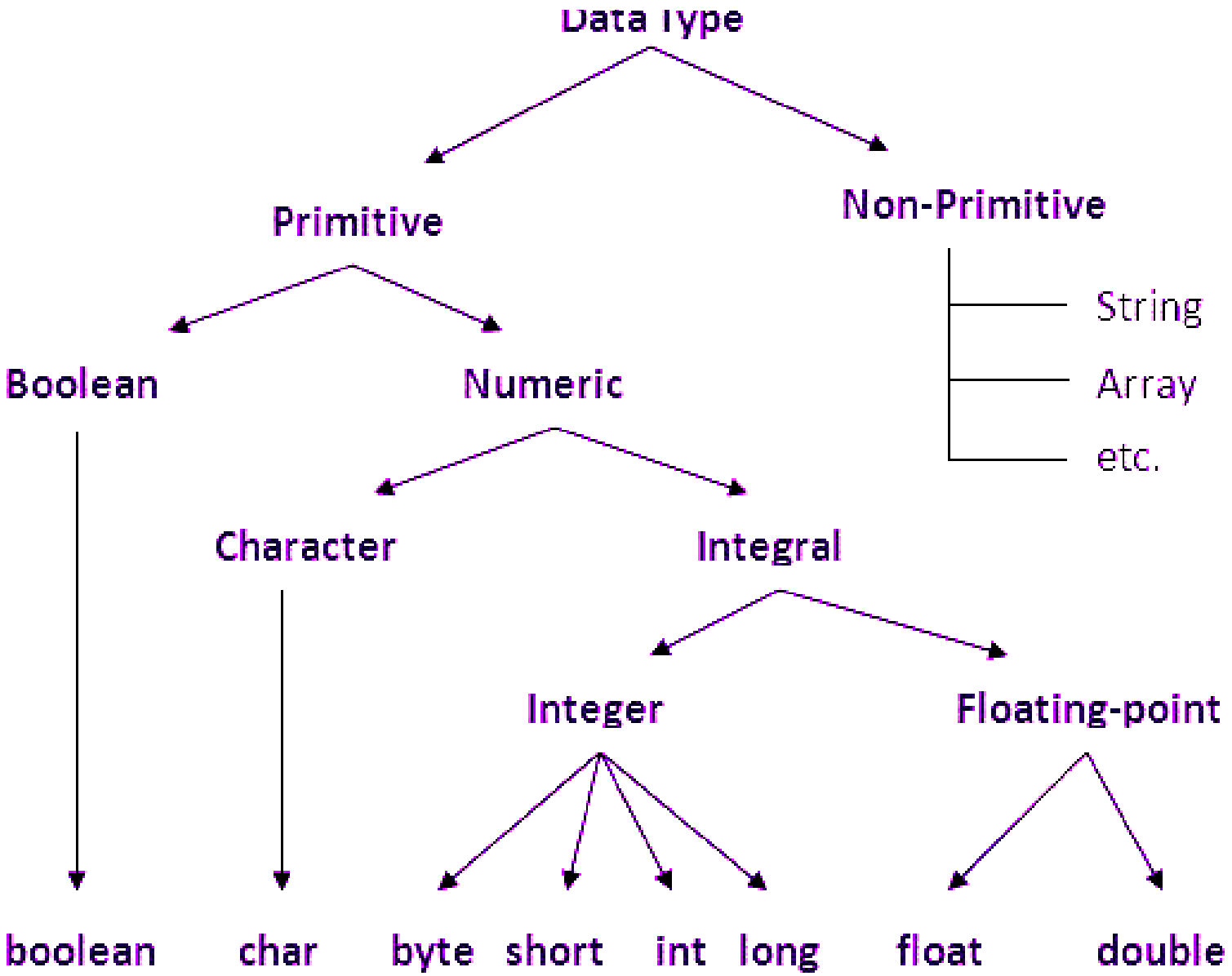
Processamento da Informação

Caracteres e Strings – Parte 2

Prof. Jesús P. Mena-Chalco
CMCC/UFABC

Q2/2018

Tipos de dados em Java



(*) Fonte: http://www.javatpoint.com/variable-datatype

Vetor de caracteres

`v.length → 6`

v:=

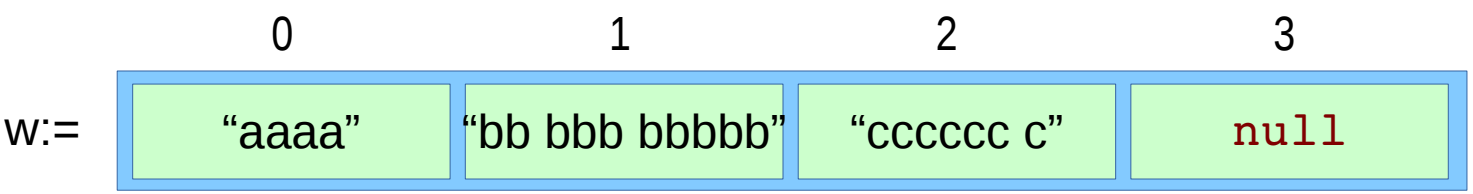
0	1	2	3	4	5
'a'	'b'	'c'			

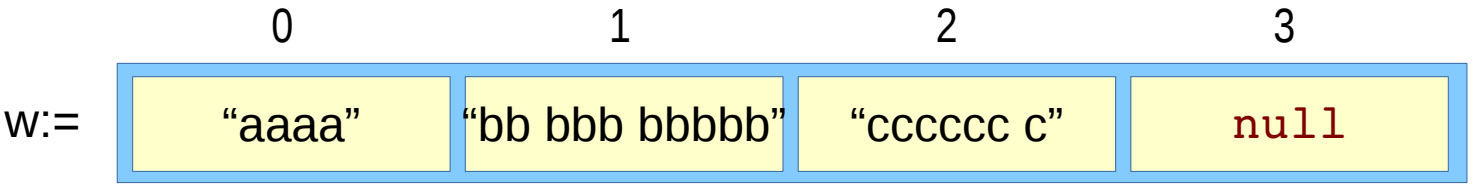
v:=

0	1	2	3	4	5
97	98	99	0	0	0

Vetor de strings

`w.length → 4`





Métodos especiais de Strings

(1) toUpperCase / toLowerCase

```
String s1 = "aaaa AAAA";
String s2 = "bbbBBB";

System.out.println( s1.toUpperCase() );
System.out.println( s1.toLowerCase() );

System.out.println( s2.toUpperCase() );
System.out.println( s2.toLowerCase() );
```

```
AAAA AAAA
aaaa aaaa

BBBBBB
bbbbbb
```

(2) toCharArray

```
int i;
String nome = "Universidade Federal do ABC";
char v[] = nome.toCharArray();

for (i=0; i<v.length; i=i+1) {
    System.out.print( v[i] + "*");
}
```

U*n*i*v*e*r*s*i*d*a*d*e* *F*e*d*e*r*a*l* *d*o* *A*B*C*

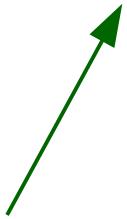
(3) toCharAt

```
                    01234567890123456789012345
String nome = "Universidade Federal do ABC";

System.out.println( nome.charAt(0) );
System.out.println( nome.charAt(13) );
System.out.println( nome.charAt(24) );
System.out.println( nome.charAt(25) );
System.out.println( nome.charAt(26) );
```

```
U
F
A
B
C
```

(4) length()

```
String nome = "Universidade Federal do ABC";

System.out.println( nome.length() );
```



```
27
```

(5) substring

```
                  01234567890123456789012345678
String nome = "Universidade Federal do ABC";

System.out.println( nome.substring(13) );
System.out.println( nome.substring(13, 20) );
```

