Java Core](https://javarush.ru/quests/QUEST_JAVA_CORE)

[Уровень 6](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=6), Лекция 2

— Привет, Амиго! Элли рассказала тебе о нитях, а я расскажу тебе о том, как с ними работать. Чтобы породить новую нить нужно:

**1)** Создать объект класса Thread (нить)

**2)** Передать в него метод, который нужно выполнить

**3)** Вызвать у созданного объекта Thread метод start.

Пример:

|  |  |
| --- | --- |
| **Код** | **Описание** |
| class Printer implements Runnable  {  public void run()  {  System.out.println("I’m printer");  }  } | Класс, который реализует интерфейс **Runnable**. |
| public static void main(String[] args)  {  Printer printer = new Printer();  Thread childThread = new Thread(printer);  childThread.start();  } | 1 Создали объект класса Printer, который содержит метод run. 2 Создали новый объект класса Thread, передали ему в конструкторе объект printer, чей метод run()нужно будет исполнить. 3 Запустили новую нить в работу, вызовом метода **start().** |

Маленькие программы на Java обычно состоят из одной нити, называемой «главной нитью» (main thread). Но программы побольше часто запускают дополнительные нити, их еще называют «дочерними нитями». Главная нить выполняет метод main и завершается. Аналогом такого метода main, для дочерних нитей служит метод run интерфейса **Runnable**.

— Ага, много нитей, много методов main.

[](https://cdn.javarush.ru/images/article/8e544491-c43f-4c11-9cf8-d33a9a153c3d/original.jpeg)

— Чтобы указать, с какого именно метода нужно начать выполнение объекту Thread, нужно как-то передать метод этому объекту. В Java это реализовано с помощью интерфейса **Runnable**. Этот интерфейс содержит единственный абстрактный метод – **void run()**. Класс Thread имеет конструктор **Thread(Runnable runnable)**, в который можно передать любой объект, который реализует интерфейс **Runnable**.

Ты должен унаследовать свой класс от интерфейса **Runnable**, затем переопределить метод run в своем классе. Именно с вызова этого метода начнется работа новой нити. В методе run ты можешь написать все, что хочешь.

|  |  |
| --- | --- |
| **Код** | **Описание** |
| class Printer implements Runnable  {  private String name;  public Printer(String name)  {  this.name = name;  }  public void run()  {  System.out.println("I’m " + this.name);  }  } | Класс, который реализует интерфейс Runnable. |
| public static void main(String[] args)  {  Printer printer1 = new Printer("Коля");  Thread thread1 = new Thread(printer1);  thread1.start();  Printer printer2 = new Printer("Вася");  Thread thread2 = new Thread(printer2);  thread2.start();  } | Создаем две нити, каждая на основе своего объекта типа Printer. |
| public static void main(String[] args)  {  Printer printer = new Printer("Наташа");  Thread thread1 = new Thread(printer);  thread1.start();  Thread thread2 = new Thread(printer);  thread2.start();  Thread thread3 = new Thread(printer);  thread3.start();  } | Создаем три нити, на основе одного объекта Printer. |

Более того, можно совместить это все в одном классе. **Класс Thread унаследован от интерфейса Runnable**, **и достаточно просто переопределить его метод run:**

|  |  |
| --- | --- |
| **Второй способ создания новой нити** |  |
| class Printer extends Thread  {  private String name;  public Printer(String name)  {  this.name = name;  }  public void run()  {  System.out.println("I’m " + this.name);  }  } | Унаследовались от класса **Thread**, который реализует интерфейс **Runnable**, и переопределили метод run. |
| public static void main(String[] args)  {  Printer printer = new Printer("Вася");  printer.start();  Printer printer2 = new Printer("Коля");  printer2.start();  } | Создаем две нити, каждая на основе своего объекта типа **Printer**. |

— Это решение красивее.

— Да, но у него есть минусы:

**1)** Вам может понадобиться запустить несколько нитей на основе одного единственного объекта, как это сделано в «примере с Наташей».

**2)** Если вы унаследовались от класса Thread, вы не можете добавить еще один класс-родитель к своему классу.

**3)** Если у вашего класса есть класс-родитель, вы не можете добавить второго – Thread.

— Т.е. каждая из нитей после вызова метода start начнет выполнять метод run того объекта, который ему передали в конструкторе?

— Да. А если в конструкторе ничего не передали, то Thread просто исполняет свой внутренний метод run.

— А почему нельзя просто вызвать этот метод, например, так:

Код

public static void main(String[] args)

{

Printer printer1 = new Printer("Коля");

printer1.run();

}

— Когда главная нить дойдет до метода run, наш «маленький роботик», просто зайдет внутрь и начнет исполнять все команды, которые там есть внутри, и только после их выполнения, вернется в метод main, и продолжит работу дальше. Создания второго «маленького робота» не произойдет, и вся работа будет делаться последовательно, а не параллельно (одновременно).

— Ясно. А можно вызвать какой-нибудь другой метод, а не run?

— Нет. Все привязано к интерфейсу Runnable, а он «знает» только об одном своем методе – run.