Окончательный вариант класса Tag

В данном уроке я приведу окончательный вариант класса **Tag**, который мы и будем использовать в дальнейшем в следующих уроках.

Так как код класса достаточно большой и в нем много методов, было бы удобно сделать так, чтобы наш класс реализовывал некоторый интерфейс, в котором в компактном виде были бы прописаны все публичные методы, который должен иметь наш класс.

Добавим в интерфейс все публичные методы, описанные в уроках, а также те методы, которые были описаны в виде задач.

Итак, вот наш интерфейс:

```php
<?php
  interface iTag
  {
    // Геттер имени:
    public function getName();

    // Геттер текста:
    public function getText();

    // Геттер всех атрибутов:
    public function getAttrs();

    // Геттер одного атрибута по имени:
    public function getAttr($name);

    // Открывающий тег, текст и закрывающий тег:
    public function show();

    // Открывающий тег:
    public function open();

    // Закрывающий тег:
    public function close();

    // Установка текста:
    public function setText($text);

    // Установка атрибута:
    public function setAttr($name, $value = tru
      e);

    // Установка атрибутов:
    public function setAttrs($attrs);

    // Удаление атрибута:
    public function removeAttr($name);

    // Установка класса:
    public function addClass($className);

    // Удаление класса:
    public function removeClass($className);
  }
?>
```

А вот код нашего класса **Tag**, реализующего интерфейс **iTag**:

```php
<?php
  class Tag implements iTag
  {
    private $name;
    private $attrs = [];
    private $text = '';

    public function __construct($name)
    {
      $this->name = $name;
    }

    public function getName()
    {
      return $this->name;
    }

    public function getText()
    {
      return $this->text;
    }

    public function getAttrs()
    {
      return $this->attrs;
    }

    public function getAttr($name)
    {
      if (isset($this->attrs[$name])) {
        return $this->attrs[$name];
      } else {
        return null;
      }
    }

    public function show()
    {
      return $this->open() . $this->text . $th
        is->close();
     }

    public function open()
    {
      $name = $this->name;
      $attrsStr = $this->getAttrsStr($th
        is->attrs);

        return "<$name$attrsStr>";
      }

      public function close()
      {
        $name = $this->name;
        return "</$name>";
      }

      public function setText($text)
      {
        $this->text = $text;
        return $this;
      }

      public function setAttr($name, $value = t
        rue)
       {
        $this->attrs[$name] = $value;
        return $this;
      }
```

```php

    public function setAttrs($attrs)
    {
      foreach ($attrs as $name => $value) {
        $this->setAttr($name, $value);
      }

      return $this;
    }

    public function removeAttr($name)
    {
      unset($this->attrs[$name]);
      return $this;
    }

    public function addClass($className)
    {
      if (isset($this->attrs['class'])) {
        $classNames = explode(' ', $this->att
          rs['class']);

        if (!in_array($className, $classNames)) {
          $classNames[] = $className;
          $this->attrs['class'] = implode(' ',
            $classNames);
        }

      } else {
        $this->attrs['class'] = $className;
      }

      return $this;
    }

    public function removeClass($className)
    {
      if (isset($this->attrs['class'])) {
        $classNames = explode(' ', $this->att
          rs['class']);

        if (in_array($className, $classNames)) {
          $classNames = $this->removeElem($className, $cl
            assNames);
          $this->attrs['class'] = implode(' ',
              $classNames);
        }

      }

      return $this;
    }

    private function getAttrsStr($attrs)
    {
      if (!empty($attrs)) {
        $result = '';

        foreach ($attrs as $name => $value) {
          if ($value === true) {
            $result .= " $name";
          } else {
            $result .= " $name=\"$value\"";
          }
        }

        return $result;
      } else {
        return '';
      }
    }
```

```
131
132       private function removeElem($elem, $arr)
133       {
134         $key = array_search($elem, $arr);
135         array_splice($arr, $key, 1);
136
137         return $arr;
138       }
139     }
140   ?>
```