Programação I

Aula 2

Professor: Diogo Passos Ranghetti

Sumário

Paradigma

No paradigma de programação orientado a objetos, a construção de um programa para implementar um determinado sistema baseia-se em uma correspondência natural e intuitiva entre esse sistema e a simulação do comportamento do mesmo: a cada entidade do sistema corresponde, durante a execução do programa, um objeto.

Em outras palavras, em um programa em linguagem orientada a objetos, todos os dados, ferramentas usadas para manipular dados, ou qualquer informação usada no programa será representada como um objeto (NOTA: existem algumas exceções, que examinaremos mais adiante). Note que objetos do mundo real em geral possuem duas características: estado e comportamento.

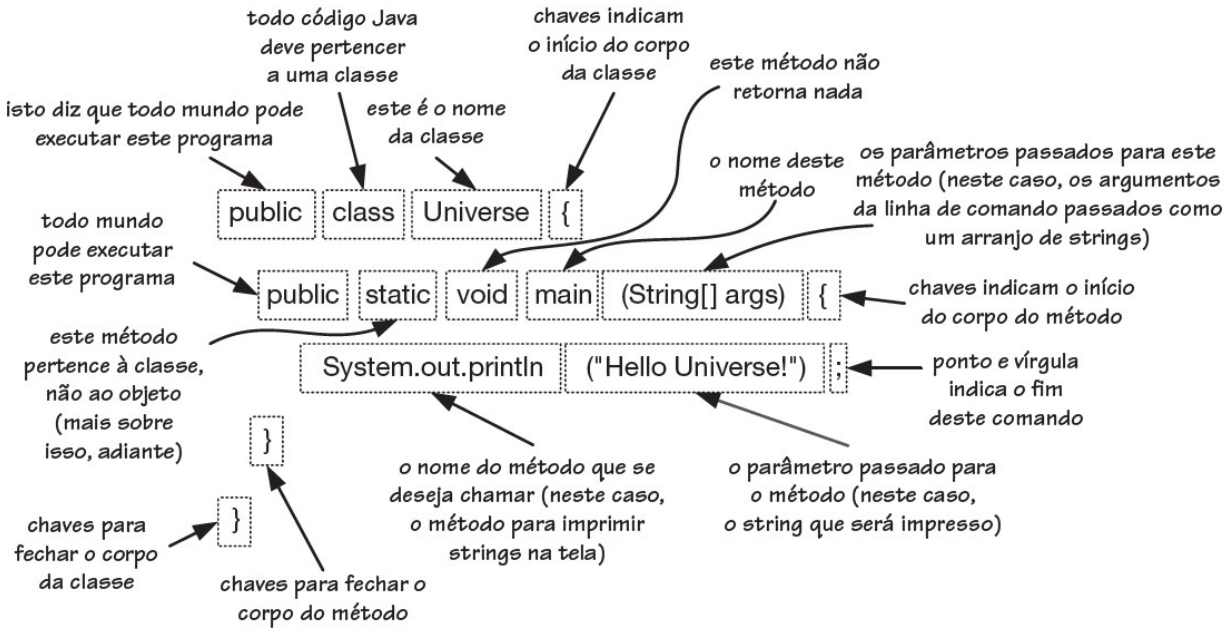
Por exemplo, uma bicicleta pode estar parada, ou em movimento, ter uma determinada cor, marca, etc. Ela pode “executar” ações como acelerar e desacelerar, mudar de direção etc. Objetos em Java não são muito diferentes – são componentes da execução de programas que modelam objetos do mundo real, no sentido de que também possuem um estado e comportamento.

Os principais “atores” em um programa Java são os objetos. Os objetos armazenam dados e fornecem os métodos para acessar e modificar esses dados. Ou seja, mantém informação sobre seu estado por meio de variáveis e implementa seu comportamento (ou as ações que o objeto é capaz de executar) por meio de métodos. Todo objeto é instância de uma classe que define o tipo do objeto, bem como os tipos de operações que executa.

Um objeto é uma combinação específica de dados e dos métodos capazes de processar e comunicar esses dados. As Classes definem os tipos dos objetos por isso, objetos também são chamados de instâncias da Classe que os define e usam o nome da classe como seu tipo.

Estrutura básica

No exemplo abaixo podemos ver a estrutura básica de um programa escrito em Java, note o uso do método “main” esse é o principal método, pois o programa inicia a partir dele.



Variáveis

Uma variável é um objeto normalmente localizado na memória utilizado para representar valores, quando declaramos uma variável estamos associando seu nome (identificador) ao local da memória onde está armazenada sua informação, as variáveis em Java podem ser do tipo primitivo ou objeto:

Variáveis primitivas: podem ser do tipo byte, short, int, long, float, double, char ou boolean.

Variáveis de referência: usada para referenciar um objeto. Quando usamos uma variável de referência definimos qual o tipo do objeto ou um subtipo do tipo do objeto (veremos isso mais para frente).

Quando declaramos uma variável primitiva, estamos associando esta a um espaço na memória que vai guardar o seu valor. No caso da variável de referência, podemos tê-la apontando para algum lugar vazio (null) ou para algum espaço da memória que guarda os dados de um objeto.

As variáveis primitivas e de referência são guardadas em locais diferentes da memória. Variáveis primitivas ficam em um local chamado **stack** e as variáveis de referência ficam em um local chamado **heap**.

Dados de objetos Java são armazenados em variáveis de instância (também chamadas de campos). Por essa razão, se um objeto de uma classe deve armazenar dados, então sua classe deve especificar variáveis de instância para esse fim. As variáveis de instância podem ser de tipos básicos (tais como inteiros, números de ponto flutuante ou booleanos) ou podem se referir a objetos de outras classes.

Tipos Primitivos

A linguagem Java não é totalmente Orientada a Objetos, e isto se deve principalmente aos atributos do tipo primitivo, pois são tipos de dados que não representam classes, mas sim valores básicos.

Os tipos primitivos, assim como em várias outras linguagens tais como o C, existem para representar os tipos mais simples de dados, sendo eles dados numérico, booleano e caractere. Os tipos primitivos da linguagem Java são:

		Valores possíveis				
Tipos	***Primitivo***	***Menor***	***Maior***	***Valor Padrão***	***Tamanho***	***Exemplo***
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;
	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d; ou double ex6 = 10.20;
Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

Uma variável declarada como tendo um desses tipos simplesmente armazena um valor desse tipo, em vez de uma referência para um objeto. Constantes inteiras, tais como 14 ou 195, são do tipo ***int***, a menos que seguidas de imediato por um “L”ou “l”, sendo, neste caso, do tipo ***long***. Constantes de ponto flutuante, como 3.1415 ou 2.158e5, são do tipo ***double***, a menos que seguidas de imediato por um “F” ou um “f”, sendo, neste caso, do tipo ***float***.

Palavras Reservadas

Java possui algumas palavras chaves que são da linguagem e não podem ser usados como nome de variáveis, métodos e classes.

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

Operadores

Aritméticos:

+	adição	-	subtracção
*	multiplicação	/	divisão
%	módulo	++	incremento
--	decremento		

Lógicos:

!	NOT lógico
&&	AND lógico
||	OR lógico

Bits:

~	NOT binário	&	AND binário
|	OR binário	^	XOR binário
<<	shift binário esquerda	>>	shift binário direita

Relacionais ou de comparação:

==	igualdade	!=	diferente
<	menor que	<=	menor ou igual que
>	maior que	>=	maior ou igual que

Os comandos *if* e *switch*

Em Java, comandos condicionais funcionam da mesma forma que em outras linguagens. Eles fornecem a maneira de tomar uma decisão e então executar um ou mais blocos de comandos diferentes baseados no resultado da decisão.

O comando *if*

A sintaxe básica do comando if é a que segue:

```
if (expr_booleana)
    comando_se_verdade
else
    comando_se_falso
```

Onde expr_booleana é uma expressão booleana e comando_se_verdade e comando_se_falso podem ser um comando simples ou um bloco de comandos entre chaves (“{” e “}”). Observa-se que, diferentemente de outras linguagens de programação, os valores testados por um comando *if* devem ser uma expressão booleana. Particularmente, não são uma expressão inteira. Por outro lado, como em outras linguagens similares, a cláusula *else* (e seus comandos associados) são opcionais. Existe também uma forma de agrupar um conjunto de testes booleanos como segue:

```
if (primeira_expressão_booleana)
    comando_se_verdade
else if (segunda_expressão_booleana)
    segundo_comando_se_verdade
else
    comando_se_falso
```

Se a primeira expressão booleana for falsa, então a segunda expressão booleana será testada e assim por diante. Um comando *if* pode ter qualquer quantidade de cláusulas *else*. Por segurança, quando se define um comando *if* complicado, usam-se chaves para agrupar o corpo do comando.

Por exemplo, a estrutura a seguir está correta:

```
public class IfElseTest {
        public static void main(String[] args) {
                int testscore = 76;
                char grade ;

                if ( testscore >= 90) {
                        grade = 'A';
                } elseif ( testscore >= 80) {
                        grade = 'B';
                } elseif ( testscore >= 70) {
                        grade = 'C';
                } elseif ( testscore >= 60) {
                        grade = 'D';
                } else {
                        grade = 'F';
                }

                System.out.println(" Grade = " + grade );
   }
}
```

Comando *switch*

Java oferece o comando *switch* para controle de fluxo multivalorado, o que é especialmente útil com tipos enumerados.

```
switch (d) {
        case MON:
                System.out.println("This is tough.");
                break;
        case TUE:
                System.out.println("This is getting better.");
                break;
        case WED:
                System.out.println("Half way there.");
                break;
        case THU:
                System.out.println("I can see the light.");
                break;
        case FRI:
                System.out.println("Now we are talking.");
                break;
        default:
                System.out.println("Day off!");
                break;
}
```

O comando switch avalia uma expressão inteira ou enumeração e faz com que o fluxo de controle desvie para o ponto marcado com o valor dessa expressão. Se não existir um ponto com tal marca, então o fluxo é desviado para o ponto marcado com "*default*".

Entretanto, este é o único desvio explícito que o comando *switch* executa, e, a seguir, o controle "cai" através das cláusulas case se o código dessas cláusulas não for terminado por uma instrução *break* (que faz o fluxo de controle desviar para a próxima linha depois do comando switch).

Entrada de dados via console

Em Java temos uma classe chamada java.util.Scanner que neste momento utilizaremos para receber entradas do usuário via console, mas esta classe também é pode ser utilizada para outros fins, tais como leitura de arquivos por exemplo.

No exemplo abaixo vamos utilizar a classe Scanner para pedir que o usuário digite sua idade, depois iremos imprimir qual foi o número lido:

```
import java.util.Scanner;

public class ExemploScanner {
  public static void main(String[] args) {
      // TODO code application logic here
      System.out.println("teste");
      Scanner s = new Scanner(System.in);

      System.out.println("Digite seu nome: ");
      String nome = s.nextLine();

      System.out.println("\nDigite sua Altura: ");
      double altura = s.nextDouble();

      System.out.println("\nDigite sua Idade: ");
      int idade = s.nextInt();

      System.out.println(nome + " tem " + altura + " de altura, e " + idade + " anos.");
   }
}
```

Para ter saída de dados no console podemos usar o seguinte comando:

```
System.out.println("Minha mensagem no console");
```

Note que no exemplo temos um exemplo de concatenação de valores utilizando o comando “+” :

```
System.out.println(nome + " tem " + altura + " de altura, e " + idade + " anos.");
```

Para entrada de valores de tipos diferente note que a leitura usa métodos diferentes:

```
double altura = s.nextDouble();
int idade = s.nextInt();
```

Referências bibliográficas

DEITEL, H. M.; DEITEL, P. J. Java como Programar. 8ª.ed. São Paulo: Pearson Brasil, 2010.

FIGUEIREDO, LUCILIA C. DE; MOTTA, CARLOS E. H. DE S. Cadernos de Informatica: CURSO DE PROGRAMAÇÃO EM JAVA.

http://www.universidadejava.com.br

http://srvd.grupoa.com.br/uploads/imagensExtra/legado/G/GOODRICH_Michael_T/Estruturas_Dados_Algoritmos_Java_5ed/Lib/Cap_01.pdf