Министерство образования и науки Российской Федерации
Новосибирский государственный технический университет
Кафедра теоритической и прикладной информатики

Практическая работа
по дисциплине «Технологии разработки программного обеспечения»

Приложение «Конвертер систем счисления»

Факультет: ПМИ
Группа: ПМИ-41
Студент: Кислицын И. О.
Преподаватель: Зайцев М. Г.

Новосибирск
2017

1 Цель работы

Объектно-ориентированный анализ, проектирование и реализация приложения «Конвертер систем счисления» для преобразования действительных чисел представленных в системе счисления с основанием p_1 в действительные числа представленные в системе счисления с основанием p_2.

2 Ход работы

2.1 Класс Конвертер (Converter)

2.1.1 Описание

Преобразователь действительных чисел из одной системы счисления в другую.

2.1.2 Текст программы

Листинг 1: converter.js

```
/* converter.js */

/**
 * Converts given number to decimal.
 *
 * @param {string} num - Number.
 * @param {number} base - Base of num (from 2 to 16).
 * @param {number} accuracy - Number of digits after delimiter.
 * @return {number} Given number in decimal format
 */
this.toDec = function(num, base, accuracy) {
  var parts = num.split('.');
  if (parts.length > 1 && parts[1] !== '') {
    return Number.parseInt(parts[0], base)
      + Number.parseInt(parts[1].substr(0,accuracy), base)
      / Math.pow(base, parts[1].length);
  }
  return Number.parseInt(parts[0], base);
}

/**
 * Converts number from decimal to given base.
 *
 * @param {number} num - Number in decimal format.
 * @param {number} base - Base of the system to convert to.
 * @return {string} Given number in given system.
 */
this.fromDec = function(num, base) {
  return num.toString(base).toUpperCase();
}

/**
 * Converts number from one system to another.
 *
 * @param {string} num - A number to convert.
 * @param {number} from - A base of the system to convert from.
 * @param {number} to - A base of the system to convert to.
 * @param {number} accuracy - Number of digits after delimiter.
 * @return {string} Given number in from-based system.
 */
this.convert = function(num, from, to, accuracy) {
  return this.fromDec(this.toDec(num, from, accuracy), to);
}

/**
 * Converts a single digit to number.
 *
 * @param {char} dig - Single char hexadecimal digit (case-insensitive).
 * @return {number} A number denoted by given hexadecimal digit.
 */
this.digit = function(dig) {
```

```
  /* digit must be a single char */
  if (dig.length != 1) return -1;

  var code = dig.toUpperCase().charCodeAt();
  if (code >= '0'.charCodeAt() && code <= '9'.charCodeAt()) {
    return code - '0'.charCodeAt();
  }
  if (code >= 'A'.charCodeAt() && code <= 'F'.charCodeAt()) {
    return code - 'A'.charCodeAt() + 10;
  }
  return -1;
}
```

2.1.3 Тестирование

```
> // Конвертирование из заданной с.с. в десятичную
< undefined
> // toDec(число, основание с.с., точность)
< undefined
> Converter.toDec('10000',2,0)
< 16
> Converter.toDec('ffff.a8',16,10)
< 65535.65625
> Converter.toDec('ffff.1',16,10)
< 65535.0625
> Converter.toDec('ffff.8',16,10)
< 65535.5
> // Конвертирование из десятичной в заданную с. с.
< undefined
> // fromDec(число, основание с. с.)
< undefined
> Converter.fromDec(65536, 16)
< "10000"
> Converter.fromDec(2.25, 2)
< "10.01"
> Converter.fromDec(2.25, 10)
< "2.25"
```

```
> // Конвертирование из произвольной с. с. в произвольную
< undefined
> // convert(число, исходная с. с., требуемая с. с., точность)
< undefined
> Converter.convert('11111111',2,16,5)
< "FF"
> Converter.convert('11111111.00001',2,16,5)
< "FF.08"
```

2.2 Класс Редактор (Editor)

2.2.1 Описание

Редактор чисел с заданным основанием.

