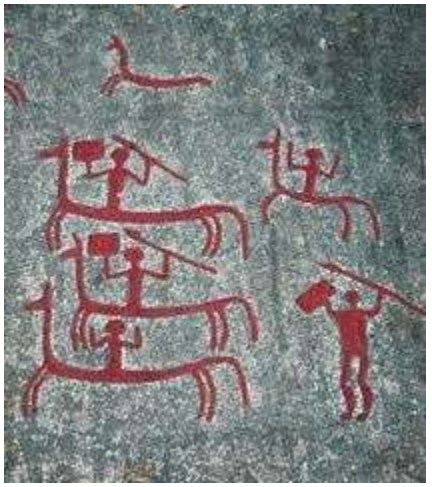
Java
2
tipos, operadores, String

Vitor Vaz da Silva

Tipos de Dados Primitivos

- Inteiro
  - byte
  - short
  - int
  - long
- Vírgula flutuante
  - float
  - double
- Caractere
  - char
- Lógico
  - boolean



Inteiros

Tipo	Bits	Mínimo	Máximo
byte	8	-128	+127
short	16	-32 768	+32 767
int	32	-2 147 483 648	+2 147 483 647
long	64	-9 223 372 036 854 775 808	+9 223 372 036 854 775 807

Vírgula Flutuante

Tipo	Bits	Mínimo	Máximo
float	32	1.4×10^{-45}	$3.4028235 \times 10^{+38}$
double	64	4.9×10^{-324}	$1.7976931348623157 \times 10^{+308}$

A mesma gama de valores para os números negativos

Caractere

Tipo	Bits	Mínimo	Máximo
char	16	0	65 535

‘A’
\x41
\0101
\u0041

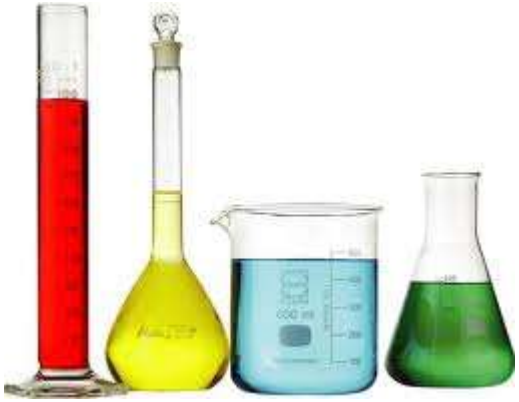
65 em decimal

\n	Mudança de Linha
\r	Voltar ao início da linha
\b	Ir uma posição para a esquerda
\t	Um tab
\f	Mudança de página
\’	Um apóstrofo
\”	Uma aspa
\\	A própria barra invertida
\uhhhh	Caractere unicode (0-9, A-F)
\0ooo	Caractere octal (0-7)
\xhh	Caractere hexadecimal (0-9, A-F)

Lógico

Tipo	Bits	Mínimo	Máximo
boolean	8	false	true

Variáveis



- Podem começar com _ ou $
- Não podem ser palavras reservadas

abstract assert boolean break byte case catch char class const continue default do double else enum extends false final finally float for goto if implements import instanceof int interface long native new null package private protected public return short static strictfp super switch synchronized this throw throws transient true try void volatile while

Atribuição

- Atribuir valores às variáveis

```
double precoBatatas=1.99;
boolean portaAberta=true;
int quantidade=4, idade=20;
int cadeiras=quantidade;
float valorReal=12.234F;
long inteiro=123L;
```

Casting

- Permite converter um tipo noutro sem alterar o seu valor binário.
- Pode gerar erros
- Pode ser necessário:
  - Acrescentar bits
  - Remover bits

Diferenças

null	0
int i;	Integer j;
int i=null;	Integer j=null;
int i=(Integer) null;	Integer j=(Integer) null;
int i=87;	Integer j=87;
int i=j;	Integer j= new Integer(87);

