Departamento de Informática

Introdução à Programação

Capítulo 5

Tipos estruturados

Desenvolvimento para Web e Dispositivos Móveis

Joana Fialho

E-mail: jfialho@estgv.ipv.pt

Gilberto Rouxinol

E-mail: rouxinol@estgv.ipv.pt

Escola Superior de Tecnologia e Gestão de Viseu

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Problema: Aceitar e registar as notas dos alunos de IP

Solução 1: usar tantos atributos quantos os alunos com o nome do aluno e aceitar a nota de cada aluno, atribuindo-a a cada um dos atributos.

No entanto, irão existir muitos atributos e muito código repetido; não se podem usar ciclos; é preciso saber quantos alunos são para poder definir um número suficiente de atributos.

Solução 2: criar um atributo especial que permite guardar as notas todas, e.g., como uma tabela. O atributo do tipo tabela tem o nome da turma em vez do nome do aluno. Para obter o nome de um determinado aluno o atributo especial do tipo tabela tem um índice cuja função é indicar a posição do nome do aluno na tabela. À posição UM na tabela corresponde o índice ZERO, à posição DOIS na tabela corresponde o índice UM, ... , à posição N corresponde o índice N-1.

Assim, só existe um atributo (nome da turma); podem usar-se uma das estruturas de repetição, estudadas em aulas anteriores, para aceder a cada um dos seus atributos (nome do aluno); o código fica condensado; eventualmente, no tempo de execução, dá para criar a tabela com a dimensão adequada, e.g., a dimensão da turma.

Nota: Uma tabela armazena atributos de igual tipo.

A **solução 2** vai usar um atributo do tipo estruturado**:** ***TABELA ou ARRAY***

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

Uma tabela tem um determinado nome único e contém um conjunto de elementos todos do mesmo tipo

-Declaração e criação de uma tabela

(1) O exemplo seguinte de declaração mostra que tab faz referência a uma tabela de inteiros

```
...
int tab[];
...
```

ou

```
...
int [] tab;
...
```

qual usar ?

```
...
int [] tab1, tab2;
...
int tab1[], n, tab2[];
```

permite mistura

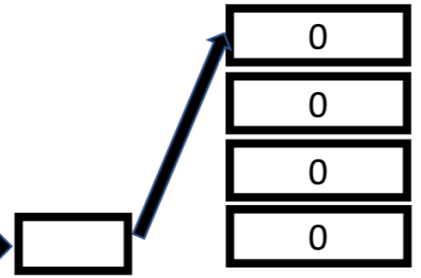
Não dá indicação da dimensão da tabela tab e nenhum valor foi atribuído à tabela tab. A sua declaração é muito próxima da declaração de um objeto

(2) A declaração de uma tabela não deve indicar a sua dimensão – a instrução será rejeitada pelo compilador

```
...
int tab[4];
...
```

Erro

(3) Para criar uma tabela usa-se o mesmo principio que se usa para criar um objeto



Esta declaração reserva memória suficiente para armazenar 4 elementos do tipo int e coloca a referência em tab. Os 4 elementos são inicializados por defeito com o valor zero (trata-se de int's)

```
...
tab = new int [4];
...
```

```
...
n = sc.nextInt();
tab = new int [n];
...
```

```
int r, s;
...
int tab[] = {0, r, 2*r, s, 2*s, 5};
...
```

ou

```
int r, s, tab[];
...
tab = new int [6];
tab[0] = 0;
tab[1] = r;
tab[2] = 2*r;
tab[3] = s; tab[4] = 2*s; tab[5] = 5};
```

```
tab = {0,r, 2*r, s, 2*s, 5} Erro
As chavetas só se utilizam numa declaração
```

Rouxinol, G. A. F.

