Ciências ULisboa

Programação Centrada em Objetos

2021/2022

Série 2

I – A classe `String`

Revisões

1. O que sabe sobre a mutabilidade/imutabilidade dos objetos do tipo `String`?
2. Contrariamente à classe `Math` já sua conhecida, a maior parte dos métodos da classe `String` não é `static` ("de classe").
   a) qual a diferença entre métodos "de instância" e métodos "de classe"?
   b) dê exemplos de uns e outros na classe `String` e exemplifique a sua invocação.

Exercícios

Nos exercícios que se seguem, sempre que adequado construa métodos com bom potencial de reutilização. Para cada método definido deve escrever um cabeçalho incluindo uma descrição sucinta e geral do método e, se for caso disso, `@param`, `@requires` *e* `@return`.
Pode incluir os métodos numa classe `UtilsPCO` *(que não tem método* `main`*). No método* `main` *de outras classes, sempre que quer invocar um método* `m(...)` *da classe* `UtilsPCO`*, deverá usar uma invocação qualificada:* `UtilsPCO.m(...)`

3. Escreva um método que, dadas uma frase e uma letra, calcule e devolva o número total de vezes que a letra aparece na frase.

   Construa um programa que invoque o método para uma dada frase e caráter e imprima no ecrã o seu resultado.

4. Escreva um método que, dada uma frase `f` e um valor positivo `n` menor que metade do comprimento da frase, devolva a *string* resultante de concatenar os `n` primeiros carateres de `f` com os últimos `n`.

   Construa um programa que invoque o método para uma dada frase e inteiro e imprima no ecrã o seu resultado em maiúsculas.

5. Escreva um programa que, começando numa palavra definida por si, imprima no ecrã as palavras resultantes de ir retirando, um a um, os carateres do meio da palavra anterior, até chegar a uma palavra com um só caráter. Se o número de carateres da palavra for par, será retirado o primeiro dos dois carateres do meio.

   Exemplo: se for `rapsodia` a palavra por si escolhida, o programa deverá imprimir as palavras `rapsodia`, `rapodia`, `rapdia`, `radia`, `raia`, `ria`, `ra` e `a`.

   Para isso, construa uma função com a assinatura `static String semLetraMeio (String s)` que devolva a *string* resultante de "retirar" o caráter do meio de `s`.

6. A equipa TUTEDECC (Tem Um TExto DEmasiado Claro? Contacte-nos!) pretende fazer um estudo sobre várias técnicas de obscurecimento de mensagens. Vamos ajudá-los a transformar textos, obscurecendo-os, de variadas e diferentes maneiras, para posteriormente serem apresentados a um grupo de "cobaias" que os tentarão ler.

Para isso, e recorrendo às classes `String` e `StringBuilder` do pacote `java.lang`, crie métodos que implementem as várias técnicas de obscurecimento de textos que a equipa TUTEDECC pretende. São eles:

a) `static String semVogais (String frase)` que devolve uma *string* resultante de retirar as vogais a todas as palavras de `frase`, excetuando aquelas palavras totalmente compostas por vogais; essas não serão alteradas.

Exemplo: O texto `O Pai Natal nao existe` daria origem a `O P Ntl n xst`

b) `static String espacoNaN (String frase, int n)` que devolve uma *string* resultante de, após apagar os espaços entre as palavras de `frase`, inserir um espaço a cada `n` carateres (o texto obtido nunca deverá terminar com espaço).

Exemplo. Para `n`=3, o texto `O Pai Natal nao existe` ficaria `OPa iNa tal nao exi ste`

c) `static String palavrasInversas (String frase)` que devolve uma *string* resultante de inverter todas as palavras de `frase` individualmente.

Exemplo: O texto `O Pai Natal nao existe` ficaria `O iaP lataN oan etsixe`

d) `static String inversaNemN (String frase, int n)` que devolve uma *string* resultante de inverter os carateres do texto `n` a `n` carateres (incluindo espaços); o último bloco é sempre invertido mesmo que tenha menos que `n` carateres.

Exemplo. Para `n`=3, o texto `O Pai Natal nao existe` ficaria `P O iataN laoanxe tsie`

e) `static String shiftDeN (String frase, int n)` que devolve uma *string* resultante de fazer um *shift* à direita de `n` posições a cada palavra (`n` pode não ser menor que o tamanho das palavras).

Exemplo. Para `n`=2, o texto `O Pai Natal nao existe` ficaria `O aiP talNa ona teexis`

7. Escreva um programa que defina duas variáveis e as inicialize com um texto e um inteiro e imprima os textos resultantes de aplicar todas as técnicas do exercício anterior.

II – *Arrays* de uma dimensão / Vetores

Revisões

8. Qual o primeiro e último índice que podemos usar para acesso aos elementos do vetor criado pela expressão `new int[7]`?
9. O que acontece se usarmos um índice fora desses limites para aceder ao vetor?
10. Quais os valores iniciais dos elementos dos vetores criados pelas expressões `new int[7]`, `new double[7]`, `new boolean[7]`, `new char[7]`, `new String[7]` , `new Dado[7]`?
11. Podemos alterar o número de elementos de um vetor em tempo de execução?
12. Podemos passar um vetor como argumento na invocação de um método? Podemos ter um vetor como resultado de um método?

Exercícios

13. Escreva um programa em Java que

a) declara duas variáveis inteiras `m` e `n` e inicializa-as com valores à sua escolha

b) cria um vetor de inteiros com `n` posições

c) preenche esse vetor com os `n` primeiros múltiplos de `m`

d) imprime no ecrã os elementos desse vetor

e) calcula e imprime o elemento desse vetor mais próximo da média

14. Acrescente os seguintes métodos à classe `UtilsPCO`, não se esquecendo de, para cada um deles, apresentar um cabeçalho *javadoc* apropriado:

a) `boolean existe (int[] v, int x)` que verifica se existe alguma ocorrência de `x` em `v`;

b) `int quantos (int[] v, int x)` que conta o número de ocorrências do valor `x` em `v`;

c) `boolean iguais (int[] v, int[] w)` que verifica se os vetores `v` e `w` são iguais;

d) `int[] junta (int[] v, int[] w)` que devolve um vetor contendo os elementos de `v` seguidos dos elementos de `w`;

e) `int[] juntaOrdenado (int[] v, int[] w)` que recebe dois vetores ordenados de forma crescente, e devolve um vetor ordenado da mesma forma cujos elementos são todos os que pertencem a `v` ou `w`;

f) `void substitui (int[] v, int valAntes, int valDepois)` que substitui todas as ocorrências do valor `valAntes` em `v` pelo valor `valDepois`;

g) `void inverte (int[] v)` que inverte a ordem dos elementos de `v`. Faça-o de duas formas diferentes:

   i. recorrendo a um vetor auxiliar;

   ii. sem recorrer a outro vetor*;*

h) `int[] maxMin (int[] v)` que constrói e devolve um vetor com 2 inteiros, correspondentes às posições onde se encontram, no vetor `v`, os seus valores máximo e mínimo, respetivamente;

i) `int[] doisMaiores (int[] v)` que constrói e devolve um vetor de duas posições contendo os dois máximos do vetor `v`, assumindo que `v` tem pelo menos duas posições;

j) `int[] pedeVetor (int n, Scanner sc)` que, assumindo que `n` é positivo e `sc` não é `null`, pede ao utilizador `n` valores inteiros e devolve um vetor com esses valores;

k) `void imprimeVetor (int[] v)` que escreve no *standard output* os elementos do vetor `v` entre parenteses retos, separados por vírgulas.

15. Estenda o programa do exercício 13 de modo a executar também os seguintes passos:

a) declarar e inicializar uma variável inteira `k`;

b) imprimir no ecrã o número de valores do vetor original que são iguais a `k`;

c) imprimir no ecrã os valores máximo e mínimo do vetor original;

d) criar um novo vetor de inteiros com `k` elementos dados pelo utilizador;

e) imprimir no ecrã os elementos do vetor resultante de juntar os dois vetores;

f) inverter a ordem dos elementos do vetor original e imprimir o vetor no ecrã.

Sempre que possível invoque os métodos do exercício anterior.

16. Acrescente à classe `UtilsPCO` o método

```
int quantosMaiores (double[] v, int n)
```

que, assumindo que `v` é diferente de `null`, determina o número de elementos de `v` cujo valor arredondado às unidades é igual ou superior a `n`. *Exemplo*: se `v` for `{10.34 , 9.73 , 10.8 , 9.42 , 8.6 , 7.0}` e `n` for `10`, a função deverá devolver o valor `3`. Use a função `long round(double f)` da classe `Math`. Apresente também um cabeçalho *javadoc* apropriado.

17. Usando o método da alínea anterior, escreva um programa que:

a) cria um vetor com os valores `10.34 , 9.73 , 10.8 , 9.42 , 8.6 e 7.0`

b) imprime no ecrã o número de elementos desse vetor cujo valor arredondado às unidades é maior que 10.

Assuma que o método `quantosMaiores` está definido na classe `UtilsPCO` já criada.

18. Um ponto no plano é definido por duas coordenadas, as suas componentes horizontal e vertical. Suponha que tem disponível uma classe `Point` cujos objetos representam pontos no plano, e cuja documentação inclui a informação apresentada abaixo.

a) Assumindo que a classe `Point` está implementada corretamente, descreva o que acontece quando são executados os dois seguintes pedaços de código.

```
Point origin = new Point(0,0);
int x = 6;
int n = -1;
Point p = new Point(4, x);
p.translate(n, n*2);
System.out.println("Abcissa= " + p.x() + "   Ordenada= " + p.y());
System.out.println(p.distance(origin));

Point p1 = new Point(0, 0);
Point p2 = p1;
Point p3;
p1.translate(1, 1);
System.out.println(p2.toString());
p2.translate(3, 3);
System.out.println(p1.toString());
System.out.println(p3.toString());
```

b) Construa os seguintes métodos para a classe `UtilsPCO`:

i. `static Point maisDistante (Point[] vp)` que devolve o ponto mais distante da origem em `vp`;

ii. `static int comprimento (Point[] vp)` que calcula o comprimento total dos segmentos de reta que ligam os pontos consecutivos em `vp`;

iii. `static Point[] distantesEntreSi (Point[] vp)` que devolve os dois pontos consecutivos mais distantes entre si em `vp`.

Constructor Summary	
Point(double x, double y) Constructs and initializes a point at the specified (*x*, *y*) location in the coordinate space.	
Method Summary	
boolean	**equalsPoint**(Point other) Determines whether or not this point is equal to another point.
double	**x**() Returns the X coordinate of this point in double precision.
double	**y**() Returns the Y coordinate of this point in double precision.
Point	**copy**() Returns a copy of this point.
double	**distance**(Point p) Returns the distance from this Point to a specified point.
String	**toString**() Returns a string representation of this point and its location in the (*x*, *y*) coordinate space.
void	**translate**(double dx, double dy) Moves this point by dx along the *x* axis and dy along the *y* axis.

19. Acrescente à classe `UtilsPCO` o método

```
String palavrasNovas (int[] v, String fonte)
```

que constrói e devolve uma *string* contendo uma palavra com tantas letras quantas o tamanho de `v`, formada por letras da *string* `fonte`; o conteúdo do vetor `v` indica as posições na *string* `fonte` onde se devem ir buscar as letras para formar a nova *string*; se a posição indicada não existe na *string* `fonte`, a letra a usar deverá ser a da última posição de `fonte`. Use a classe `StringBuilder`. Apresente também um cabeçalho *javadoc* apropriado**.**

Exemplo: se `v` for `{5,0,4,3,15,7,5,4}` e `fonte = "ABEDILOGUY"`, o resultado será `"LAIDYGLI"`.

20. Acrescente à classe `UtilsPCO` o método

```
static String melodia (String[] notas, int n)
```

que constrói e devolve uma string resultante de juntar aleatoriamente `n` *strings* do vetor `notas` separadas por pontos. *Exemplo*: se `notas` for `{"la" , "mi" , "sol"}` e `n` for `7`, um possível resultado será `"mi.mi.sol.la.mi.la.la"`

Use as classes `Random` e `StringBuilder`. Apresente também um cabeçalho *javadoc* apropriado**.**

21. Acrescente à classe `UtilsPCO` o método

```
int[] contaOcorrencias (int[] v, int top)
```

que devolve um vetor com `top` elementos, contendo em cada *posição* o número de ocorrências no vetor `v` do valor correspondente à *posição* (onde *posição* varia entre 0 e `top` -1). Caso `v` tenha valores **fora** do intervalo 0 a `top` -1 (inclusive) estes **não** serão contabilizados. Por exemplo, se o procedimento for chamado com o vetor `{0,-1,3,1,6,0,5,2,12,3,0}` e valor `top 6`, então o resultado será `{3,1,1,2,0,1}` (zero tem três ocorrências, um e dois têm uma ocorrência cada, três tem duas ocorrências, quatro não tem ocorrências, cinco tem uma ocorrência). Assuma que o vetor a analisar não é vazio e que `top` é um valor positivo. Apresente também um cabeçalho *javadoc* apropriado.

22. Acrescente à classe `UtilsPCO` o método

```
static String familia(String s, String[] v)
```

que constrói e devolve uma *string* resultante de juntar as *strings* do vetor `v` que contêm `s`, separadas por pontos. *Exemplo*: se `v` for {"`mareado`" , "`longe`" , "`Pai Natal`" , "`amarelo`", "`rema`"} e `s` for "`mar`", o resultado será "`mareado.amarelo`". Apresente também um cabeçalho *javadoc* apropriado**.**

Use a classe `StringBuilder`.

Use também o método `public int indexOf (String str)`, da classe `String`, que retorna a posição da primeira ocorrência de `str` na *string* alvo da invocação, ou -1 se `str` não ocorre. Exemplos: Se String `alvo = "OlaAdeus"`, então `alvo.indexOf("Ola")` retorna 0, `alvo.indexOf("Adeus")` retorna 3 e `alvo.indexOf("ela")` retorna -1.

III – *Arrays* multidimensionais / Tabelas

Revisões

23. Como são implementadas em Java tabelas bidimensionais?

24. O que caracteriza uma matriz?

25. No contexto da declaração `int[][] m = new int[5][3]`

a) na expressão `m[i][j]` que valores pode `i` tomar? E `j`?

b) qual o tipo de `m[0][0]`? E de `m[1]`? E de `m`? E de `m[5][3]`?

Exercícios

26. Escreva um programa em Java que

a) declara e inicializa duas variáveis inteiras (chamemos-lhe `m` e `n`)

b) cria uma tabela bidimensional de inteiros com `m x n` posições

c) preenche essa tabela com os sucessivos múltiplos de 3, começando com o próprio 3

d) imprime no ecrã os elementos dessa tabela, linha a linha.

27. Os *arrays* bidimensionais em que todas as linhas têm o mesmo número de elementos, são chamados de **matrizes**. Programe em Java um método

1	0	0
0	1	0
0	0	1

```
boolean identidade (int[][] m)
```

que, assumindo que `m` é uma matriz, determina se `m` é a matriz identidade, ie., se `m` é uma matriz quadrada (mesmo número de linhas e colunas) e se `m` tem `0` em todas as posições menos nas da diagonal principal, onde tem `1` (ver exemplo ao lado, de matriz identidade `3x3`).

