Instituto Superior de Engenharia de Lisboa

PROGRAMAÇÃO

Relatório do Primeiro Trabalho Prático

Docente

Profª. Matilde Pato

Alunos

42502 Afonso Rodrigues
44225 Rúben Rosário
43552 Samuel Costa

Outubro 2016

Índice

Introdução

Este relatório, referente ao primeiro de três trabalhos práticos, foi elaborado no âmbito da unidade curricular de Programação e acompanha a realização de seis exercícios propostos. Estes exercícios constituíram uma aplicação de alguns aspetos introdutórios à linguagem Java que o presente relatório aprofunda e expande.

Ao longo do documento apresentam-se os principais objetivos metodológicos e práticos delineados pelos exercícios; a metodologia adotada; as várias etapas do raciocínio subjacente à resolução de cada problema; e, por fim, uma breve reflexão sobre o trabalho realizado em que se explicitam duas conclusões principais: uma referente ao modo como as ferramentas adotadas condicionam as soluções desenvolvidas e outra em que se avançam possíveis melhoramentos de robustez do código a contemplar no futuro.

Objetivos

O trabalho prático visa, em termos gerais, aplicar as ferramentas introduzidas em contexto de sala de aula e aprofundar as suas potencialidades de utilização, por meio de resolução de exercícios.

Definem-se os seguintes objetivos:

- do ponto de vista metodológico: determinar possíveis aproximações aos exercícios a partir de pseudocódigo e algoritmo e escolher a abordagem mais adequada a cada problema;
- do ponto de vista prático, empregar e dominar os seguintes instrumentos da linguagem java: (i) respeitar a sintaxe do JAVA; (ii) conhecer características dos tipos primitivos de modo a selecionar o que mais se adequa em cada passo; (iv) criar e manipular variáveis de tipo primitivo ou *String*; (v) conceber estruturas de controlo do tipo *if*, *else if* e *switch* e, por último, (vi) interagir com o programa recorrendo a input e output na consola.

Métodos e Recursos Utilizados

Para a realização deste trabalho foi utilizada a linguagem *Java™ Versão 8 Update 111.*

Todos os elementos da linguagem necessários para a resolução dos problemas propostos foram adquiridos nas aulas da unidade curricular Programação (PG). Foram estudados todos os tipos primitivos e as suas características, no entanto, para a realização deste trabalho foram apenas aplicados os seguintes: (i) *int* para guardar valores inteiros, (ii) *double* valores reais, (iii) *char* para um único caracter e (iv) *boolean* para dados binários (um único bit de informação, podendo representar os valores *true* ou *false*).

No que diz respeito a instruções de entrada e saída de dados foram usados os métodos *next()*, *nextInt()* e nextDouble() sobre objetos da classe *Scanner* para guardar valores introduzidos através do teclado (quer sejam *strings*, reais ou inteiros), e *System.out.print()*, *System.out.println()* e *System.out.printf()* para saída de dados na consola.

Foram ainda aplicados os métodos *charAt()* e *length() a variáveis do tipo String.*

Em relação a estruturas de controle de fluxo foram usados os tipos *if, if else* e *switch.*

Procurou-se que as soluções cumprissem escrupulosamente o enunciado proposto, tanto no que diz respeito às ferramentas utilizadas como ao output dos programas.

Exercícios

A presente secção deste relatório está dividida em seis partes. Cada uma delas corresponde a um dos exercícios propostos. Apresenta-se a resolução na forma de Pseudocódigo, explicando-se cada passo com o detalhe adequado. Sempre que se revele importante para o entendimento de um passo específico, são mostrados trechos de código java no contexto desse passo.

É de salientar que em alguns dos exercícios foram programadas instruções adicionais que melhoram a robustez dos programas (por exemplo, foram previstos diferentes formatos de input dados pelo utilizador). O critério utilizado para avaliar a pertinência desta opção é explicado nas Conclusões.

Exercício 1 – *Kbyte*

Para este exercício foi proposto a criação de um programa em que o utilizador introduza um valor real em *kilobytes*, e o programa apresenta este valor em *bytes* arredondado para o inteiro mais próximo. É importante realçar que está convencionado que 1 *Kilobytes* (KB) é igual a 1024 *bytes*.

De seguida é apresentado o pseudocódigo desenvolvido para a resolução do problema.