Capítulo 5 – Tipos estruturados
Tabelas: conceito e declaração

Tabelas em JAVA

-Utilização de uma tabela

(1) Acesso individual

```
...
int tab[] = new int[4];
...
t[0] = 1;
...
t[3]++ ;
...
```

Na posição 0 da tabela tab é colocado o valor 1

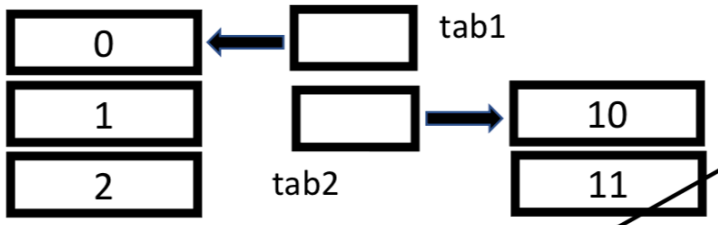
É incrementado de uma unidade o valor que está na quarta posição

Para tabelas "à medida" ver a classe Vector do pacote java.util

(2) Acesso global à tabela

É possível manipular globalmente as tabelas através das suas referências

```
...
int [] tab1 = new int[3];
for (int  i=0; i<3; i++) tab1[i]=i;
int [] tab2 = new int[2];
for (int  i=0; i<2; i++) tab2[i]=10+i;
...
```

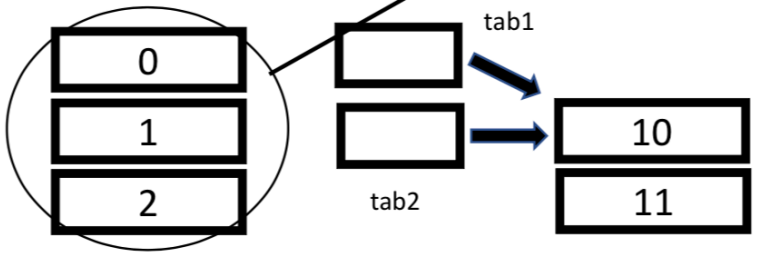


Se durante um período de tempo nada se fizer com este objeto que contém os 3 inteiros, no passado referenciado por tab1, este será apanhado pelo garbage collection

Se se fizer

```
...
tab1 = tab2;
...
```

tab1 e tab2 são agora as mesmas tabelas




```
...
tab1[0] = 1000;
System.out.println(tab2[0]);
```

será afixado 1000 e não 10

xinol, G. A. F.

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

-Dimensão de uma tabela

(1)A declaração de uma referência de uma tabela não indica a dimensão da tabela

Para tal usar o atributo length

```
…
int tab[] = new int[4];
System.out.println("Número de elementos de Tabela:" + tab.length);  mostra 4
tab = new int[2];
System.out.println("Número de elementos de Tabela:" + tab.length);  mostra 2
…
```

Nota: length é um atributo e não um método - não se escreve length()

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Curiosidade: Para estudar mais tarde – em vez de ser do tipo int é do tipo Ponto !

Lembrete: int é tipo primitivo (letra minúscula); Ponto é tipo classe (letra Maíscula)

Tabelas em JAVA

-Exemplo de uma tabela de objetos da classe Ponto

```
public class ExemploTabelaObjetos{
    public static void main(String[] args){
        Ponto [] pt;
        tp = new Ponto[3];
        tp[0] = new Ponto(10,40);
        tp[1] = new Ponto(20,50);
        tp[2] = new Ponto(30,60);

        for( int i = 0; i < 3; i++ ){
            tp[i].escreve();
        }
    }
}
```

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

-Utilização do ciclo for…each

(1) Trata-se de uma estrutura de controlo adaptada às coleções, às tabelas e às cadeias

Considere-se:

```
…
double tab [];
…
for (double var : tab) System.out.println(var);
```

equivalente a

```
…
double tab [];
…
for (int i = 0; i < tab.length; i++) System.out.println(tab[i]);
```

A variável var toma sucessivamente os valores da tabela tab e mostra-os

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

-Utilização da tabela em argumento

(1) Quando se utiliza o nome de uma tabela como argumento num determinado método o que se está a fazer é a passar uma referência da tabela. O método vai assim atuar diretamente na tabela e não na cópia.

```
public class TabelaArgumento {
    public static void main (String[] args) {
        int tab [] = {1, 2, 3, 4};

        System.out.print("Antes: ");
        Funcao.escreve(tab);
        Funcao.zera(tab);

        System.out.print("Depois: ");
        Funcao.escreve(tab);
    }
}
```

```
class Funcao {

    static void zera (int tabela[]) {
        for (int i = i; i < tabela.length; i++)
            tabela[i] = 0;
    }

    static void escreve (int tabela[]) {
        for (int var : tabela) System.out.println(var + " ");
    }

}
```