System.out.println(“i “ + i + “ j ” + j);

Operadores

- Matemáticos
- Incremento e Decremento
- Relacionais
- Lógicos

Operadores Matemáticos

\+ adição

\- subtracção

* multiplicação

/ divisão

% resto da divisão inteira

7 % 3 tem como resultado 1

Operadores Incremento e Decremento

++

--

+=

-=

*=

/=

%=

```
++a;          // a=a+1;
a++;          // a=a+1
```

Operação e Atribuição

```
b=++a;        //a=a+1; b=a;
b=a++;        //b=a; a=a+1;

a/=3;         // a=a/3;

a%=5;         //a=a%5;
```

Operadores Relacionais

==

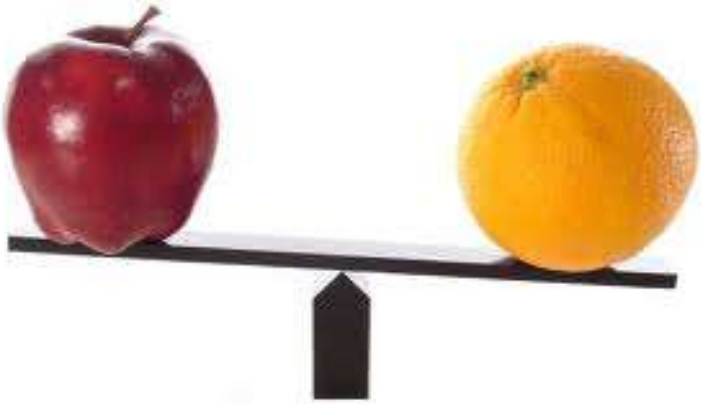
!=

>

<

>=

<=



Operadores Lógicos

true

false

&&

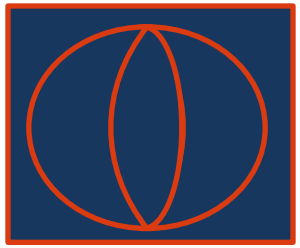
||

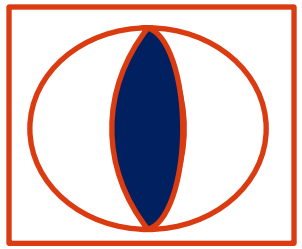
!

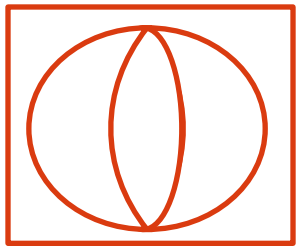
^

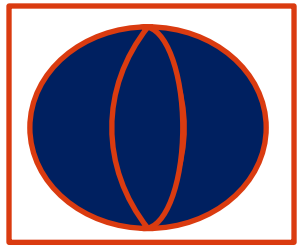
&

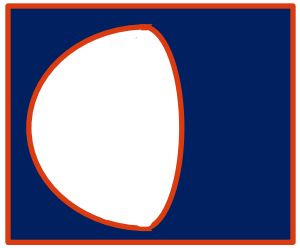
|

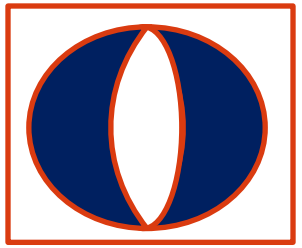






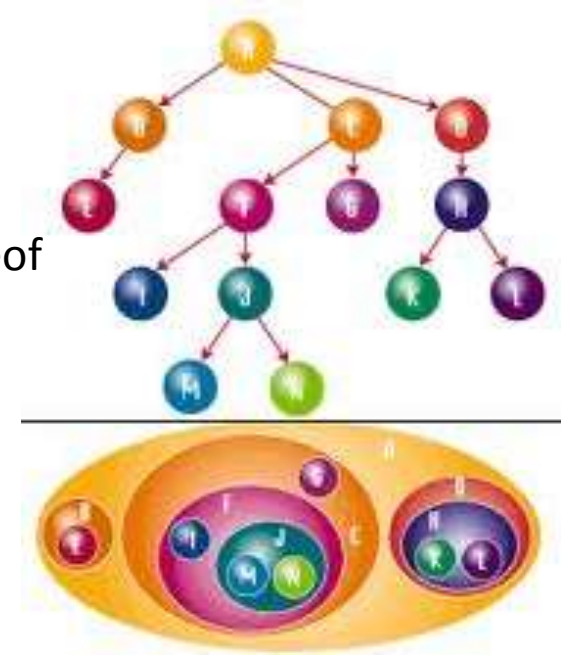




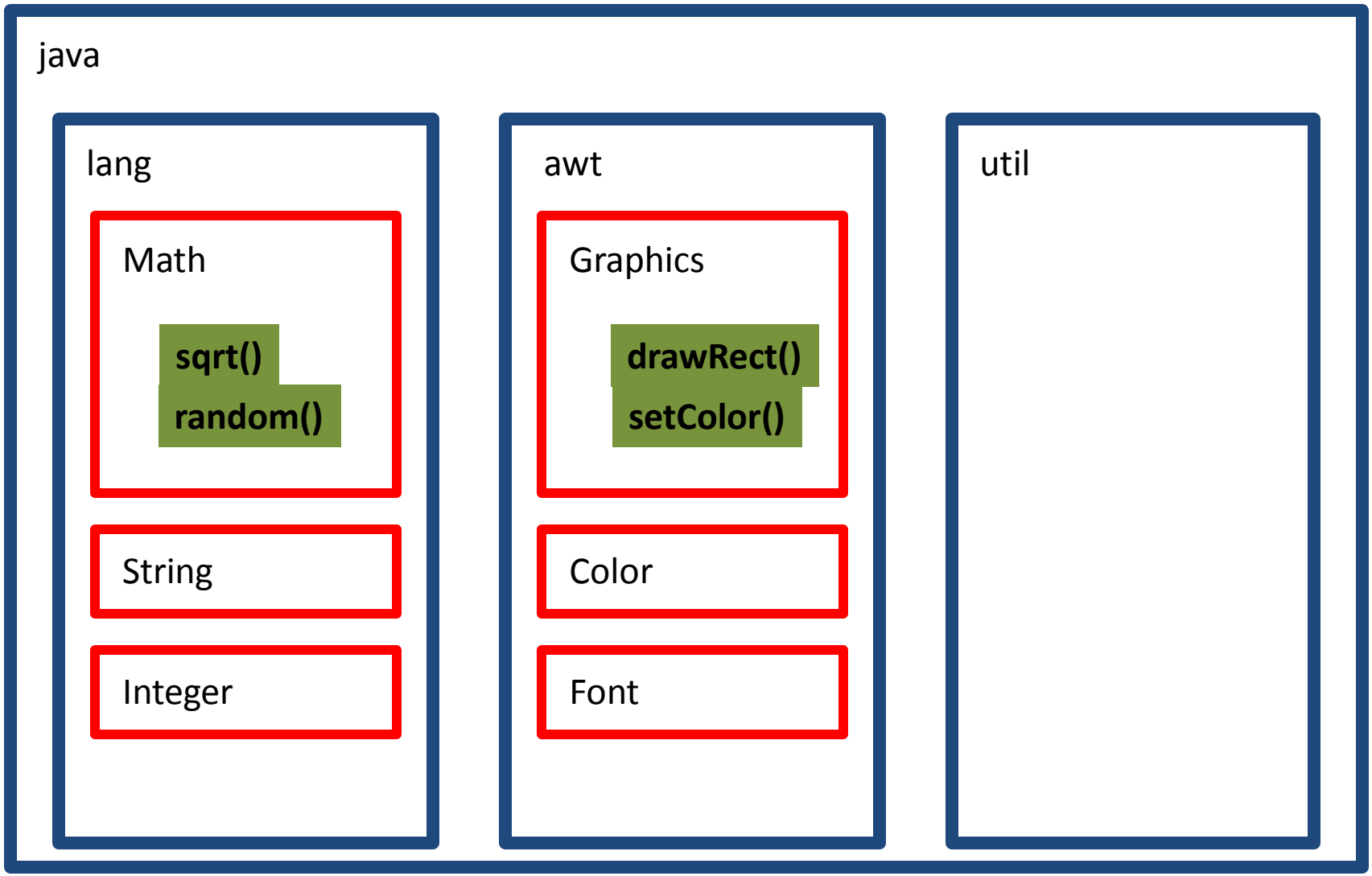


Operadores

Operadores	Precedência
array index, method call, member access	[] () .
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=



Pacotes – Classes - Métodos



Pacotes - Classes - Objectos - Métodos

- Convenção

pacote.Classe.objecto.metodo();

java.lang.System.out.println()

java.awt.Rectangle

String

- Manipula caracteres que não se pretende serem alterados

```
String s = “Olá”;
    // String s = new String(“Olá”);

s = s + “Bom “ + “Dia”;
    // s = new String (s + new String(“Bom ”) + new String(“Dia”));
```

String

Concatenção

O operador + concatena duas strings numa.

Se apenas um dos operandos for uma String, o outro é convertido em string (útil para escrever números)

Se um objecto (não String) for concatenado com uma String, o método toString() desse objecto é evocado();

String

Métodos

- Assumir nestes exemplos:

```
int i,j; //índices dentro da string
String s,t;
Char c;
CharacterSequence cs; // String ou StringBuilder
Boolean b;