Pseudocódigo:

1. Perguntar ao utilizador para introduzir um valor de *kilobytes*;
   - utilizando instruções de saída de dados (*System.out.print("Valor em kBytes? ");*).
2. *Input*: Ler o valor introduzido através do teclado e guardar numa variável do tipo *Double*;
   - utilizando a classe *Scanner* do *package java.util.Scanner* (instrução: *double kvalor = keyboard.nextDouble();*).
3. Converter o valor de *kilobytes* para bytes (*kvalor * 1024 = valor_ bytes*);
4. Realizar o arredondamento do valor em *bytes* para o inteiro mais próximo, avaliando o resto da divisão do valor por um;
   - se o resto da divisão (*valor_bytes%1*) for maior ou igual a 0.5, é adicionado 1 à parte inteira do valor.
   - se o resto da divisão (*valor_bytes%1*) for menor que 0.5, é guardado apenas considerada a parte inteira do valor.

5. *Output*: mostrar o valor introduzido em *kilobytes* e o seu equivalente em *bytes* (exemplo: *512.75 kBytes = 525056 Bytes*):
    - instrução: *System.out.println (kvalor + "kByte = " + (int)valor + 1 + " Byte.");*.

Exercício 2 – *Triang*

Para este exercício foi proposto a criação de um programa que leia o comprimento dos lados de um triângulo introduzido pelo utilizador e o classifique em relação aos lados: equilátero, isósceles ou escaleno. É assumido à partida que o utilizador introduz valores inteiros sendo cada um deles menor que a soma dos outros dois.

Pseudocódigo:

1. *Input*: Solicitar ao utilizador a introdução através do teclado os valores dos comprimentos (valores inteiros) dos lados do triângulo;
2. Ler valores introduzidos pelo utilizador e guardá-los em três variáveis distintas do tipo inteiro *(int)*;
3. Comparar os lados dos triângulos através de uma variável do tipo *boolean*: *(boolean equilatero = a1==a2 & a1==a3; boolean isosceles = a1==a2 || a1==a3 || a2==a3;)* ;
4. Avaliar as variáveis *boolean equilatero* e *isosceles* (quando ambas retornam *false* o triângulo é classificado como escaleno);
5. *Output*: mostrar a classificação do triângulo através de classe *println* (exemplo: *System.out.println ("Triangulo equilátero")*).

Exercício 3 – *DigiChar*

Para este exercício foi proposto o programa ler um valor inteiro e, caso o caractere seja um dígito, apresentar o caractere que corresponde à soma do valor inteiro e o código *UNICODE* do dígito. Se o utilizador introduzir um caractere que não seja um dígito, apresentar um output com a mensagem a mencionar esse facto.

Pseudocódigo:

1. Pedir ao utilizador para introduzir um dígito e um valor (exemplo: “Digito e valor? ”);
2. Ler o primeiro caractere introduzido através do teclado e guardar numa variável do tipo char, usando a classe *charAt()* para ler o caractere introduzido (exemplo: *char c = keyboard.next().charAt(0);*);
3. Ler o segundo valor inteiro através do teclado e guardar numa variável do tipo *int* (exemplo: *int n = keyboard.nextInt();*);
4. Verificar se o caracter é um dígito, por meio do código *UNICODE*;
   - Para ser um dígito tem o seu valor *UNICODE* tem de estar compreendido no intervalo [48,57].
   - Caso o caractere introduzido não seja um dígito, é devolvido ao utilizador uma mensagem de *output* "caractere não é um dígito", e o programa termina.
5. Se o caracter introduzido for um dígito, é realizada a soma do respetivo *UNICODE* com o valor inteiro introduzido, e devolvido o caractere resultante dessa soma através do *Output= (“caracter (unicode) + valor_inteiro = caracter_soma (soma_unicode)”)* (ver Listagem 1).

```
      System.out.println("'" + (char)c + "'" + "(" + (int)(c) + ")" + "+" + n + "=" +
"'" + (char)(c+n) + "'" +"(" + (int)(c+n) +")");
```

Listagem 1 - Excerto de Output do programa DigiChar, para o cálculo do caractere relativo à soma do UNICODE do Digito com o valor, e o respetivo output.