Хранит строковое представление числа в заданной системе счисления. Обеспечивает добавление и удаление цифр числа, добавление разделителя целой и дробной части, отчистку (установку нулевого значения), чтение числа из строки.

2.2.2 Текст программы

Листинг 2: editor.js

```
/* editor.js */

/** Current number. '' means it is zero.*/
this.number = '';
/** Base of current number. */
this.base = 10;
this.minus = false;
this.delimiter = -1;

/**
 * Get number.
 *
 * @return {string} Number.
 */
this.getNumber = function() {
  if (this.isZero()) return '0' + (this.delimiter > -1 ? '.' : '');
  if (this.delimiter != -1) {
    var ipart = this.number.substr(0, this.delimiter);
    return (ipart.length == 0 ? '0' : ipart)
      + '.' + this.number.substr(this.delimiter, 20);
  }
  return this.number;
}

/**
 * Set number.
 *
 * @param {number} num - number.
 */
this.setNumber = function(num) {
  var parts = num.split('.');
  if (parts[0][0] == '-') {
    this.minus = true;
    parts[0] = parts[0].substr(1);
  } else {
    this.minus = false;
  }
  if (parts[0] === '0') {
    parts[0] = '';
  }
  this.number = parts.join('');
```

```
  this.delimiter = parts[0].length;
}

/**
 * Check if current number is zero.
 */
this.isZero = function() {
  if (this.number.length == 0) return true;
  return false;
}

/**
 * Check if current number is negative.
 *
 * @return {bool} True, if the number is negative.
 */
this.isNegative = function() {
  if (this.isZero()) this.minus = false;
  return this.minus;
}

/**
 * Add digit to current number.
 *
 * @param {string} dig - single char hexadecimal digit.
 */
this.addDigit = function(dig) {
  // no leading zeros in my number!
  if (this.isZero() && dig == '0') return;
  var num = Converter.digit(dig);
  if (num != -1 && num < this.base)
    this.number += dig;
}

/**
 * Toggles the sign of the number.
 */
this.toggleSign = function() {
  if (this.isZero()) return;
  this.minus = this.minus ? false : true;
}

/**
 * Remove the last digit of current number.
 */
this.backspace = function() {
  if (this.delimiter == this.number.length) {
    this.delimiter = -1;
    return;
  }
  if (this.isZero()) return;
  this.number = this.number.substr(0, this.number.length - 1);
}

/**
 * Clear the current number.
 */
this.clear = function() {
  this.number = '';
  this.minus = false;
  this.delimiter = -1;
}

/**
 * Places delimiter at the end of the number.
 */
this.placeDelimiter = function() {
  if (this.delimiter != -1) return;
  this.delimiter = this.number.length;
}
```

2.2.3 Тестирование

```
> Editor.getNumber()
< "0"
> Editor.addDigit('1'); Editor.getNumber()
< "1"
> Editor.addDigit('A'); Editor.getNumber()
< "1A"
> Editor.addDigit('8'); Editor.getNumber()
< "1A8"
> Editor.placeDelimiter(); Editor.getNumber()
< "1A8."
> Editor.addDigit('0'); Editor.getNumber()
< "1A8.0"
> Editor.addDigit('6'); Editor.getNumber()
< "1A8.06"
> Editor.backspace(); Editor.getNumber()
< "1A8.0"
> Editor.clear(); Editor.getNumber()
< "0"
```

2.3 Класс История (History)

2.3.1 Описание

Документирование операций перевода чисел из одной системы счисления в другую.

2.3.2 Текст программы

Листинг 3: history.js

```
/* history.js */

/**
 * Records.
 */
this.rs = [];

/**
 * Add a record to history.
 *
 * @param {string} src_num - Source number.
 * @param {number} src_base - Source base.
 * @param {string} dst_num - Destination number.
 * @param {number} dst_base - Destination base.
 */
this.add = function(src_num, src_base, dst_num, dst_base) {
  var r = {
    src: { num: src_num, base: src_base },
    dst: { num: dst_num, base: dst_base }
  };
  this.rs.push(r);
}
```

```
/**
 * Print history in HTML format.
 */
this.print = function() {
  return '<table>' + this.rs.map( (x) => {
    return '<tr class="history"><td>' + x.src.num + '<span class="sub">'
      + x.src.base.toString()
      + '</span></td><td class="arrow"> &#8594; </td><td>'
      + x.dst.num + '<span class="sub">'
      + x.dst.base.toString() + '</span></td></tr>';
  }).join('\n') + '</table>';
}

this.printText = function() {
  return this.rs.map( (x) => {
    return '' + x.src.num + '_' + x.src.base.toString()
      + ' -> ' + x.dst.num + '_' + x.dst.base.toString();
  }).join('\n');
}

/**
 * Clear the history.
 */
this.clear = function() {
  this.rs = [];
}
```