Idem para tabelas devolvidas por métodos

```
public static int [] maisCoisas (int n) {
    int [] nova = new int[n];
    for (int i = i; i < n; i++) nova[i] = 3*i;
    return nova;
}
```

Antes: 1 2 3 4
Depois: 0 0 0 0

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

-Tabelas com vários índices

(1)Uma tabela com dois índices, e.g., permite criar uma matriz

(2)O JAVA não tem esta noção matemática de matriz mas permite-a simular criando tabelas de tabelas

(3)Esta abordagem acaba por ser mais rica do que a das tabelas convencionais a vários índices de outras linguagens

(4)Permite tabelas irregulares, cada linha tem uma determinada dimensão

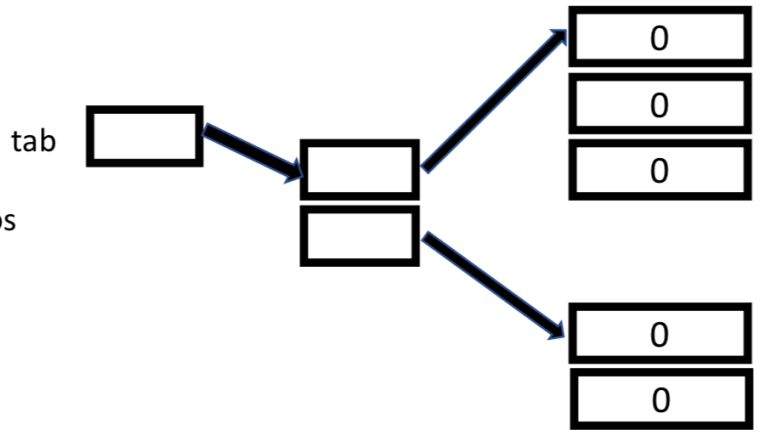
(5)Obviamente é possível construir e manipular tabelas equivalentes às matrizes

```
int tab [] [];
int [] tab [];
int [] [] tab;
```

A declaração de tab é uma referência a uma tabela, onde cada elemento é ele próprio uma referência a uma tabela de int's

```
int tab [] [] = { new int [3], new int [2] };
```

A inicialização de tab é feita com 2 elementos que por sua vez criam duas tabelas: a primeira com 3 int e a segunda com 2 int



tab[0] – referência da 1.ª tabela
tab[0][1] – 2.º elemento da 1.ª tabela
tab[] – referência da 2.ª tabela
tab[1][i-1] – o i-ésimo elememto da 2.ª tabela
tab.length vale 2
tab[0].length vale 3
tab[1].length vale 2

Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

-Tabelas com vários índices

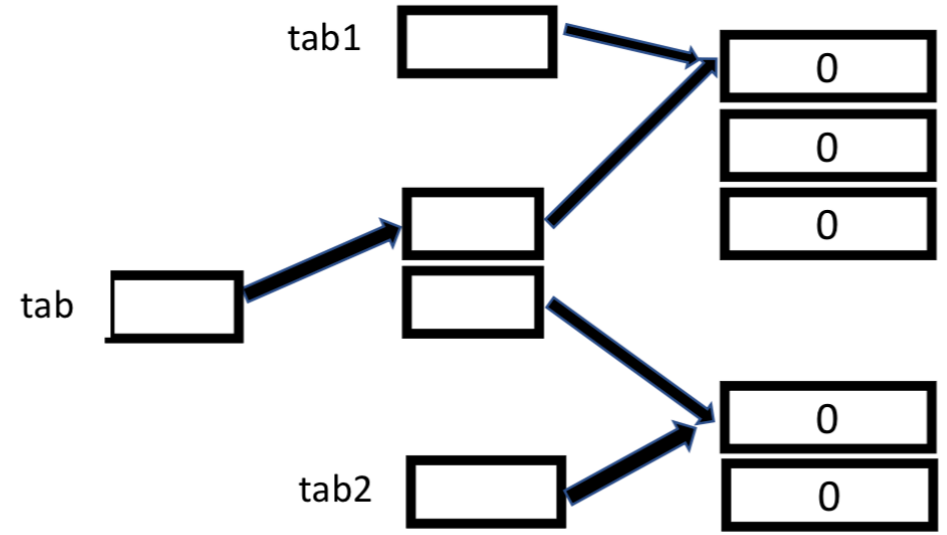
```
int tab [] [];

tab = new int [2] [];

int [] tab1 = new int [3];
int [] tab2 = new int [2];

tab[0] = tab1;
tab[1] = tab2;
```