Exercício 4 – *Greeting*

Para este exercício foi proposto a criação de um programa que pergunte a hora e o nome do utilizador e apresente uma mensagem de cumprimento (Bom dia, Boa tarde ou Boa noite), consoante a hora introduzida. Caso o utilizador não introduza a palavra auto, ou outra qualquer, o programa terá de utilizar a hora atual através da expressão *LocalTime.now().getHour()* usando a classe *LocalTime* do package *java.time.LocalTime*.

Pseudocódigo:

1. Solicitar ao utilizador a introdução através do teclado as horas e o seu nome;
2. *Input*: Ler valores introduzidos pelo utilizador e guardá-los em duas variáveis distintas do tipo *string*;
3. Verificar se a hora introduzida pelo utilizador é válida, isto é, está compreendida no domínio [0,23];
4. Ler os caracteres da *string* através da classe *charAt()* e guardá-los numa variável do tipo *int* ;
5. Caso o utilizador introduza a palavra "*auto*", uma hora não válida, ou por erro de introdução, outros caracteres quaisquer:
   5.1. Definir a hora através da classe *LocalTime.now().getHour()* (necessário importar o *package java.time.LocalTime*);
6. Definir o tipo de cumprimento através de uma estrutura de controle *if* , verificando em que intervalo de horas o utilizador se encontra:
   - [20h 23h] → Cumprimento = Boa noite;
   - [0h 5h[→ Cumprimento = Boa noite;
   - [5h 12h[→ Cumprimento = Bom dia;
   - [12h 20h] → Cumprimento = Boa tarde;
7. *Output*: mostrar a mensagem "cumprimento nome" (exemplo: "Bom dia Rita").

```
          if (horas>=20 & horas<=23 || horas>=0 & horas<5) { // se a variável horas
tiver compreendida entre os intervalos [20h 23h] e [0h 5h]
            System.out.println("Boa noite " + nome + "."); // Output do cumprimento e
nome do utilizador
          } else if (horas>=12 & horas<=20) {
            System.out.println("Boa tarde " + nome + "."); // Output do cumprimento e
nome do utilizador
          } else if (horas>=5 & horas<12) {
            System.out.println("Bom dia " + nome + "."); // Output do cumprimento e
nome do utilizador
          } else { // se hora estiver fora do dominio [0 23], devolve a seguinte
mensagem
            System.out.println("A hora introduzida está fora do dominio [0 23].");
          }
```

Listagem 2 - Excerto do programa Greeting, onde é analisado o tipo de cumprimento em função das horas obtidas.

Exercício 5 – *FichasPG*

Para este exercício foi proposto a criação de um programa que leia a nota de 3 ou 4 fichas já realizadas em Programação (PG), apresentar as notas das 3 melhores por ordem decrescente e a respetiva média. Caso tenham sido realizadas apenas 3 fichas, o programa teria de indicar a nota mínima da 4ª ficha, por forma o aluno não reprovar ou melhorar a média. O programa deverá ainda retornar uma mensagem a dizer "Reprovado", caso já não seja possível ter média superior ou igual a 8 valores.

Pseudocódigo:

1. Pedir ao utilizador para introduzir o número de fichas realizadas (exemplo: *"Fichas realizadas? "*);
2. Verificar se o número introduzido é igual a 3 ou 4:
   2.1. Se não for verificada essa condição, mostrar uma mensagem ao utilizador dizendo que o número de fichas é inválido, terminando o programa.
3. Pedir ao utilizador que insira tantos valores de notas das fichas quanto o numero de fichas realizadas;
4. Ler e guardar esses valores em variáveis do tipo *int*;
5. Verificar se todos os valores introduzidos estão no intervalo [0,20] (através de uma variável *boolean*). Caso contrário exibir uma mensagem que diga "Valores introduzidos são inválidos";
6. Comparar os valores introduzidos recorrendo a estruturas de controlo de fluxo (*if else*) e seriar os 3 melhores por ordem decrescente (exemplo: *f1>=f2>=f3>=f4*). Guardar os valores em variáveis do tipo inteiro (exemplo: *p=f1; s=f2;t=f4*);
7. Caso *número_fichas=3*, Calcular a média (*(soma_valor_tres_fichas)/3*), guardá-la numa variável real;
8. Output: mostrar ao utilizador o resultado da média com 6 casas decimais, apresentando discriminadamente as três notas das fichas mais altas, por ordem decrescente.
   Para formatar o output usar o método *printf* (ver Listagem 3).
9. Averiguar situação do aluno, considerando os casos:
   9.1. O aluno tem média superior a 8, logo não reprova.
      9.1.1. Caso condição se verifique, calcular a nota mínima da 4ª ficha necessária para melhorar a média. Output: *"ficha4>terceiro_melhor_valor"*;
   9.2. O aluno tem média inferior a 8.