x qualquer tipo ou valor (primitivo ou object).
```

Dimensão

```
i = s.length();     //número de caracteres na string
```

String

Comparar (usar em vez de == ou !=)

i = *s*.**compareTo**(*t*); // <0 (s<t) - 0 (s==t) - >0 (s>t)

i = *s*.**compareToIgnoreCase**(*t*);

b = *s*.**equals**(*t*); // true (s==t)

b = *s*.**equalsIgnoreCase**(*t*)

b = *s*.**startsWith**(*t*); // true *s* = t + qualquercoisa;

b = *s*.**startsWith**(*t*, *i*); // true s = inicio + *t(i)* + fim;

b = *s*.**endsWith**(*t*); true s = qulquercoisa + t;

String

Pesquisa – (devolve -1 se não encontrar)

i = *s*.**contains**(*cs*); // true cs está contido em s

i = *s*.**indexOf**(*t*); // índice da primeira ocorrência de t em s

i = *s*.**indexOf**(*t*, i); // encontrar t a partir do índice i (incluído)

i = *s*.**indexOf**(*c*);

i = *s*.**indexOf**(*c*, i);

i = *s*.**lastIndexOf**(*c*); // índice da última ocurrência de c em s

i = *s*.**lastIndexOf**(*c*, *i*); // encontrar c antes do índice i (inc.)

i = *s*.**lastIndexOf**(*t*);

i = *s*.**lastIndexOf**(*t*, *i*);

String

Saber o Conteúdo

c = *s*.**charAt**(*i*); // caracter na posição i

t = *s*.**substring**(*i*); // substring desde i (inc) até ao fim

t = *s*.**substring**(*i*, *j*); // substring entre i (inc) e j (exc)

String

- Criar uma string a partir de outra

  t = *s*.**toLowerCase**(); // converter tudo para minúsculas

  t = *s*.**toUpperCase**() ; // converter tudo para maiúsculas

  t = *s*.**trim**(); // retira espaços do início e do fim

  t = *s*.**replace**(*c1*, *c2*); //substituir todos caracteres c1 por c2;

  t = *s*.**replace**(*cs2*, *cs3*); // idem para substrings

String

Expressões Regulares

(ver java.util.regex.Pattern)

b = *s*.**matches**(*expReg*); // true se expReg valida toda a string
//(idêntico a Pattern.matches(regexStr, s))

s1 = *s*.**replaceAll**(*expReg*, *t*); // substitui toda a substring validada por expReg pela String t

s1 = *s*.**replaceFirst**(*expReg*, *t*);

sa = *s*.**split**(*expReg*); // array de todas substrings terminadas por expReg

sa = *s*.**split**(*expReg*, *num*); // idem apenas num vezes

String

Métodos Estáticos

para converter para Strings

s = String.**valueOf**(*x*); // Converte x para String

s = String.**format**(*f*, x...); // utiliza um formato f para converter um numero variável de parâmetros x, (como o caso do fprintf)

Caractere

Tipo	Bits	Mínimo	Máximo
char	16	0	65 535

‘A’
\x41
\0101
\u0041

65 em decimal

\n	Mudança de Linha
\r	Voltar ao início da linha
\b	Ir uma posição para a esquerda
\t	Um tab
\f	Mudança de página
\’	Um apóstrofo
\”	Uma aspa
\\	A própria barra invertida
\uhhhh	Caractere unicode (0-9, A-F)
\0ooo	Caractere octal (0-7)
\xhh	Caractere hexadecimal (0-9, A-F)

Character

Métodos

- b = Character.**isDigit**(*c*);
- b = Character.**isLetter**(*c*);
- b = Character.**isLetterOrDigit**(*c*);
- b = Character.**isLowerCase**(*c*);
- b = Character.**isUpperCase**(*c*);
- b = Character.**isWhitespace**(*c*);
- c = Character.**toLowerCase**(*c*);
- c = Character.**toUpperCase**(*c*);

StringBuffer

- Manipula caracteres que serão alterados durante a execução do programa.

```
StringBuffer b = new StringBuffer();
b.append(“Olá”);
b.append(“Bom ”);
b.append(“Dia.”);
String s = b.toString();
```

StringBuilder - StringBuffer

StringBuilder = StringBuffer (sincronizado)

Construtores

sb = **new StringBuilder**() ; //vazio

sb = **new StringBuilder**(*n*); //capacidade inicial de n

sb = **new StringBuilder**(*s*) ; //iniciado com s

Dimensão

i = *sb*.**length**(); // dimensão do conteúdo de sb

StringBuilder

Alteração de conteúdo

(devolvem o original; o mesmo objecto alterado)

sb =*sb*.**append**(*x*); // acrescenta x no fim de sb

sb =*sb*.**insert**(*i*, *x*); // insere x na posição i

sb =*sb*.**setCharAt**(*i*, *c*); //Substitui o caracter em i por c

sb =*sb*.**deleteCharAt**(*i*);

sb =*sb*.**delete**(*i*, *j*); //Apaga a gama entre i e j

sb =*sb*.**reverse**(); // odúetnoc o etrevni

sb =*sb*.**replace**(*i, j*, *s*); // substitui cracteres de i a j por s

StringBuilder

Saber o conteúdo

c = *sb*.**charAt**(*i*);

s = *sb*.**substring**(*i*); // substring do índice i até ao fim

s = *sb*.**substring**(*i*, *j*);

s = *sb*.**toString**(); //nova String com o conteúdo

StringBuilder

Pesquisa – (devolvem -1 caso não seja encontrado)

i = *sb*.**indexOf**(*t*); // índice da primeira ocorrência de t em sb
i = *sb*.**indexOf**(*t*, i); // encontrar t a partir do índice i (incluído)

i = *sb*.**lastIndexOf**(*t*); // índice da última ocurrência de t em sb
i = *sb*.**lastIndexOf**(*t*, *i*); // encontrar t antes do índice i (inc.)

Comparação

b = *sb*.**equals**(*sb2*); // true (sb==sb2)

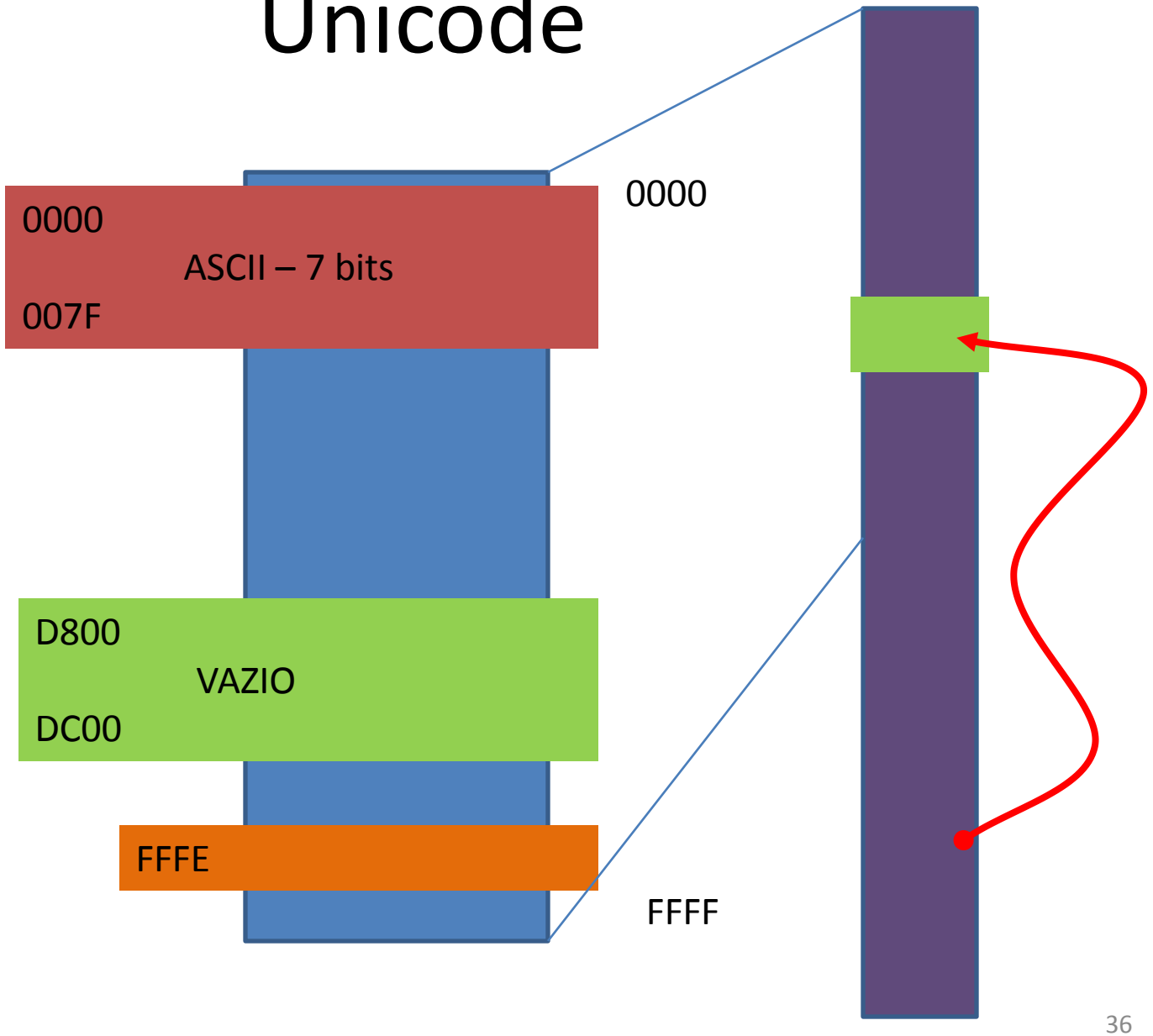
Unicode

- Quantos caracteres diferentes existem no mundo e que possam ser representados num computador?
- Até ver mais de 107 mil => 17 bits

- ASCII – 8 bits => 128 padrões fixos + 128
- UTF-16 – 16 bits => 65536
- UTF-8
- UTF-32

Unicode

- UTF-16



Unicode

- UTF-16
- Valores maiores do que FFFF (16 bits)
  - Ex: 10 FFFD (21 bits) - retira-se 10 0000

  u = 0 FFFD (20 bits) 0000 1111 1111 1111 1101

  uh = 0000 1111 11 (10 bits) 0000 0000 0011 1111 (16 bits)

  ul = 11 1111 1101 (10 bits) 0000 0011 1111 1101 (16 bits)

  Utf-16 = 0xD800 | uh , 0xDC00 | ul

D800
VAZIO
DC00

Big Endian - Little Endian

0x12345678

12	34	56	78
HH	HL	LH	LL

HH	12
HL	34
LH	56
LL	78

LL	78
LH	56
HL	34
HH	12

Unicode

Utf-16 = 0xD83F , 0xDFFD

D83F	DFFD
H	L

FEFF	D83F	DFFD

BOM – Byte Order Mask

FFFE	DFFD	D83F

Mensagem

BOM							

Iterator

Lista (simples) de elementos ligados; um sentido

Disponibilidade de valores

b=it.**hasNext**(); // true se existe um elemento seguinte

x=it.**next**(); // o próximo elemento

it.**remove**(); // remove o elemento devolvido por it.next();

ListIterator

Lista de elementos duplamente ligados

Disponibilidade de valores

b=lit.**hasNext**(); // true se existe um elemento seguinte

x=lit.**next**(); // o próximo elemento

i=lit.**nextIndex**(); // Índice do elemento a devolver por next();

b=lit.**hasPrevious**(); // true se existe elemento anterior

x=lit.**previous**(); // o elemento anterior

x=lit.**previousIndex**(); // Índice do elemento a devolver por previous();

ListIterator

Alteração de valores

lit.**add**(x); // insere x antes do elemento a ser devolvido por lit.next();

lit.**set**(x); // substitui o elemento devolvido por lit.next(); por x

lit.**remove**(); // remove o elemento devolvido por lit.next();

Scanner

Lê valores de uma String, File ou InputStream

Constructores

sc = new Scanner(System.in);

sc = new Scanner(*s*);

sc = new Scanner(*f*);

Scanner

Disponibilidade de dados

b = *sc*.**hasNext**(); // true se há um token (palavra)

b = *sc*.**hasNextLine**(); // true se há mais uma linha

b = *sc*.**hasNextInt**(); // true se houver um inteiro

b = *sc*.**hasNextDouble**();

b = *sc*.**hasNext*XYZ***(); // true se houver BigDecimal, BigInteger, Boolean, Byte, Float, Short, ou tipo primitivo.

Scanner

Entrada

s = *sc*.**next**(); // devolve o token (palavra) seguinte

s = *sc*.**nextLine**(); // devolve a linha seguinte

i = *sc*.**nextInt**(); // devolve o inteiro seguinte

d = *sc*.**nextDouble**(); // devolve o double seguinte

x = *sc*.**nextXYZ**(); // devolve um valor do tipo primitivo se possível, ou BigDecimal, BigInteger, Boolean, Byte, Float, Short.

x = *sc*.**next*XYZ***(*radix*); // idem Int, Long, Short, Byte, ou BigInteger convertido para a base radix

Scanner

Fechar

sc.**close**(); //fecha o canal de leitura, liberta o recurso para ser usado por outros threads

Referências

http://www.leepoint.net/notes-java/index.html

http://www.exampledepot.com/

http://tektonia.com