IT ШКОЛА SAMSUNG

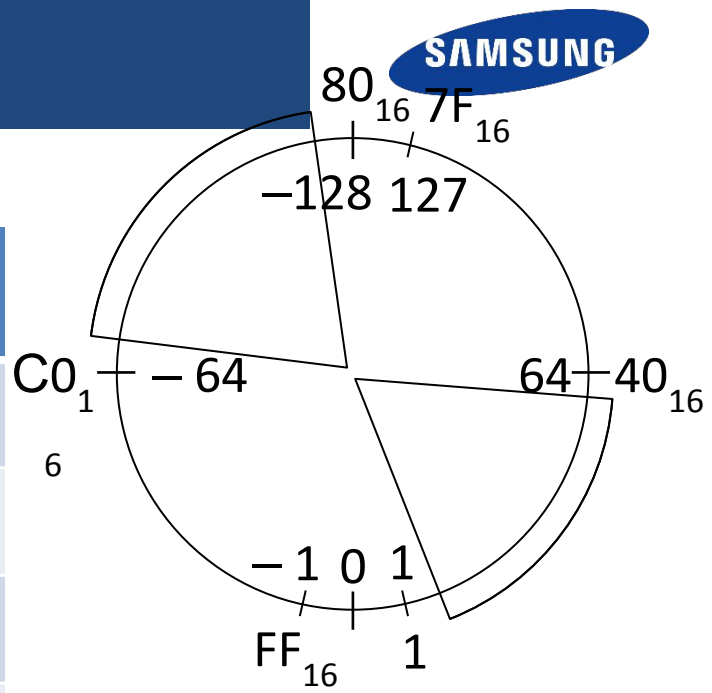
Модуль 1. Основы программирования

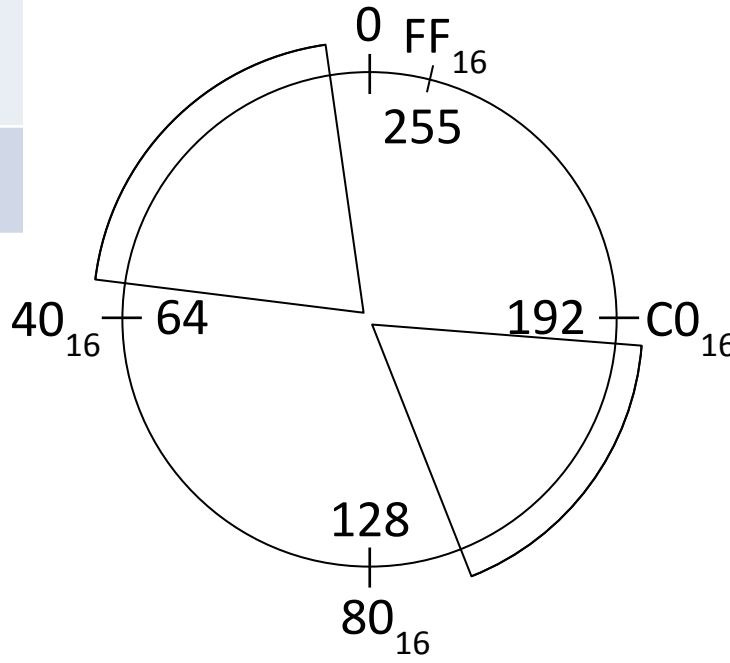
Урок 5-6. Представление данных в памяти

SAMSUNG

JAVA. ЦЕЛОЧИСЛЕННЫЕ ТИПЫ

Название	Длина (байт/бит)	Область значений
Byte	1 (8)	-128..127
Short	2 (16)	-32 768 .. 32 767
Int	4 (32)	-2 147 483 648 .. 2 147 483 647
long	8 (64)	-9 223 372 036 854 775 808 .. 9 223 372 036 854 775 807 (примерно 10^{19})
char	2 (16)	'\u0000' .. '\uffff', или 0 .. 65 535






```
    1111 1111
+   0000 0001
-------------
  1 0000 0000
```

Все целочисленные типы Java хранят числа в дополнительном коде!

JAVA. ПРЕДСТАВЛЕНИЕ ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ

Старший (знаковый) бит числа определяет его знак. Если он равен 0, число положительное, если 1, то отрицательное.

Прямой код:

≥ 0

78 = 1001110_2	0	1	0	0	1	1	1	0
– 78 = $–1001110_2$	1	1	0	0	1	1	1	0

< 0

операции с положительными и отрицательными числами выполняются по-разному!