9.2.1. O aluno está reprovado (*somatório das 3 fichas <= 4*). Output: *"Reprovado"*. Termina o programa.

9.2.2. O aluno não está reprovado (*5 < somatório das 3 fichas <= 24*). Calcula a nota mínima necessária da 4ª ficha para não reprovar (*24 - segunda_melhor_ficha - primeira_melhor_ficha*) e realiza o *output* (*System.out.println("ficha4 > " + (24 - s - p) + " para não reprovar.");*). (exemplo de *output*: *"ficha4> 6 para não reprovar"*). Termina o programa.

9.2.3. O aluno não está reprovado, mas o somatório das 3 fichas é igual a 5 valores: O output para este caso é *"ficha4 = 20 para não chumbar"*. Termina o programa.

10. Caso *numero_fichas=4*: Elabora uma cadeia de controlo de fluxo com vários construtores *else if* que cubram todas as 16(4!) combinações possíveis de quatro elementos;
11. Atribuir às variáveis *p, s, t* os valores das fichas correspondentes ao *primeiro_melhor_resultado*, *segundo_melhor_resultado* e *terceiro_melhor* resultado, respetivamente, em cada um dos 16 casos;
12. Calcular a média dos 3 melhores valores, realizar o *output* formatado de acordo com ponto [8], descrito na Listagem 3.

```
System.out.printf("Média=(%d+%d+%d)/%d = ", p, s, t, n);
System.out.printf("%.6f", med); System.out.println();
```

Listagem 3- Excerto do programa FichasPG - Instruções para formatação da saída de dados na consola.

Exercício 6 – *YearsDays*

Para este exercício foi proposto a criação de um programa que leia uma data introduzida pelo utilizador (Dia Mês Ano), e calcular os dias que passaram desde o inicio desse ano até à respetiva data introduzida.

Pseudocódigo:

1. Solicitar ao utilizador a introdução através do teclado uma data com o formato *"Dia Mês Ano"* (exemplo: 22/11/1999). *Output: "Dia Mês Ano? "*;
2. Ler valores introduzidos pelo utilizador e guardá-los em três variáveis distintas do tipo *int*;
3. Verificar se o ano é bissexto (fevereiro com 29 dias) através de uma variável *boolean*. O ano é bissexto se for múltiplo de 4, exceto se for múltiplo de 100 e não de 400 (ver Listagem 4);

4. Caso seja um ano bissexto atribuir 29 dias ao mês de fereveiro, caso contrário 28 dias (instrução feita através de um operador ternário – ver Listagem 4);
5. Verificar se o utilizador introduziu uma data válida (exemplos de datas inválidas: 32/1/2016 , 30/2/2016, 31/9/2016, 29/2/2015, 0/4/1999). Esta instrução é realizada através das condições:
   - Mês introduzido fora do domínio [0,12]: *boolean mesForaDos12 = mes < 1 || mes > 12*;
   - Introdução de um dia superior ao número de dias de um mês de 30 dias: *boolean diaForaMes30 = dia > 30 && (mes == 4 || mes == 6 || mes == 9 || mes == 11);*
   - Introdução de um dia superior ao número de dias de um mês de 31 dias: *boolean diaForaMes31 = dia > 31 && (mes == 1 || mes == 3 || mes == 5 || mes == 7 || mes == 8 || mes == 10 || mes == 12);*
   - Introdução de um dia inferior a 1: *boolean diaInf_0 = dia < 1;*
6. Caso a data introduzida seja válida, é criada uma estrutura de control de fluxo *switch*, que dependente do mês que foi introduzido, atribui às variáveis *mesesAnteriores31* e *mesesAnteriores30*, um valor inteiro que corresponde ao número de meses com 31 e 30 dias, respetivamente, desde o inicio do ano até à data de *input*;
7. Calcula o número de dias que passaram desde o inicio do ano até à data em de *input* através de uma formula matemática (ver Listagem 5);
8. *Output*: mostrar número de dias que passaram (exemplo: *"282 dias desde 1/1/2016"*) (instrução java: *System.out.println(nDias + " dias desde " + "1" + "/" + "1" + "/" + ano);*)

```
boolean  anoBissexto = ano%4==0 && (ano%100!=0 || ano%400==0);
int fevereiro = (anoBissexto == true)? 29 : 28;
```