São criadas duas referências tab1 e tab2 para as duas tabelas uma de 3 int e outra de 2 int



Capítulo 5 – Tipos estruturados

Tabelas: conceito e declaração

Tabelas em JAVA

-Tabelas com vários índices

```
…
int tab [] [] = { {1, 2, 3}, {10, 20} };
…
```

```
…
for (int i = 0; i < NLINHAS; i++)
    for(int j = 0; j < NCOLUNAS; j++)
        tab [i] [j] = 0;
…
```

Inicialização de uma matriz e.g.

Dimensão da tabela:
Número de linhas -> tab.length
Número de colunas -> tab[0].length

Capítulo 5 – Tipos estruturados

Tabelas: Mais métodos

A class Arrays (java.util.Arrays**) – Métodos**

1 public static int binarySearch(Object[ ] a, Object key)

Procura na tabela de Objetos (byte, int, double, etc.) o valor especificado usando o algoritmo de pesquisa binária. A tabela deve ser ordenada antes da sua chamada. O método devolve o índice do valor da pesquisa, se estiver contido na tabela; caso contrário, devolve (- (ponto de inserção + 1))

2 public static boolean equals(long[ ] a, long[ ] a2)

Devolve true se as duas tabelas de longs forem iguais. Duas tabelas são consideradas iguais se ambas contiverem o mesmo número de elementos e todos os pares de elementos correspondentes nas duas tabelas forem iguais. O método devolve verdadeiro se as duas tabelas forem iguais. O método pode ser usado com outros tipos de dados primitivos (byte, short, int, etc.)

3 public static void fill(int[ ] a, int val)

Atribui o valor int a cada elemento da tabela de ints. O método pode ser usado com outros tipos de dados primitivos (byte, short, int, etc.)

4 public static void sort(Object[ ] a)

Ordena a tabela de objetos por ordem crescente de acordo com a ordem natural dos seus elementos. O método pode ser usado com outros tipos de dados primitivos (byte, short, int, etc.)

Capítulo 5 – Tipos estruturados

Tabelas: Exemplos

Determinar a média da turma e o número de alunos com nota superior à média

```java
package ip.tabelas;
import ip.teclado.Teclado;
public class MainMediaTurma {
    public static void main (String args[]) {
        int i;
        int nAlunos;
        int nAlunosSupMed;
        double soma;
        double media;

        System.out.print ("Numero de alunos = ") ;
        nAlunos = Teclado.lerInt();

        double notas[] = new double[nAlunos] ;

        for (i = 0; i < nAlunos; i++) {
            do {
                System.out.print ("Nota do aluno numero " + (i+1) + " = " ) ;
                notas[i] = Teclado.lerDouble();
            } while( notas[i] < 0 || notas[i] > 20 );
        }

        for (i = 0, soma = 0; i < nAlunos; i++)
            soma += notas[i] ;

        media = soma / nAlunos ;

        System.out.println ("\nMedia da turma =  " + media) ;

        for (i = 0, nAlunosSupMed = 0 ; i < nAlunos ; i++ )
            if (notas[i] > media) nAlunosSupMed++ ;

        System.out.println ("Numero de alunos com nota superior a media = " + nAlunosSupMed) ;
    }
}
```