Apresente também um cabeçalho *javadoc* apropriado**.**

28. Programe em Java um método

```
boolean simetrica (int[][] m)
```

2	5	10
5	11	15
10	15	30

que, assumindo que `m` é uma matriz, determina se `m` é simétrica, ie., se `m` é uma matriz quadrada (mesmo número de linhas e colunas) onde as linhas são iguais às colunas, na respetiva ordem. Ver exemplo ao lado, de matriz simétrica `3x3`: a primeira linha (2, 5, 10) é igual à primeira coluna, a segunda linha (5, 11, 15) é igual à segunda coluna e a terceira linha (10, 15, 30) é igual à terceira coluna. Apresente também um cabeçalho *javadoc* apropriado**.**

29. Programe em Java um método

```
void incrementaValoresNoIntervalo (int[][] m, int inf, int sup)
```

que incrementa de uma unidade todos os elementos de `m` cujos valores estejam no intervalo [`inf`,`sup`]. Apresente também um cabeçalho *javadoc* apropriado**.**

30. Programe em Java um método

```
boolean transposta (int[][] m1, int[][] m2)
```

que, assumindo que `m1` e `m2` são matrizes, determina se `m1` é transposta de `m2`, ie., se as linhas de `m1` são iguais às colunas de `m2`, na ordem correspondente. Ver exemplo ao lado, de matrizes transpostas. Apresente também um cabeçalho *javadoc* apropriado**.**

2	50
5	11
10	15

2	5	10
50	11	15

31. Num contexto em que existem as declarações `double[][] notas` e `double[] pesos` e assumindo que:

- na linha `i` de `notas` estão as notas do aluno nº `i` nas diferentes componentes de avaliação de uma disciplina (notas parciais)
- os elementos de `pesos`, tantos quantos o número de colunas da matriz de notas, representam os pesos de cada componente na nota final (valores positivos que somam 1)

a) escreva um método que, dada uma matriz de notas e um vetor de pesos, devolve a média das notas finais;

b) escreva um método que, dada uma matriz de notas e um vetor de pesos, devolve um vetor com as notas finais dos alunos;

c) escreva um método que, dada uma matriz de notas e dois valores `inc` e `sup`, incrementa de `inc` valores todos os elementos da matriz menores que `sup`;

d) escreva um método que, dada uma matriz de notas e um vetor de pesos, devolve um vetor de `int` com duas posições: na primeira posição estará o número de ordem da componente de avaliação onde houve a melhor nota e na segunda posição estará o número de ordem do aluno que obteve essa melhor nota parcial;

e) escreva um método que, dada uma matriz de notas e um vector de pesos, devolve um vetor de `doubles` com duas posições: na primeira posição estará a média das notas finais dos alunos na disciplina e na segunda posição o desvio padrão;

f) para o caso particular de uma disciplina com 4 componentes de avaliação, em que a primeira componente vale 10% e as restantes valem todas o mesmo, indique a forma correta de criar e inicializar o vetor dos pesos.

32. Escreva um programa Java que crie uma matriz de notas de alunos e um vetor de pesos e os preencha com valores dados pelo utilizador. De seguida imprima um menu de opções que permita ao utilizador escolher as informações que deseja obter (estas incluem todas as informações possíveis de obter com os métodos do exercício anterior). Para cada opção, o programa deve invocar o método correspondente.

IV – Utilização de algumas classes da biblioteca Java

Revisões

33. Quais as principais diferenças entre as classes `String` e `StringBuilder`? A que pacote pertencem?
34. Para que serve a classe `Random`? Que métodos conhece nesta classe? A que pacote pretence?
35. Para que serve a classe `Scanner`? Que métodos conhece nesta classe? A que pacote pertence?
36. O que representa o objeto `System.in`? Como se cria e utiliza um objeto da classe `Scanner`?

Exercícios

Nos exercícios que se seguem, sempre que adequado construa métodos com bom potencial de reutilização. Para cada método definido deve escrever um cabeçalho incluindo uma descrição sucinta e geral do método e, se for caso disso, `@param`, `@requires` e `@return`.

37. Recorrendo à classe `Random` do pacote `java.util` escreva métodos para:

a) Gerar um inteiro aleatório positivo menor que um dado valor;

b) Gerar um valor aleatório no intervalo [`x`, `y`] onde `x` e `y` são inteiros.