Federal do ABC
Federal

Operadores relacionais (e.g., <, ==) não são apropriados para **comparar** objetos.

```
String nome1 = "Paulo";
String nome2 = "XPaulo";

nome2 = nome2.substring(1);

if ( nome1 == nome2 ) {
    System.out.println("Nomes iguais");
}
else {
    System.out.println("Nomes diferentes");
}
```

Nomes diferentes

(6) equals

```
String nome1 = "Paulo";
String nome2 = "XPaulo";

nome2 = nome2.substring(1);

if ( nome1.equals(nome2) ) {
    System.out.println("Nomes iguais");
}
else {
    System.out.println("Nomes diferentes");
}
```

Nomes iguais

(6) equals

```
String nome1 = "Paulo";
String nome2 = "XPAULO";

nome2 = nome2.substring(1);

if ( nome1.equalsIgnoreCase(nome2) ) {
    System.out.println("Nomes iguais");
}
else {
    System.out.println("Nomes diferentes");
}
```

Nomes iguais

Exercícios

Exercício 01: Contando palavras

Crie um método que permita **contar** o número de palavras presentes em uma string.

Assinatura:

```
static int palavras1( String frase )
```

Exercício 01: Contando palavras

```
static int palavras1( String frase ) {
    int i, cont=0;

    for (i=0; i<frase.length(); i=i+1) {
        if (frase.charAt(i)==' ') {
           cont = cont+1;
        }
    }

    return cont+1;
}
```

```
System.out.println( palavras1("Processamento da Informacao Teoria") );
System.out.println( palavras1("  Processamento  da  Informacao  Teoria  ") );
```

4
11

Exercício 01: Contando palavras

```
static int palavras1( String frase ) {
    int i, cont=0;

    for (i=0; i<frase.length(); i=i+1) {
        if (frase.charAt(i)==' ') {
           cont = cont+1;
        }
    }

    return cont+1;
}
```

O método conta o número de espaços em branco.

Modifique o método!

Exercício 01: Contando palavras

```
static int palavras2( String frase ) {
    int i, cont=0;
    char anterior=' ';

    for (i=0; i<frase.length(); i=i+1) {
        if (frase.charAt(i)!=' ' && anterior==' ') {
           cont = cont+1;
        }
        anterior = frase.charAt(i);
    }

    return cont;
}
```

```
System.out.println( palavras2("Processamento da Informacao Teoria") );
System.out.println( palavras2("  Processamento  da  Informacao  Teoria  ") );
```

```
4
4
```

Exercício 02: Primeiro nome

Crie um método que devolva o primeiro nome de uma pessoa.

Observação:
Assuma que existe apenas um espaço em branco entre os nomes.

Assinatura:

```
static String primeiroNome( String nome )
```

Exemplos:

```
nome = "Joao Carlos da Costa"
```
Resposta = **"Joao"**

```
nome = "JoaoCarlos"
```
Resposta = **"JoaoCarlos"**

Exercício 02: Primeiro nome

```
static String primeiroNome( String nome ) {
    int i;
    String resposta  = "";

    for (i=0; i<nome.length() && nome.charAt(i)!=' '; i=i+1) {
        resposta = resposta + nome.charAt(i);
    }

    return resposta;
}
```

Exercício 03: Último nome

Crie um método que devolva o último nome de uma pessoa.

Observação:
Assuma que existe apenas um espaço em branco entre os nomes.

Assinatura:

```
static String ultimoNome( String nome )
```

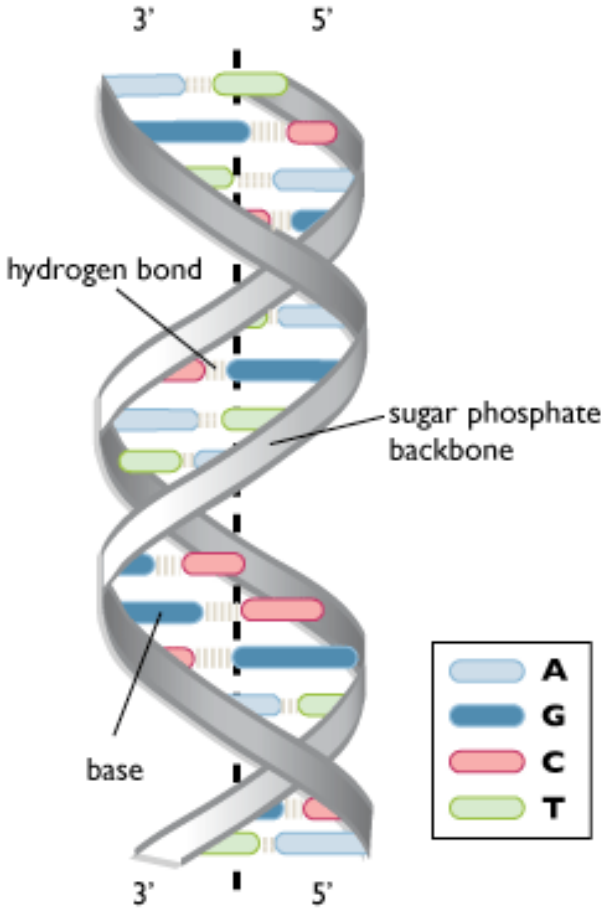
Exemplos:

```
nome = "Joao Carlos da Costa"
```
Resposta = **"Costa"**

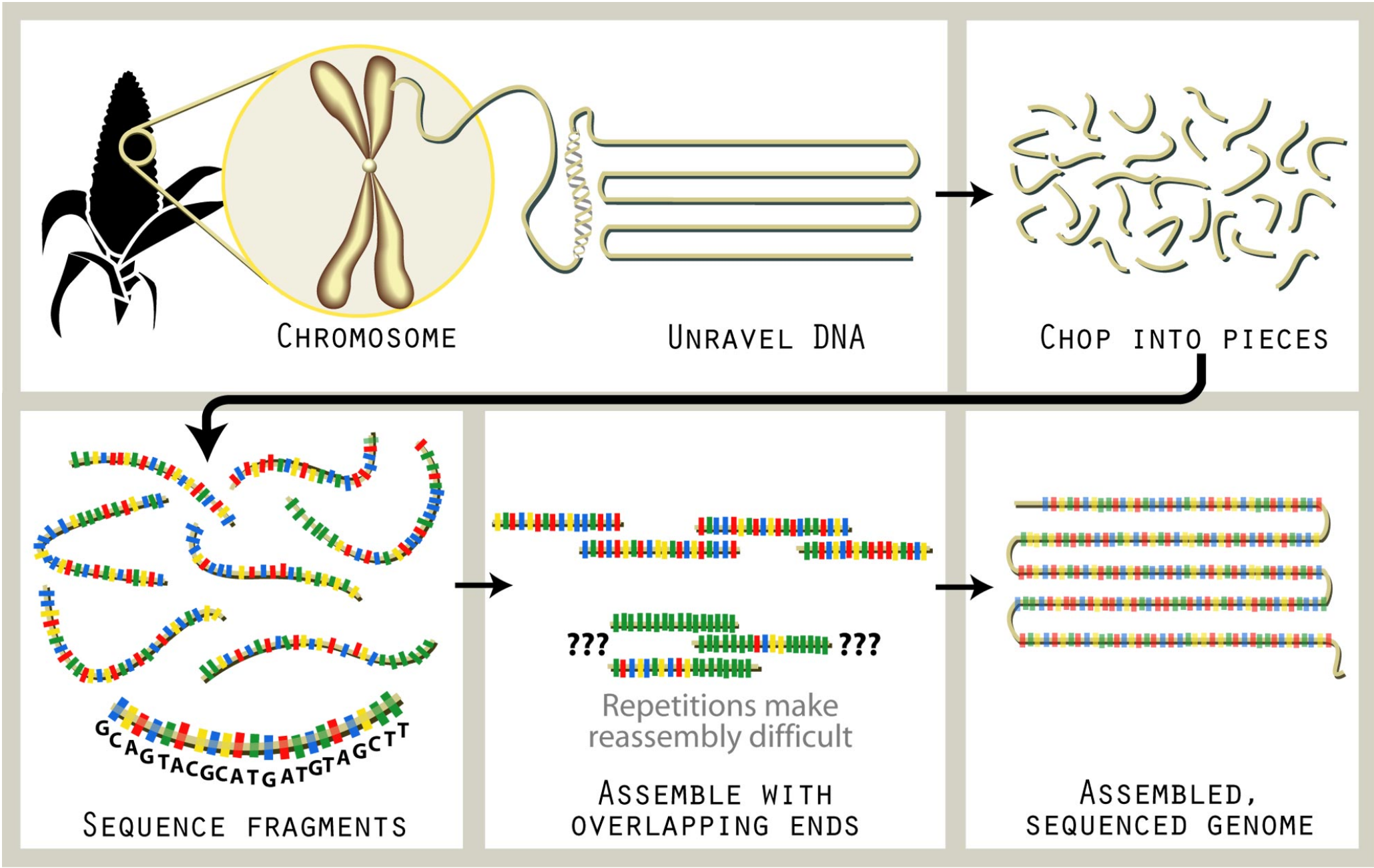
```
nome = "JoaoCarlos"
```
Resposta = **"JoaoCarlos"**