Capítulo 5 – Tipos estruturados

Tabelas: Exemplos

Manipulação de uma tabela bi-dimensional

1/2

```java
package ip.tabelas;

public class MainManipulacaoTabelas {
    public static void main(String[] args) {
        int t[] [] = { {1, 2, 3}, {11, 22}, {111, 222, 333, 444} } ;
        System.out.println ("t antes de zerar(): ") ;
        mostrar(t) ;
        mostrarForEach(t);
        zerar(t) ;
        System.out.println ("t depois de zerar(): ") ;
        mostrar(t);
        mostrarForEach(t);
    }

    public static void zerar (int t[] []) {
        int i;
        int j;
        for (i = 0 ; i < t.length; i++)       // for... each nao aplicavel aqui
            for (j = 0 ; j < t[i].length; j++) // ha modificacao dos valores de t
                t[i] [j] = 0 ;
    }

    public static void mostrar (int t[] []) {
        int i, j;
        for (i = 0 ; i < t.length; i++) {
            System.out.print ("linha " + i + " = ") ;
            for (j = 0; j < t[i].length; j++)         // para utilizar o for... each
                System.out.print (t[i] [j] + " ") ;   // ver o metodo mostrarForEach()
            System.out.println() ;
        }
    }
```

Capítulo 5 – Tipos estruturados

Tabelas: Exemplos

Manipulação de uma tabela bi-dimensional

2/2

```
    public static void mostrarForEach (int t[] []) {
        int i, j;
        int nLinha = 0;
        for (int[] linha : t) {
            System.out.print ("linha " + nLinha++ + " = ") ;
            for (int v : linha)
                System.out.print (v + " ") ;
            System.out.println() ;
        }
    }

}
```

Capítulo 5 – Tipos estruturados

Tabelas: Problemas

Problema

Escreva um programa que: (1) adicione; (2) multiplique, duas matrizes 5x5

Capítulo 5 – Tipos estruturados

Strings: conceito e declaração

Classe String em JAVA

-Uma String é uma tabela de carateres, está implementada numa classe do pacote java.lang e permite manipular cadeias de carateres

-As constantes tais que "dwdmip" não são mais do que objetos do tipo String criados automaticamente pelo compilador

-Há várias funcionalidades nesta classe: comprimento; acesso aos carateres de uma cadeia pela posição; concatenação; pesquisa de um caráter ou subcadeia; conversão entre tipos primitivos e tipos String; substituições para criar novos objetos; substituição ou passagem de minúsculas para maiúsculas e vice-versa

```
...
String nome;
...
```

como todas as declarações de uma variável objeto, declara-se que "nome" vai armazenar uma referência para um objeto do tipo String


```
...
nome =  "dwdmip";
...
```

esta notação indica a criação de um objeto do tipo String que é feita automaticamente pelo compilador

```
String nome1 = new String ();
String nome2 = new String ("dwdm");
String nome3 = new String (nome2)
```

nome1 é uma referência a uma cadeia vazia
nome2 é uma referência a uma cadeia que contém "dwdm"
nome3 é uma referência a uma cadeia cópia de nome2, i.e., contém "dwdm"

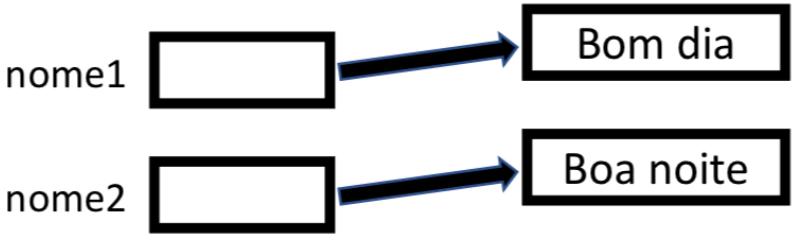
Capítulo 5 – Tipos estruturados

Strings: conceito e declaração

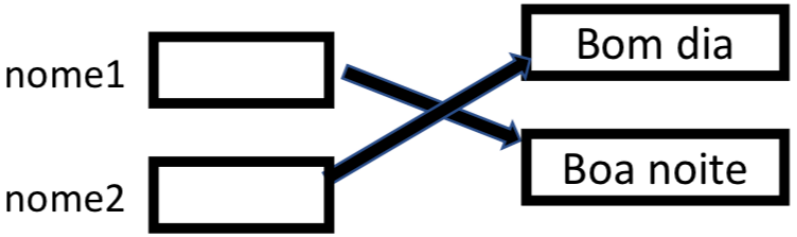
String em JAVA

-Um objeto String não é modificável

```
…
String nome1, nome2, nome;
nome1 = “Bom dia”;
nome2 = “Boa noite”;
…
nome = nome1;
nome1 = nome2;
nome2 = nome;
…
```



os dois objetos do tipo String não foram modificados mas as suas referências sim



Capítulo 5 – Tipos estruturados

Strings: conceito e declaração

String em JAVA

-Entrada e saída

```
...
System.out.println("Bom dia");
...
String nome;
...
System.out.println(nome);
...
nome = sc.nextLine();
...
```