38. Recorra à classe `Random` do pacote `java.util` para fazer um programa que gera aleatoriamente uma matrícula de automóvel portuguesa.

Sugestão: para gerar um caráter aleatório, prepare uma `String` com o conteúdo `"ABCDEFGHIJLMNOPQRSTUVXZ"`, e invoque o método `charAt` utilizando como argumento um número inteiro aleatório adequado.

39. Faça um programa que gera *passwords* aleatoriamente recorrendo à classe `Random` do pacote `java.util`. O programa define o número `n` de carateres que quer que a *password* tenha e imprime

uma password com `n` carateres de entre os seguintes: letras de A a Z, tanto maiúsculas como minúsculas, algarismos de 0 a 9 e os carateres especiais $, &, #, %.

Para isso construa uma função `static String password (int n, String possible)` que gera uma *password* dados o número `n` de carateres que a password deve ter e o conjunto `possible` de carateres possíveis para a *password*.

Faça as alterações necessárias para pedir ao utilizador o valor inteiro e a os carateres possíveis.

40. Escreva um programa que lê do teclado uma sequência de inteiros terminada por zero e que imprime o número de valores lidos e a sua média. O zero é usado como valor *sentinela*, significando o fim da introdução de valores. Como tal, não conta para o cálculo da média.

41. Programe um método `int lerValorNoIntervalo (int n, int m, Scanner canal)` que pede ao utilizador um valor inteiro entre `n` e `m`, lê-o através do canal dado e devolve-o. No caso de o utilizador inserir um valor fora do intervalo, o método deve dar uma mensagem de erro para o ecrã e voltar a pedir o valor.

42. Aumente o potencial de reutilização do método `lerValorNoIntervalo` recebendo como parâmetro também a mensagem de erro a escrever no *standard output*, no caso de o utilizador não inserir um valor no intervalo requerido.

43. Programe um método `int leInteiro (String msgErro, Scanner sc)` que lê um valor garantidamente inteiro através do canal dado e devolve-o. No caso de o utilizador inserir caracteres que não constituem um inteiro, o método deve escrever a mensagem de erro `msgErro` no ecrã e voltar a pedir o valor. Utilizando este método, faça as alterações necessárias ao método `lerValorNoIntervalo` para que se comporte de forma robusta nos casos em que o utilizador não insere valores inteiros.

44. Faça um programa completo que leia do teclado três valores inteiros entre 0 e 20 e imprima no ecrã a média arredondada às unidades. Utilize o método `lerValorNoIntervalo` para ler os três valores e o método `round` fornecido pela classe `Math`.

45. Considere que tem disponível uma classe `Dado` cujos objetos representam dados de 6 faces. Em cada momento, o dado encontra-se com uma das suas faces para cima. A classe disponibiliza:

- Construtor sem argumentos;
- método `void lancar ()` que "lança o dado ao ar";
- método `int obterValor ()` que retorna o valor da face que se encontra atualmente para cima; este método não pode ser chamado se anteriormente não tiver sido feito nenhum lançamento

a) Assumindo que a classe `Dado` está implementada corretamente, descreva o que acontece quando é executado o seguinte pedaço de código

```
Dado dado1 = new Dado();
dado1.lancar();
System.out.println(dado1.obterValor());
Dado dado2 = dado1;
dado2.lancar();
System.out.println(dado1.obterValor());
System.out.println(dado2.obterValor());
```

b) Escreva um programa que implementa um jogo de sorte que consiste no lançamento de dois dados. O jogador ganha se o resultado dos dados somar 7 e neste caso recebe o dobro do que apostou na jogada. O jogador perde no caso contrário, perdendo tudo o que apostou na jogada. O programa deve executar as seguintes tarefas:

1. lê do teclado o nome do jogador;
2. lê do teclado o valor que ele pagou inicialmente para jogar;
3. enquanto o jogador quiser continuar a jogar,
    i. lê o valor da aposta (que não pode ser superior ao valor disponível);
    ii. manda lançar os dados;
    iii. informa de resultado e atualiza valor disponível