JAVA. ЦЕЛЫЕ ЧИСЛА СО ЗНАКОМ

Идея: «– 1» должно быть представлено так, чтобы при сложении с числом «1» получить 0.

Как кодируется «-1»?

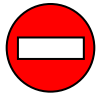
```
   1111 1111      -1 → 255
+  0000 0001        1
-------------------------
 1 0000 0000      256
```

Для 8-битных чисел: код числа «–X» равен двоичному коду числа 256 – X (*дополнение* до 256).

При *K*-битном кодировании *дополнительный* код числа «–X» равен двоичному коду числа 2^K – X (*дополнение* до 2^K).

JAVA. КАК ПОСТРОИТЬ ДОПОЛНИТЕЛЬНЫЙ КОД

Алгоритм A0: перевести число $2^K - X$ в двоичную систему счисления.



для вычислений требуется *K+1* разряд

Алгоритм A1:

1) перевести число *X* в двоичную систему счисления;
2) построить *обратный код*, выполнив *инверсию* всех битов (заменить 0 на 1 и наоборот);
3) к результату добавить 1.

```
 78 = 01001110₂
      10110001   ← инверсия
-78 → 10110010   +1
```

JAVA. КАК ПОСТРОИТЬ ДОПОЛНИТЕЛЬНЫЙ КОД

Алгоритм А2:

1) перевести число *X-1* в двоичную систему счисления;
2) выполнить инверсию всех битов.

$$78 - 1 = 77 = 01001101_2$$
$$-78 \rightarrow 10110010 \leftarrow \text{инверсия}$$

Алгоритм А3:

1) перевести число *X* в двоичную систему счисления;
2) выполнить инверсию всех старших битов числа, кроме младшей единицы и нулей после нее.

$$78 = 01001110_2$$
$$-78 \rightarrow 10110010 \leftarrow \text{инверсия}$$

JAVA. ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ

С фиксированной запятой (в первых ЭВМ):

целая часть	дробная часть

0,0000000000000012345 12345000000000000,0

для больших и маленьких чисел нужно масштабирование

С плавающей запятой (автоматическое масштабирование):

$A = \pm Z \cdot B^{P}$ $1{,}2345 \cdot 10^{-14}$ $1{,}2345 \cdot 10^{17}$

знак	порядок P	значащая часть Z
	положение запятой	цифры числа

JAVA. ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ

Теоретически оптимальный вариант (целая часть = 0):

$0{,}0012345 = 0{,}12345 \cdot 10^{-2}$

$12{,}345 = 0{,}12345 \cdot 10^{2}$

всегда 0

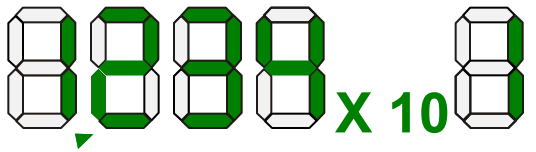
основание системы счисления

один разряд расходуется впустую!

Экономный вариант (целая часть от 1 до B):

$0{,}0012345 = 1{,}2345 \cdot 10^{-3}$

$12{,}345 = 1{,}2345 \cdot 10^{1}$



повышение точности при конечном числе разрядов

Нормализованная форма: значащая часть Z удовлетворяет условию $1 \le Z < B$, где B – основание системы счисления (стандарт *IEEE 754*).

Пример:

$17{,}25 = 10001{,}01_2 = 1{,}000101_2 \cdot 2^4$

$5{,}375 =$

$7{,}625 =$

$27{,}875 =$

$13{,}5 =$

$0{,}125 =$

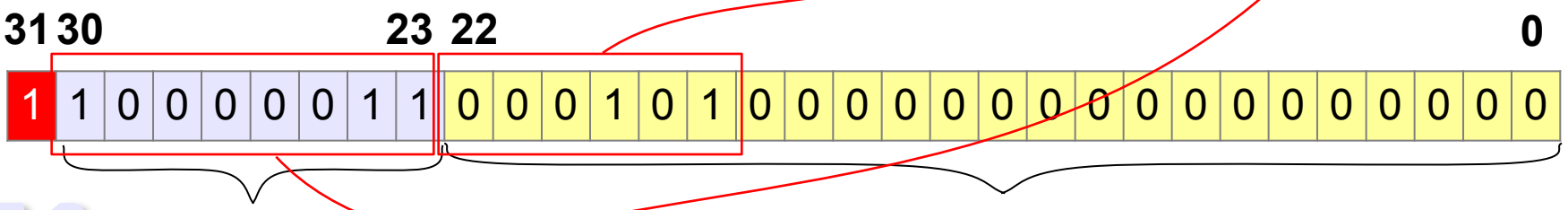
всегда 1, её можно не хранить в памяти!

JAVA. ЧИСЛО ОБЫЧНОЙ ТОЧНОСТИ (SINGLE)

знак	порядок	значащая часть

$-17{,}25 = -10001{,}01_2 = -1{,}000101_2 \cdot 2^4$

single: 4 байта = 32 бита



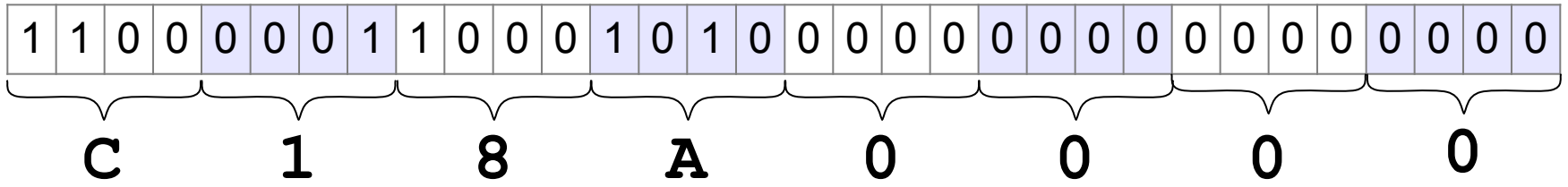
знак

порядок со смещением

мантисса = дробная часть Z

$p = 4 + 127 = 131 = 10000011_2$

для *single*



порядок со смещением был придуман, чтобы избавиться от хранения знака порядка

Нормализованная форма: значащая часть Z удовлетворяет условию $1 \le Z < B$, где B – основание системы счисления (стандарт *IEEE 754*).

Пример:

$17{,}25 = 10001{,}01_2 = 1{,}000101_2 \cdot 2^4$

$5{,}375 =$

$7{,}625 =$

$27{,}875 =$

$13{,}5 =$

$0{,}125 =$

всегда 1, её можно не хранить в памяти!

ПРИМЕР

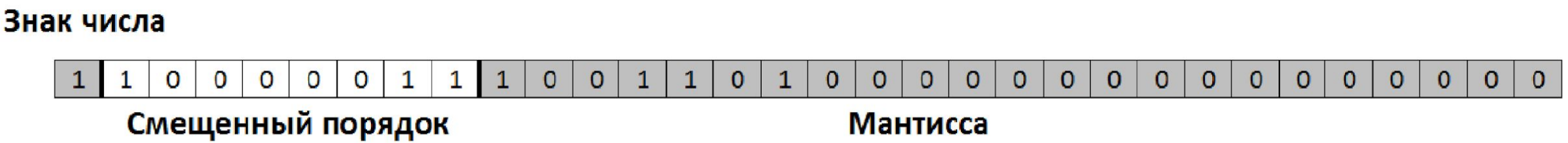
Представим число -25,625 в машинном виде с использованием 4 байтового представления (где 1 бит отводится под знак числа, 8 бит - под смещённый порядок, остальные биты - под мантиссу).

1. $25_{10}=100011_2$ $0,625_{10}=0,101_2$ $-25,625_{10}= -100011,101_2$

2. $-100011,101_2= -1,00011101_2 * 2^4$

3. СП = 127 + 4 = 131 = 1000011

4. Результат:



JAVA. ДИАПАЗОН ВЕЩЕСТВЕННЫХ ЧИСЕЛ

тип	диапазон	число десятичных значащих цифр	размер (байт)
single	$1{,}4 \cdot 10^{-45} – 3{,}4 \cdot 10^{38}$	7-8	4
double	$4{,}9 \cdot 10^{-324} – 1{,}8 \cdot 10^{308}$	15-16	8
extended	$3{,}6 \cdot 10^{-4951} – 1{,}2 \cdot 10^{4932}$	19-20	10

Extended – тип для вычислений в сопроцессоре, единица в значащей части не скрывается.

Single, double – только для хранения.

JAVA. ПРИВЕДЕНИЕ ПРИМИТИВНЫХ ТИПОВ

Если в выражении участвуют операнды разных типов, происходит их приведение.

Приведение в большую сторону происходит автоматически, в меньшую нужно делать вручную!

```
double z = 5;
```

Целое 5 при присваивании будет преобразовано в **double**

```
int x = 3 / 1.5; //НЕВЕРНО!
```

не скомпилируется, нужно приводить типы явно:

```
int x = (int)(3 / 1.5);
```

Правильно возвести 100 миллионов в квадрат можно так:

```
int x = 100 * 1000 * 1000;
out.println((long)x * x);
```

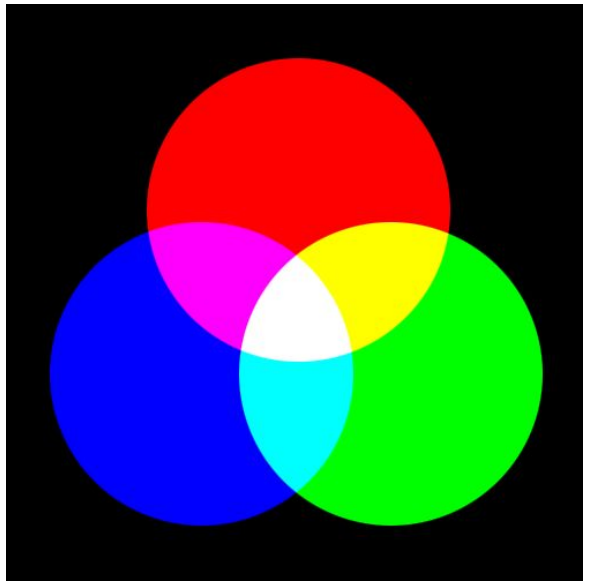
Достаточно при этом привести к типу ***long*** один из множителей.

ПРАКТИКА – СИСТЕМЫ СЧИСЛЕНИЯ – графическая информация

Д. Максвелл, 1860

цвет = (R, G, B)

red	***green***	***blue***
красный	зеленый	синий
0..255	0..255	0..255



(0, 0, 0)	(0, 255, 0)
(255, 255, 255)	(255, 255, 0)
(255, 0, 0)	(0, 0, 255)
(255, 150, 150)	(100, 0, 0)

? Сколько разных цветов можно кодировать?

256·256·256 = **16 777 216** (*True Color,* «истинный цвет»)

! RGB – цветовая модель для устройств, излучающих свет (мониторов)!

ПРАКТИКА – СИСТЕМЫ СЧИСЛЕНИЯ – графическая информация

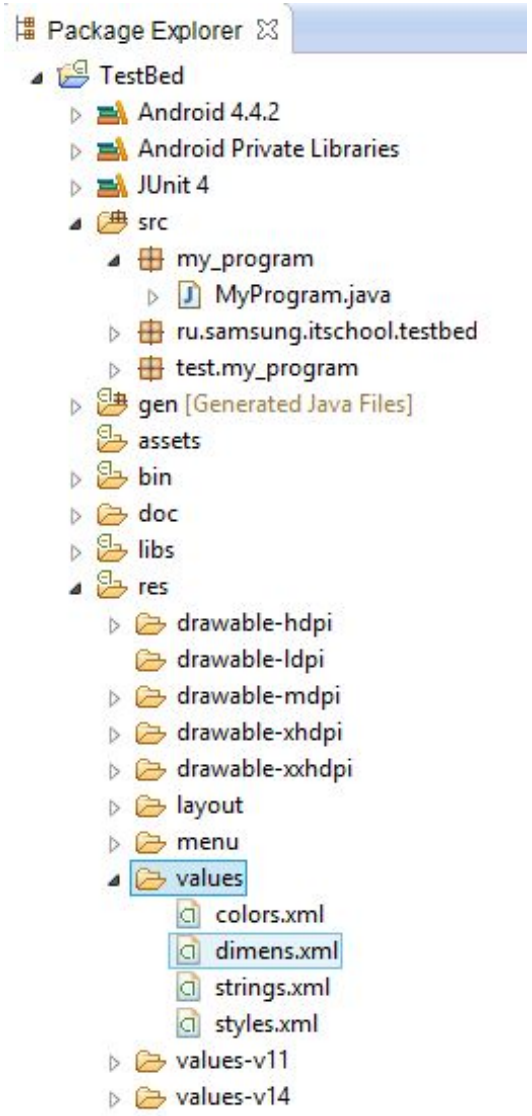
Программисты чаще пользуются шестнадцатеричной системой счисления!

(255, 255, 0) → #FFFF00

	RGB	Веб-страница
	(0, 0, 0)	#000000
	(255,255,255)	#FFFFFF
	(255, 0, 0)	#FF0000
	(0, 255, 0)	#00FF00
	(0, 0, 255)	#0000FF
	(255, 255, 0)	#FFFF00
	(204,204,204)	#CCCCCC

ПРАКТИКА. ОФОРМЛЕНИЕ АНДРОИД ПРОГРАММ

Оформление программ хранится в разделе ресурсов и помещается в XML-файл



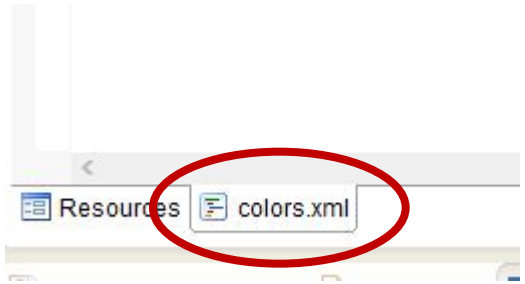
Идея проста: не указывать в программе конкретный цвет или размер, а описать его в файле ресурсов, присвоив ему идентификатор, а дальше использовать именно этот идентификатор (ID). Это дает возможность изменять внешний вид программы без изменения программного кода. Обычно для цветовых ресурсов используют файл **colors.xml** в подкаталоге **/res/values.**

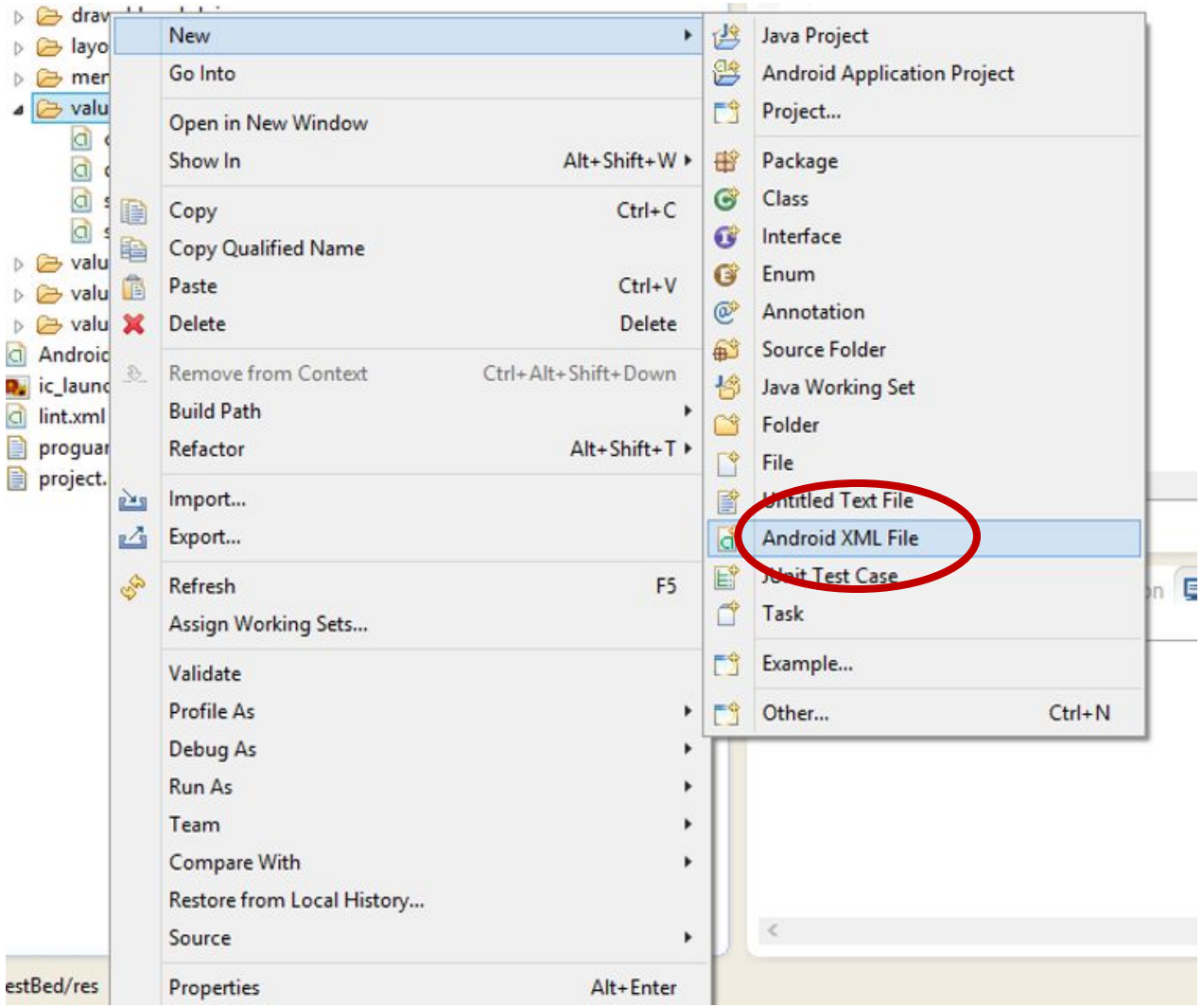
ПРАКТИКА. ОФОРМЛЕНИЕ АНДРОИД ПРОГРАММ

Обычно для цветовых ресурсов используют файл colors.xml в подкаталоге /res/values. Но можно использовать любое произвольное имя файла, или даже вставить их в файл вместе со строковыми ресурсами strings.xml.

Задание 1

Создайте файл **colors.xml** в подкаталоге **/res/values**
и переключитесь к текстовому виду





ПРАКТИКА. ОФОРМЛЕНИЕ АНДРОИД ПРОГРАММ

Для работы с цветом используется тег **<color>**, а цвет указывается в специальных значениях.
#RGB;
#RRGGBB;
#ARGB;
#AARRGGBB;
A – это альфа-канал величина обратная прозрачности. То есть цвет #4000FF00 это почти прозрачный зеленый.
Определенные таким образом цвета можно использовать в других частях кода и xml-файлах

Задание 2
Напишите в файле

```
<?xml version="1.0" encoding="utf8"?>
<resources>
<color name="red">#f00</color>
<color name="yellow">#FFFF00</color>
<color name="transpgreen">#4000FF00</color>
</resources>
```

ПРАКТИКА. ФАЙЛ РАЗМЕТКИ ОСНОВНОЙ АКТИВНОСТИ

Откройте файл разметки основной активности **res ⇒ layout ⇒ activity_main.xml** (там описываются элементы главного окна приложения), переключитесь к текстовому виду (вкладка внизу activity_main.xml)
LinearLayout – все окно, **EditText** – поле ввода, **TextView** – основное поле вывода и
Button – кнопка закрытия, которая возникает после окончания работы запущенной консольной программы.

Задание 3 задайте цвет основному полю ввода consoleWrite

```
<TextView
android:id="@+id/consoleWrite"
android:layout_width="match_parent"
android:gravity="top"
android:layout_height="0dp"
android:layout_weight="1"
android:background="@color/yellow" />
```

Цвет фона

Ссылка на ресурс в собственном пакете

Запустите приложение на планшете

ПРАКТИКА. ФАЙЛ РАЗМЕТКИ ОСНОВНОЙ АКТИВНОСТИ

Можете попробовать добавить различным элементам разметки кроме **android:background** свойства
`android:textColor` и
`android:TextColorHint` (это свойство применяется, когда в поле ввода еще нет текста пользователя и там отображается подсказка).

Разные элементы поддерживают разные свойства, надо уточнять по документации.

Также существуют предопределенные названия цветов. Такие ID доступны в пространстве имен **`android.R.color.`**
Посмотреть цветовые значения цветов можно в документации
`http://developer.android.com/reference/android/R.color.html`

Для ссылки на системный ресурс мы должны, например, записать:
`android:textColor="@android:color/black"`

САМОСТОЯТЕЛЬНАЯ РАБОТА

1) Задайте фону приложения фиолетовый цвет
2) Задайте желтый цвет текста
3) В приложении TestBed вставьте фрагмент кода, посмотрите на результат

```
Float f = new Float("124.32432");
 int intBits = Float.floatToIntBits(f);
 String binary =
Integer.toBinaryString(intBits);
 out.println("Binary = " + binary);
```

4) Используя следующие закономерности:

```
int sign = intBits & 0x80000000;
int exponent = intBits & 0x7f800000;
int mantissa = intBits & 0x007fffff;
```

Посмотрите как представлена порядок и мантисса в памяти компьютера для любого числа

5) Исправьте программу так, чтобы она отображала тип Double в двоичном виде

Домашнее задание

Повторить занятия 1,2,3 по материалам уроков в дистанционной системе обучения

IT ШКОЛА SAMSUNG / Модуль 1. Основы программирования (Java) /
/Учебные материалы 1.1-1.3

Спасибо!

В презентации использованы материалы К. Полякова
http://kpolyakov.spb.ru/