-Comprimento de uma String

```
...
String nome = "Bom dia";
int n = nome.length(); // n=7
...
```

Nota: length() é um método e não um atributo - não se escreve length

Capítulo 5 – Tipos estruturados

Strings: conceito e declaração

String em JAVA

-Acesso aos carateres numa string: charAt

```
…
String nome = “Bom dia”;
…
char a = nome.charAt(0); // devolve ‘B’
char b = nome.charAt(4); // devolve ‘d’
…
```

-Concatenação de strings

```
…
String nome1 = “Bom ”;
String nome2 = “Dia”;
String nome = nome1 + nome2;// “Bom Dia”
…
```

Capítulo 5 – Tipos estruturados

Strings: conceito e declaração

String em JAVA

-Outros métodos

substring(n,m) ou substring(n)– devolve uma substring da original (método com assinatura diferente - overloaded)

toLowerCase() – devolve string original em minúsculas (como produz alteração devolve nova string e deixa intacta a original)

equals(stringacomparare) – compara duas strings

indexOf(substring) – devolve posição da primeira letra da substring

trim() – limpa a string nas extremidades

concat – concatenação de strings (funcionamento em cascata)

Capítulo 5 – Tipos estruturados

Strings: conceito e declaração

Classe StringTokenizer em JAVA

-Permite dividir a string em substrings designadas tokens e faz parte do pacote java.util

-os separadores que identificam a separação destas sustring são os espaços, tab e newline

```
String curso = "Desenvolvimento para a Web e Dispositivos Móveis";
StringTokenizer substr = new StringTokenizer (curso);
While(substr.hasMoreElements()){
    String str = substr.nextToken();
    System.out.println(str);
}

Consola ->   Desenvolvimento
             para
             a
             Web
             e
             Dispositivos
             Móveis
```

Capítulo 5 – Tipos estruturados

Strings: mais métodos

1 char charAt (int index)

Devolve o carater no índice especificado

2 int compareTo (Object o)

Compara a String com o objeto o

3 int compareTo (String anotherString)

Compara as duas strings lexicograficamente

4 int compareToIgnoreCase (String str)

Compara as duas strings lexicograficamente ignorando as diferenças entre maiúsculas e minúsculas

5 String concat (String str)

Junta a string especificada str, no final da string

6 boolean contentEquals (StringBuffer sb)

Devolve true se e somente a string representa a mesma sequência de caracteres que o StringBuffer sb

7 static String copyValueOf (char [ ] data)

Devolve uma string que representa a sequência de caracteres na tabela data

8 static String copyValueOf (char [ ] data, int offset, int count)

Devolve uma string que representa a sequência de caracteres na tabela data

Capítulo 5 – Tipos estruturados

Strings: mais métodos

9 boolean endsWith (String suffix)

Testa se a string termina com o sufixo suffix

10 boolean equals (Object anObject)

Compara a string com o objeto anObject

11 boolean equalsIgnoreCase (String anotherString)

Compara a string com outra string, ignorando considerações de caso

12 byte [ ] getBytes ()

Codifica a string em uma sequência de bytes usando o conjunto de caracteres padrão da plataforma,

armazenando o resultado em uma nova tabela de bytes

13 byte [ ] getBytes (String charsetName)

Codifica a String em uma sequência de bytes usando o nome do “charset”, armazenando o resultado em uma nova tabela de bytes

14 void getChars (int srcBegin, int srcEnd, char [ ] dst, int dstBegin)

Copia caracteres da string na tabela de caracteres de destino

15 int hashCode ()

Devolve o código hash da string

16 int indexOf (int ch)

Devolve o índice dentro da cadeia de caracteres da primeira ocorrência do caractere especificado

Capítulo 5 – Tipos estruturados

Strings: mais métodos

17 int indexOf (int ch, int fromIndex)