Exercício 03: Último nome

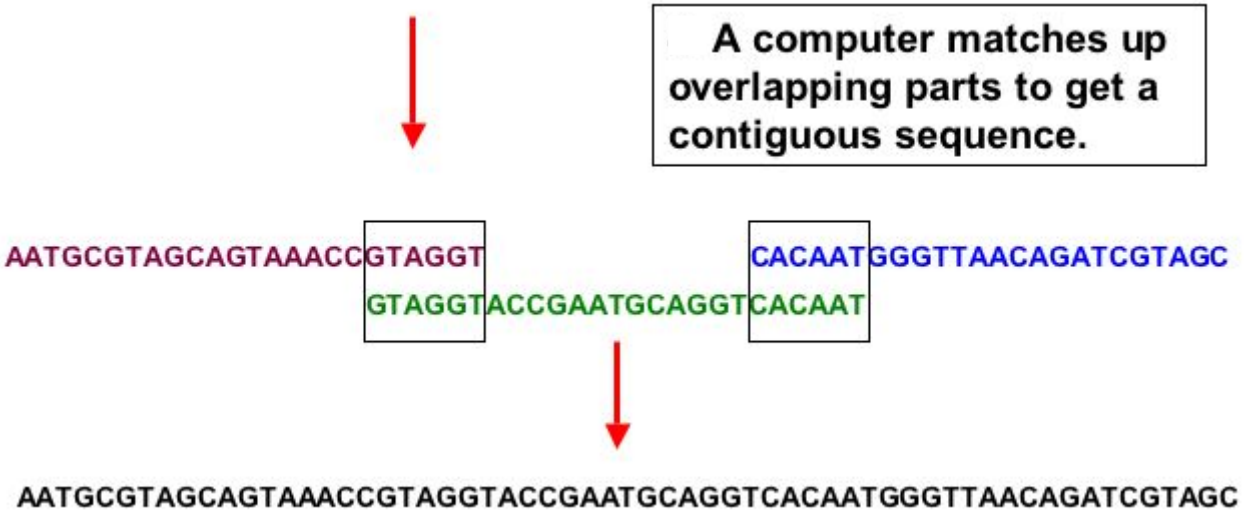
```
static String ultimoNome( String nome ) {
    int i;
    String resposta  = "";

    for (i=nome.length()-1; i>=0 && nome.charAt(i)!=' '; i=i-1) {
        resposta = nome.charAt(i)+resposta ;
    }

    return resposta;
}
```



Watson & Crick (1953)



Fonte: https://www.nsf.gov/news/news_images.jsp?cntn_id=111175&org=NSF


A computer matches up overlapping parts to get a contiguous sequence.
AATGCGTAGCAGTAAACCGTAGGT
CACAATGGGTTAACAGATCGTAGC
GTAGGTACCGAATGCAGGTCACAAT
AATGCGTAGCAGTAAACCGTAGGTACCGAATGCAGGTCACAATGGGTTAACAGATCGTAGC

Exercício 04: Contar `ttt` em sequência de DNA

Crie um método que permita **contar** o número de vezes que a subsequência '`ttt`' está em uma sequência de DNA.

Obs: Considere letras minúsculas e sequência >= a 3 caracteres.

Assinatura:

```
static int contarTTT( char V[] )
```

Exemplos:

- `V = {'t','t','t'}`
  Resposta = 1

- `V = {'t','t','t','t','t'}`
  Resposta = 0

- `V = {'t','t','t','a','t','c','t','t','t'}`
  Resposta = 2

Exercício 04: Contar `ttt` em sequência de DNA

```
static int contarTTT( char V[] ) {
    int i, cont=0, n=V.length;

    for (i=0; i<=n-3; i=i+1) {
        if ( (V[i]=='t' && V[i+1]=='t' && V[i+2]=='t') && ( i+3==n || V[i+3]!='t') && ( i==0 || V[i-1]!='t' ) ) {
            cont = cont+1;
        }
    }
    return cont;
}
```

Atividade em aula

Questão 1 - a

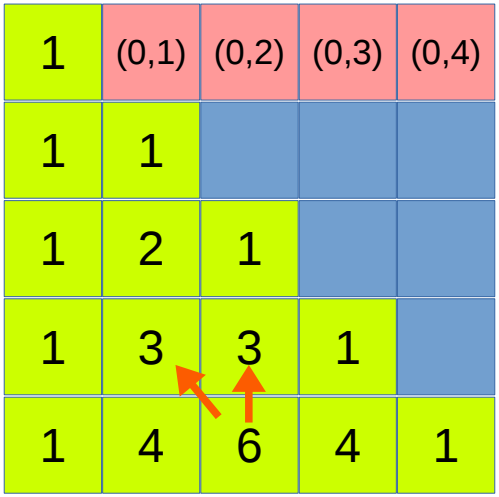
```
static void metodo1 () {
    char[] v = {'0', '1'};
    int i, j, k;

    for (i=0; i<2; i=i+1) {
        for (j=0; j<2; j=j+1) {
            for (k=0; k<2; k=k+1) {
                System.out.println(v[i]+" "+v[j]+" "+v[k]);
            }
        }
    }
}
```

```
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
```

Questão 1 - b

```
static void metodo2 () {
    int i, j;
    int M[][] = new int[8][8];

    for (i=0; i<8; i=i+1) {
        for (j=0; j<=i; j=j+1) {
            if (j==0 || i==j) {
                M[i][j] = 1;
            }
            else {
                M[i][j] = M[i-1][j] + M[i-1][j-1];
            }
        }
    }

    for (i=0; i<8; i=i+1) {
        for (j=0; j<=i; j=j+1) {
            System.out.print(M[i][j]+" ");
        }
        System.out.print("\n");
    }
}
```

Questão 1 - b



Questão 1 - b

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

```
          1
        1 1
      1 2 1
    1 3 3 1
  1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

Triângulo de pascal de ordem 8

Questão 2

```
static int brancos ( char[] v ) {
    int i, cont=0;

    for (i=0; i<=v.length; i=i+1) {
        if (v[i]==' ') {
            cont = cont+1;
        }
    }

    return cont;
}
```

Defeito:
Erro no índice `i=v.length.`

O método estaria correto considerando, no lugar de:
`i<=v.length`
A seguinte instrução:
`i<v.length`

Questão 3