2.3.3 Тестирование

```
> History.printText()
<- ""
> History.add('10',10,'a',12); History.printText()
<- "10_10 -> a_12"
> History.add('5',10,'101',2); History.printText()
<- "10_10 -> a_12
   5_10 -> 101_2"
> History.clear(); History.printText()
<- ""
```

2.4 Класс Управление (Runtime)

2.4.1 Описание

Единственный метод класса (`init()`) вызывается, когда графический интерфейс приложения загружен. Связывает событи графического интерфейса приложения с методами класса IFace.

2.4.2 Текст программы

Листинг 4: runtime.js

```
/* runtime.js */

const ipc = require('electron').ipcRenderer;

/**
 * The first function to call when the page is ready.
 */
function init()
{
  IFace.disableNumkeys(Editor.base);

  var range = document.getElementById('base-range')
  range.onchange = function(e) {
```

```
    var base = e.srcElement.value;
    IFace.convert(base);
    IFace.setBase(base);
    IFace.disableNumkeys(base);
    IFace.updateNumber();

    document.getElementById('history').innerHTML = History.print();
  }

  ipc.on('set-layout', (e, arg) => {
    var ls = document.getElementsByClassName('layout');
    for (var i = 0; i < ls.length; ++i) {
      if (ls[i].id === arg) {
        if (!ls[i].classList.contains('hidden')) continue;
        ls[i].classList.remove('hidden');
        continue;
      }
      if (ls[i].classList.contains('hidden')) continue;
      ls[i].classList.add('hidden');
    }
  });

  var keys = document.getElementsByClassName('numkey')
  for (var i = 0; i < keys.length; ++i) {
    keys[i].onclick = hitKey;
  }

  document.querySelector('body').onkeypress = hitKey;
  document.querySelector('body').onkeydown = hitKey;
}

/**
 * Event to call when you hit the key.
 *
 * @param {Event} e - event handled by function.
 */
function hitKey(e) {
  var cmd = '';

  if (e.type == 'click') {
    cmd = e.srcElement.innerHTML;
  } else if (e.type == 'keypress') {
    cmd = String.fromCharCode(e.charCode || e.keyCode);
  } else if (e.type == 'keydown') {
    switch (e.keyCode) {
    case 8: // BS
      cmd = 'BS';
    break;
    case 46: // del
      cmd = 'CE';
    break;
    }
  }

  if (cmd !== '') {
    IFace.keyCmd(cmd);
    IFace.updateNumber();
  }
}
```

2.5 Класс Интерфейс (IFace)

2.5.1 Описание

Данный класс связывает пользовательский интерфейс (HTML-layer) и остальные классы. Его методы обеспечивают управление визуальными элементами (например, включение и отключение ненужных кнопок для ввода цифр при смене основания системы счисления).

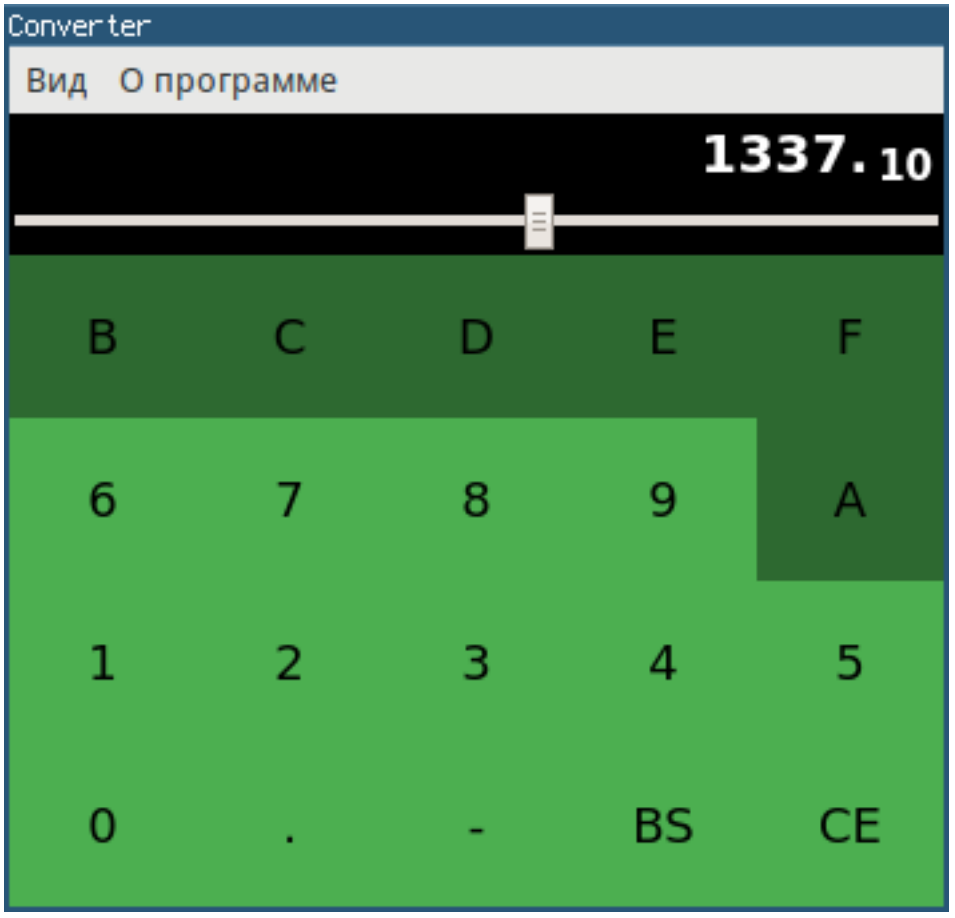
2.5.2 Текст программы

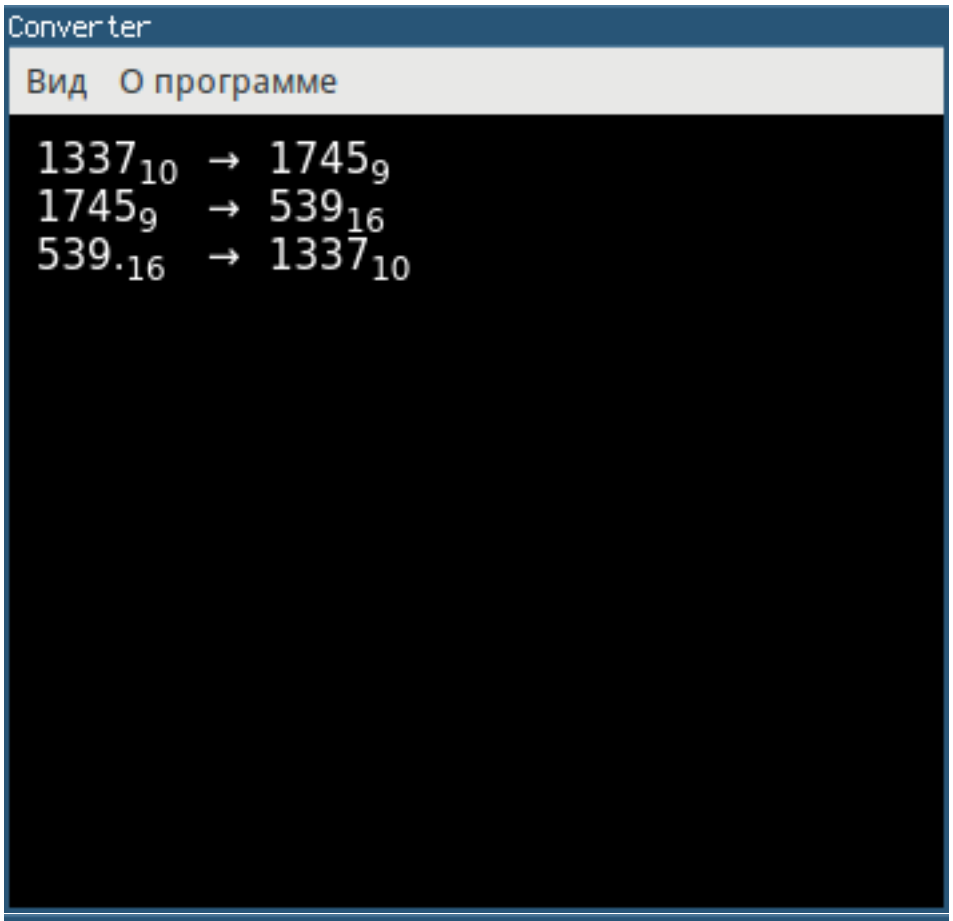
Листинг 5: iface.js

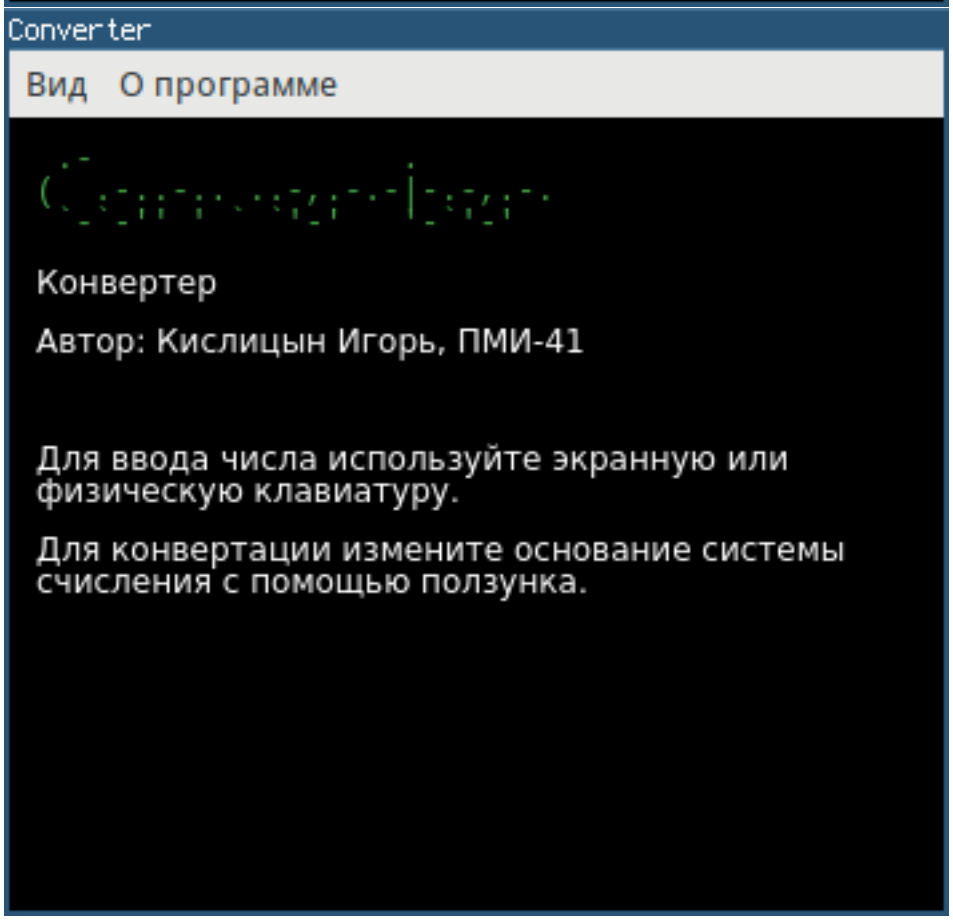
```
/* iface.js */

/**
 * Enable and disable number keys depending on given base. Keys with number <=
 * base will be disabled.
 *
 * @param {number} base - Base of the current number.
 */
this.disableNumkeys = function(base) {
  var keys = document.getElementsByClassName('numkey');
  for (var i = 0; i < keys.length; ++i) {
    var num = Converter.digit(keys[i].innerHTML);
    if (num < base) {
      keys[i].classList.remove('disabled');
    } else {
      keys[i].classList.add('disabled');
    }
  }
}

/**
 * Handle command that comes from web-interface.
 *
 * @param {string} cmd - Command to be executed. Can be hexadecimal digit, BS,
 *        CE, '-' or '.'.
 */
this.keyCmd = function(cmd) {