Devolve o índice dentro da cadeia de caracteres da primeira ocorrência do caracter ch, iniciando a pesquisa no índice fromIndex

18 int indexOf (String str)

Devolve o índice da cadeia de caracteres da primeira ocorrência da substring str

19 int indexOf (String str, int fromIndex)

Devolve o índice da cadeia de caracteres da primeira ocorrência da substring str, iniciando no índice fromIndex

20 String intem ()

Devolve uma representação canônica para o objeto string

21 int lastIndexOf (int ch)

Devolve o índice dentro da cadeia de caracteres da última ocorrência do caracter ch

22 int lastIndexOf (int ch, int fromIndex)

Devolve o índice dentro da cadeia de caracteres da última ocorrência do caracter ch, pesquisando para trás, iniciando no índice fromIndex

23 int lastIndexOf (String str)

Devolve o índice da sequência, da ocorrência mais à direita da substring str

24 int lastIndexOf (String str, int fromIndex)

Devolve o índice dentro da cadeia de caracteres da última ocorrência da substring str, pesquisando para trás, iniciando no índice fromIndex

Capítulo 5 – Tipos estruturados

Strings: mais métodos

25 int length ()

Devolve o comprimento da string

26 boolean matches (String regex)

Informa se a sequência corresponde ou não à expressão regular regex

27 boolean regionMatches (boolean ignoreCase, int toffset, String other, int ooffset, int len)

Testa se duas regiões da sequência são iguais

28 boolean regionMatches (int toffset, String other, int ooffset, int len)

Testa se duas regiões de sequência são iguais

29. String replace (char oldChar, char newChar)

Devolve uma nova string resultante da substituição de todas as ocorrências de oldChar na string por newChar

30 String replaceAll (String regex, String replacement)

Substitui cada substring da string que corresponde à expressão regular regex pela substituição replacement

31 String replaceFirst (String regex, String replacement)

Substitui a primeira substring da string que corresponde à expressão regular regex pela substituição replacement

32 String [] split (String regex)

Divide a string em torno das correspondências da expressão regular regex

Capítulo 5 – Tipos estruturados

Strings: mais métodos

33 String [] split (String regex, int limit)

Divide a string em torno das correspondências da expressão regular regex

34 boolean startsWith (String prefix)

Testa se a sequência começa com o prefixo prefix

35 boolean startsWith (String prefix, int toffset)

Testa se a sequência começa com o prefixo prefix, iniciando-o no índice toffset

36 CharSequence subSequence (int beginIndex, int endIndex)

Devolve uma nova sequência de caracteres que é uma subsequência da string

37 String substring (int beginIndex)

Devolve uma nova string que é uma subcadeia da string

38 String substring de string (int beginIndex, int endIndex)

Devolve uma nova string que é uma subcadeia da string

39 char [ ] toCharArray ()

Converte a sequência de caracteres em uma nova tabela de caracteres

40 String toLowerCase ()

Converte todos os caracteres da string em minúsculas usando as regras do código do idioma padrão

Capítulo 5 – Tipos estruturados

Strings: mais métodos

41 String toLowerCase (Locale locale)

Converte todos os caracteres da string em minúsculas usando as regras da Localidade locale

42 String toString ()

Este objeto (que já é uma string) é devolvido

43 String toUpperCase ()

Converte todos os caracteres da string em maiúsculas usando as regras do código do idioma padrão

44 String toUpperCase (Locale locale)

Converte todos os caracteres da string em maiúsculas usando as regras da Localidade locale.

45 String trim ()

Devolve uma cópia da string com espaços em branco à esquerda e à direita limpos – sem espaços brancos nas extremidades

46 static String valueOf (primitive data type x)

Devolve a representação em string do tipo de argumento passado.

Strings: Problemas

Problemas

(1)Escreva um programa que faça de corretor ortográfico para a seguinte frase:

“A melhor unidade curricular do curso de Desenvolvimento para a Web e Dispositivos Móveis é Introdução à Programação”

(2)Escreva um programa que leia o nome de 10 alunos, a nota do exame 1 e do exame 2 e depois faça a média dos dois exames. Implemente também um método que permita determinar a média da turma.

Capítulo 5 – Tipos estruturados

Enumerados: