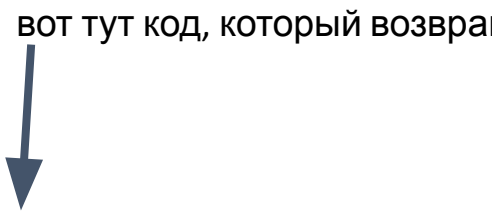
Ветвления. Методы классов

Цели: освоить ветвление. Научиться описывать поведение объектов

План урока:
- ветвления
- работа с ветвлениями
- выносим логику в методы
- параметризируем методы
- возвращаем значения
- * рекурсия

Условные конструкции в Java

Оператор «если»




```
if(условие)
{
    блок операторов
}
```

```
int a = 10;
int b = 34;

if (a > b) {
    System.out.println("a > b");
}
```

Условные конструкции в Java

Оператор «если»

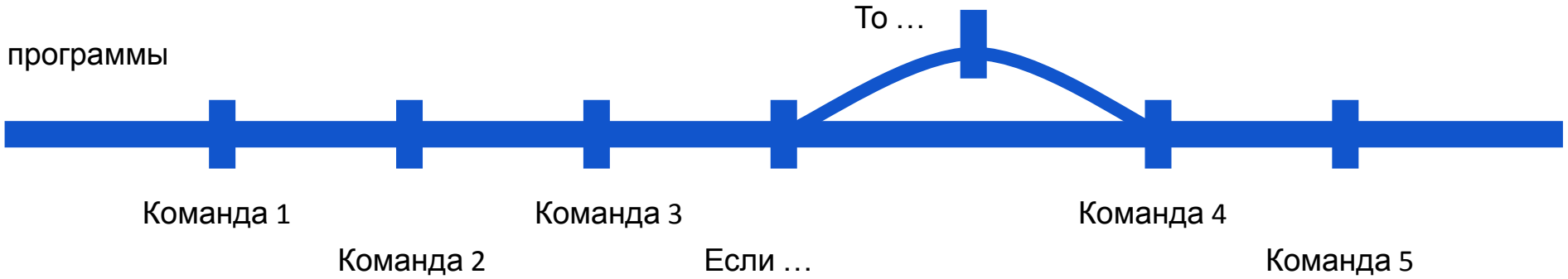
вот тут код, который возвращает true|false

```
if(условие)
{
    блок операторов
}
```

```
int a = 10;
int b = 34;

if (a > b) {
    System.out.println("a > b");
}
```

Ход выполнения программы



Условные конструкции в Java

Оператор «если-иначе»

```
if(условие)
{
    блок операторов
}
else
{
    блок операторов
}
```

```
int a = 10;
int b = 34;

if (a > b) {
    System.out.println("a > b");
}
else {
    System.out.println("a <= b");
}
```

Условные конструкции в Java

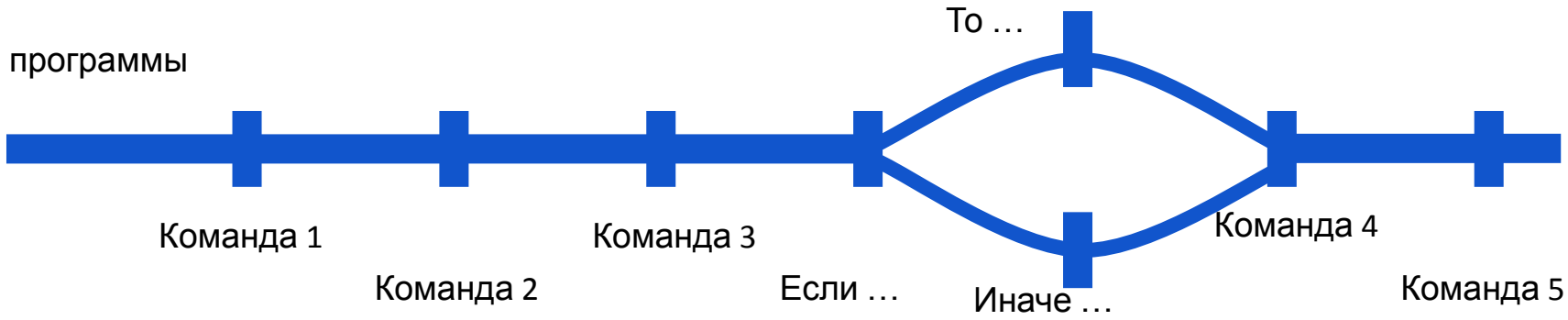
Оператор «если-иначе»

```
if (условие)
{
    блок операторов
}
else
{
    блок операторов
}
```

```
int a = 10;
int b = 34;

if (a > b) {
    System.out.println("a > b");
}
else {
    System.out.println("a <= b");
}
```

Ход выполнения программы



Условные конструкции в Java

Оператор множественного выбора

```
switch(переменная перечислимого типа)
{
    case 1-ое значение:
        блок операторов
        break;
    case 2-ое значение:
        блок операторов
        break;
    case 3-ое значение:
        блок операторов
        break;
    default:
        блок операторов - действие по умолчанию
        break;
}
```

```
Scanner sc = new Scanner(System.in)
int number = sc.nextInt();

switch(number) {
    case 1:
        System.out.println("number = 1");
    case 2:
        System.out.println("number = 2");
    default:
        System.out.println("number = ?");
}
```

Основные принципы ООП: инкапсуляция

Все внутренние данные и детали внутреннего устройства объекта должны быть скрыты от «внешнего мира». Такой подход позволяет:

- обезопасить внутренние данные (поля) объекта от изменений (возможно, разрушительных) со стороны других объектов;
- проверять данные, поступающие от других объектов, на корректность, тем самым повышая надежность программы;
- переделывать внутреннюю структуру и код объекта любым способом, не меняя его внешние характеристики (интерфейс); при этом никакой переделки других объектов не требуется.

Скрытие внутреннего устройства объектов называют инкапсуляцией

Основные принципы ООП: инкапсуляция

По умолчанию все члены класса (поля и методы) открытые, общедоступные (англ. public). Те элементы, которые нужно скрыть, в описании класса помещают в «частный» раздел (англ. private).

Чтобы упростить запись, во многие объектно-ориентированные языки программирования ввели понятие свойства (англ. property), которое внешне выглядит как переменная объекта, но на самом деле при записи и чтении свойства вызываются методы объекта.

Свойство - это способ доступа к внутреннему состоянию объекта, имитирующий обращение к его внутренней переменной.

Иногда не нужно разрешать другим объектам менять свойство, т.е. требуется сделать свойство «только для чтения». Применение свойств в этом случае позволяет регламентировать порядок доступа к полям и методам.

Вопросы?