  // digits
  if (Converter.digit(cmd) >= 0) {
    Editor.addDigit(cmd);
  } else if (cmd === 'BS') {
    Editor.backspace();
  } else if (cmd === 'CE') {
    Editor.clear();
  } else if (cmd === '-') {
    Editor.toggleSign();
  } else if (cmd === '.') {
    Editor.placeDelimiter();
  }
}

/**
 * Get current number from Editor. Use this method instead of getting
 * Editor.number directly.
 *
 * @return {string} - Current number string.
 */
this.getNumber = function() {
  return Editor.getNumber();
}

/**
 * Get current base string from editor. Use this method instead of getting
 * Editor.base directly.
 *
 * @return {string} - Current base string.
 */
this.getBase = function() {
  return Editor.base.toString();
}

/**
 * Set Editor's base.
 *
 * @param {number} base - New value of Editor.base.
 */
this.setBase = function(base) {
  Editor.base = base;
}

/**
```

```
 * Updates number view on the screen (both number and base) according to
 * Editor's properties.
 */
this.updateNumber = function() {
  var num = document.getElementById('number'),
    base = document.getElementById('base'),
    minus = document.getElementById('sign');
  num.innerHTML = this.getNumber();
  base.innerHTML = this.getBase();
  sign.innerHTML = Editor.isNegative() ? '-' : '';
}

/**
 * Start conversion.
 *
 * @param {number} base - Base to convert to.
 */
this.convert = function(base) {
  var num = this.getNumber(),
    from = this.getBase(),
    to = base,
    sign = Editor.isNegative() ? '-' : '',
    // feel free to change accuracy (4th arg of convert)
    res = sign + Converter.convert(num, from, to, 20);
  Editor.setNumber(res);
  History.add(num, from, res, to);
  //console.log(Editor.nuber);
}
```