Listagem 4 - Excerto do programa YearDays - verificador de ano bissexto e atribuição do número de dias ao mês de fevereiro.

```
if (mes > 2) {
      nDias = mesesAnteriores31 * 31 + mesesAnteriores30 * 30 + fevereiro + dia - 1 ;
} else {
      nDias = mesesAnteriores31 * 31 + dia - 1;
}
```

Listagem 5 - Excerto do programa YearsDays, para contabilizar os dias que passaram desde o primeiro dia do ano até à data de input do utilizador.

Conclusões

Com a realização deste trabalho (exercícios práticos e relatório) foi possível aplicar as ferramentas introduzidas nas aulas de Programação, consolidando e aprofundando as suas potencialidades de utilização, assim como aprender a melhor estruturar o pensamento lógico para a resolução de problemas através da programação.

Destacam-se duas principais conclusões:

(1)

Todos os programas desenvolvidos foram testados e o seu output apresenta conformidade com os exemplos fornecidos no enunciado. No entanto, existem diferentes maneiras de abordar o mesmo problema, sendo que umas são mais eficazes do que outras, ainda que aparentemente o resultado final seja o mesmo. O presente relatório permitiu avaliar criticamente a conveniência das estratégias adotadas em função dos resultados esperados.

Apesar de não ser permitida a utilização de ciclos repetitivos, *arrays*, métodos da class *Math* ou qualquer outra classe da API do JAVA (com exceção do caso do exercício 4, onde foi permitido o uso do *package java.time.LocalTime*), foi possível resolver os problemas recorrendo apenas a decisão binária, decisão múltipla, estruturas de controle de fluxo *if*, *else if* e *switch*.

Em relação às estruturas de controle de fluxo, pode-se concluir que, quando é necessário fazer muitas comparações com um valor de uma única variável, o *switch* representa uma melhor opção em detrimento do *if else,* tornando o código mais estruturado e facilitando a leitura. Quando é necessário comparar mais do que uma variável ou valor, com poucos casos, a estrutura de controle de fluxo *if else* pode tornar-se uma opção viável.

(2)

A elaboração deste relatório constituiu também um ponto de partida para pensar possíveis soluções de melhoramento dos programas no âmbito da robustez e fiabilidade.

Durante o desenvolvimento dos programas foram encontradas algumas situações que poderiam influenciar os resultados obtidos, nomeadamente a forma ou conteúdo dos dados fornecidos pelo utilizador. Isto é, caso o utilizador introduza dados com um determinado formato que não tenha sido previsto pelo programador, o programa pode devolver um resultado incorreto, ou mesmo não conseguir processar corretamente os dados. Algumas destas situações foram pensadas e implementadas nos exercícios, como por exemplo no programa *Greeting*, onde foi considerado que o utilizador poderia introduzir um conjunto de caracteres quaisquer,

e em diferentes formatos, como resposta à questão *"Hora (auto)?"*. Este tipo de apêndices torna os programas mais robustos para uma eventual falha, ou mau uso por parte do utilizador. Posto isto, é necessário ter em mente que os programas desenvolvidos neste trabalho poderiam ser significativamente aperfeiçoados, bem como ampliados em termos de funcionalidade. Por exemplo, no exercício *5.FichasPG,* poder-se-ia ter acrescentado uma opção de exibir para o utilizador a mensagem *"Reprovado"* se a média das fichas fosse menor do que 8 e tivessem sido realizadas 4 fichas, e 6.*YearDays*, poderia acrescentar-se a opção de o utilizador introduzir duas datas distintas e o programa calcular os dias que passaram entre as mesmas.

Referências

[1] Savitch W. *Java: An Introduction to Problem Solving and Programming*. 7st Ed ed: Pearson; 2015.

[2] Java Documentation - Java Platform, Standard Edition (Java SE) 8. https://docs.oracle.com/javase/8/. Accessed 29-10-2016.