```
static boolean palindromo ( char[] v ) {
    int i;

    for (i=0; i<v.length; i=i+1) {
        if (v[i]!=v[v.length-i-1]) {
            return false;
        }
    }

    return true;
}
```

Defeito:
O método funciona corretamente, entratanto, são realizadas mais consultas das necessárias.

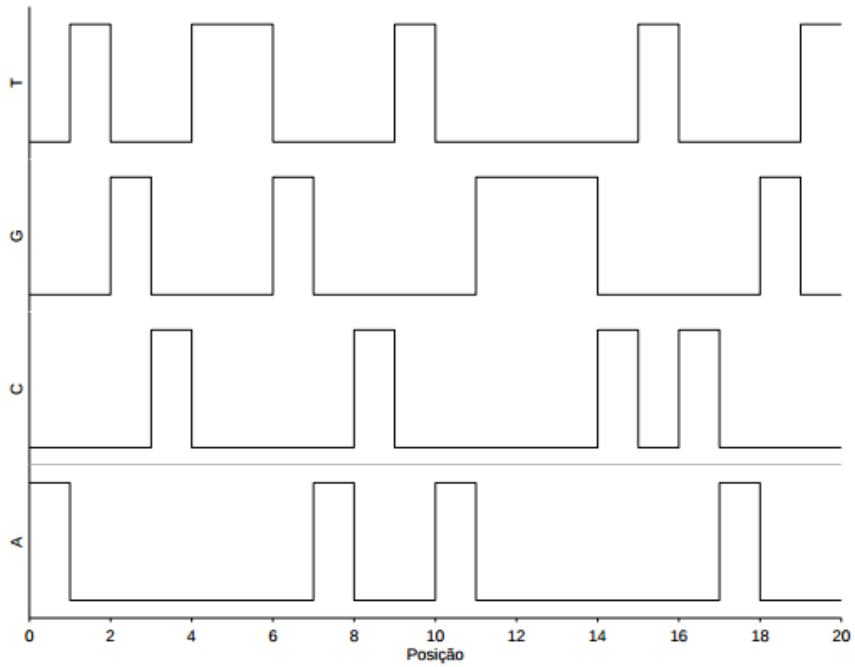
O método seria mais eficiente quando considerado, no lugar de:
`i<v.length`
A seguinte instrução:
`i<v.length/2`

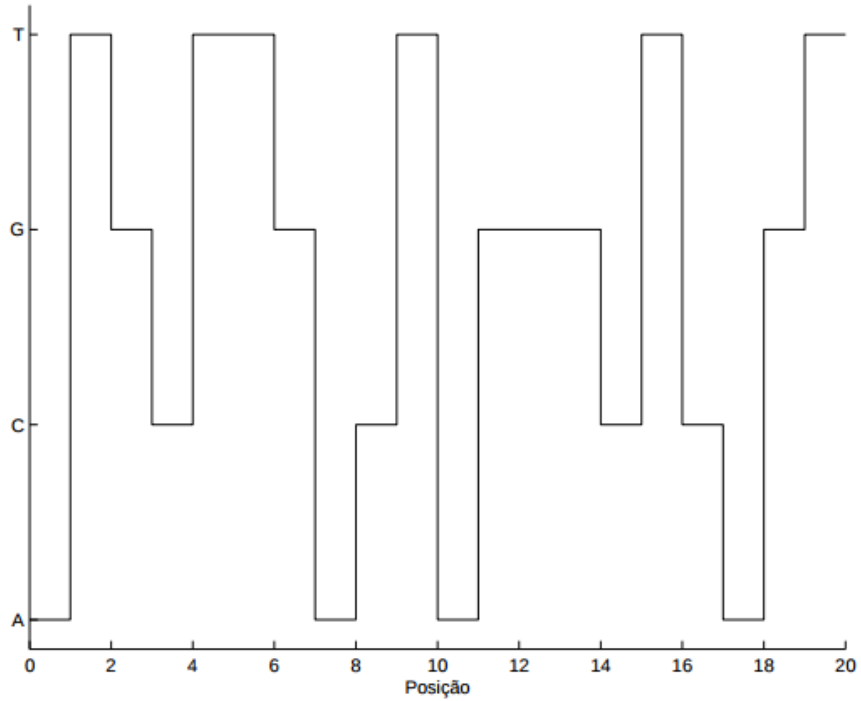
Questão 4

```
static int[][] dna2bin( char DNA[] ) {
    int i;
    int M[][] = new int[4][DNA.length];

    for (i=0; i<DNA.length; i=i+1) {
        if ( DNA[i]=='a') {
            M[0][i] = 1;
        }
        if ( DNA[i]=='c') {
            M[1][i] = 1;
        }
        if ( DNA[i]=='g') {
            M[2][i] = 1;
        }
        if ( DNA[i]=='t') {
            M[3][i] = 1;
        }
    }
    return M;
}
```

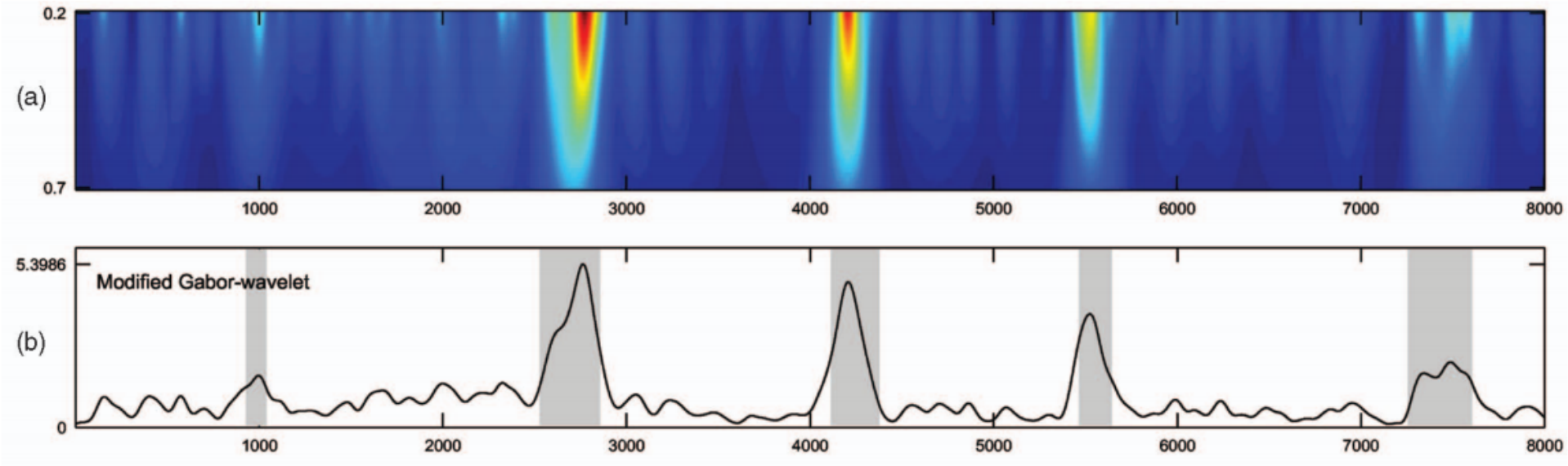
Questão 4

Seqüência	A	T	G	C	T	T	G	A	C	T	A	G	G	G	C	T	C	A	G	T
u_A	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
u_C	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0
u_G	0	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	0
u_T	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1





Questão 4


0.2
(a)
0.7
1000
2000
3000
4000
5000
6000
7000
8000
5.3986
Modified Gabor-wavelet
(b)
0
1000
2000
3000
4000
5000
6000
7000
8000