2.5.3 Тестирование




Converter
Вид О программе
$1337_{10} \rightarrow 1745_{9}$
$1745_{9} \rightarrow 539_{16}$
$539._{16} \rightarrow 1337_{10}$


Converter
Вид О программе
Конвертер
Автор: Кислицын Игорь, ПМИ-41
Для ввода числа используйте экранную или физическую клавиатуру.
Для конвертации измените основание системы счисления с помощью ползунка.

2.6 Класс About

2.6.1 Описание

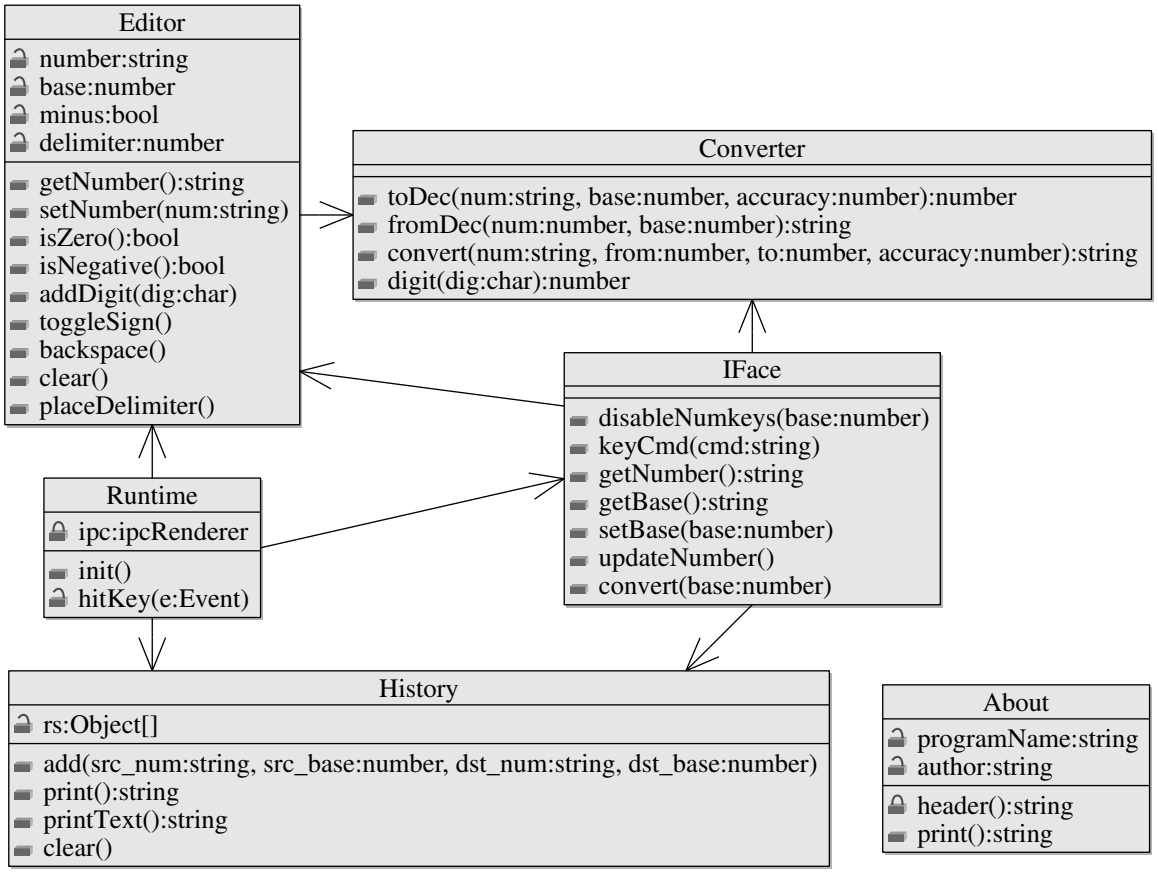
Хранит справочную информацию о приложении.

2.6.2 Текст программы

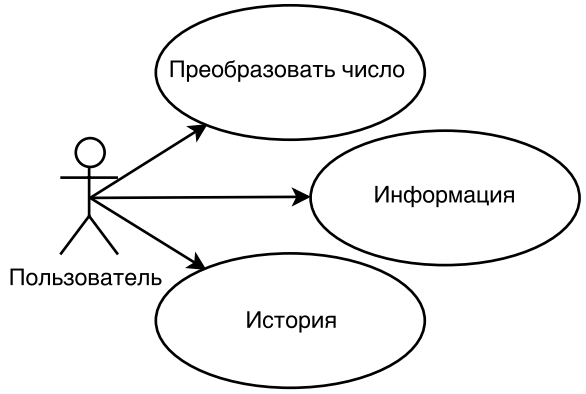
Листинг 6: about.js

```
/* about.js */

this.header = function() {
  return [" .-                    .       ",
      "(  .-..-.. ..-,.-.-|-.-,.-.",
      " '-'-'' ' ' '','-'   '-''-'  "]
    //.join('<br />');
    .join('\n');
}

this.programName = /**/'Конвертер'/**/;
this.author = /**/'Кислицын Игорь, ПМИ-41'/**/;

this.print = function() {
  return ['<pre id="about-header">' + this.header() + '</pre>',
      '<p>' + this.programName + '</p>',
      /**/'<p>Автор: '/**/ + this.author + '</p><br />',
      /**/'<p>Для ввода числа используйте экранную или физическую клавиатуру.</p>',/**/
      /**/'<p>Для конвертации измените основание системы счисления с помощью ползунка.</p>'/**/].join('');
}
```

2.7 Диаграмма классов



2.8 Usecase-диаграмма



2.9 Диаграмма последовательности