Dia 29 – Projeto de Revisão

Variáveis

Na linguagem Java, declaramos uma variável informando o tipo de dados que ela poderá receber e seu nome.

Exemplos de declaração de variáveis:

int numero;

String nome;

Neste código informamos que a variável numero é do tipo inteiro e, por isso, só poderá receber valores desse tipo. O mesmo comportamento é válido para a variável nome do tipo String.

Nota: Como a linguagem Java é fortemente tipada, a declaração do tipo é obrigatória.

O tipo de dado pode ser qualquer um dos tipos primitivos (como int, float, boolean), assim como qualquer outra classe/interface, seja ela nativa do Java (como String, ArrayList) ou criada por você ou por terceiros (como Produto, ProdutoDAO).

Nomeação de variáveis

A nomeação de variáveis precisa ser feita respeitando algumas regras e convenções:

Pode conter letras, números e o caractere sublinhado (_), mas não pode começar com um número;

Devem ser declaradas em minúsculo. Caso sejam nomes compostos, a primeira letra de todas as palavras, menos da primeira, deve ser maiúscula (Camel Case);

Java é uma linguagem case sensitive. Assim, numeroUm é diferente de numeroum.

Exemplos de declaração de variáveis:

Estrutura Sequencial

ESTRUTURA SEQUENCIAL

Existem três estruturas básicas em programação:

Estrutura sequencial;

Estrutura Condicional;

Estrutura Repetitiva.

Dia 29 – Projeto de Revisão

Vamos estudar aqui a sequencial.

No momento certo vamos estudar as outras.

Como o próprio nome sugere, estrutura sequencial é um conjunto de intruções no qual cada instrução será executada em sequencia.

Obviamente, esta sequencia obedece a uma lógica de programação.

Construtores:

Construtores

Também conhecidos pelo inglês constructors, os construtores são os responsáveis por criar o objeto em memória, ou seja, instanciar a classe que foi definida. Eles são obrigatórios e são declarados conforme a Listagem 1.

Nota:

Em Java apenas as Interfaces não possuem construtores.

Listagem 1. Declaração de Construtores

```
public class Carro{

/* CONSTRUTOR DA CLASSE Carro */
public Carro(){
//Faça o que desejar na construção do objeto
}

}
```

O construtor sempre tem a seguinte assinatura:

modificadores de acesso (public nesse caso) + nome da classe (Carro nesse caso) + parâmetros (nenhum definido

neste caso). O construtor pode ter níveis como: public, private ou protected.

Porém, por que alguém colocaria um construtor private, para que assim ninguém pudesse instanciar essa classe? Um dos motivos é a aplicação do padrão de projeto Singleton, que

controla se um objeto já foi ou não criado. Para que isso ocorra ele não pode deixar que ninguém chame diretamente o construtor da classe.

Para criar um objeto da classe Carro simplesmente usamos a palavra reservada “new” e o nosso construtor é chamado, como mostra o exemplo da Listagem 2.

Array:

Matrizes Java

Os arrays são usados para armazenar vários valores em uma única variável, em vez de declarar variáveis separadas para cada valor.

Para declarar uma matriz, defina o tipo de variável com suportes quadrados:

String[] cars;

Nós agora declaramos uma variável que contém uma matriz de cordas. Para inserir valores a ele, podemos usar uma matriz literal - colocar os valores em uma lista separada por círias, dentro de chaves:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

Para criar uma matriz de inteiros, você pode escrever:

int[] myNum = {10, 20, 30, 40};

Acesse os elementos de uma matriz

Você acessa um elemento de matriz referindo-se ao número do índice.

Esta instrução acessa o valor do primeiro elemento em carros:

Exemplo

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars[0]);

// Outputs Volvo

Dia 29 – Projeto de Revisão

Nota: Os índices de matriz começam com 0: [0] é o primeiro elemento. [1] é o segundo elemento, etc.

Alterar um elemento de matriz

Para alterar o valor de um elemento específico, consulte o número do índice:

Exemplo

```
cars[0] = "Opel";
```

Exemplo

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

cars[0] = "Opel";

System.out.println(cars[0]);

// Now outputs Opel instead of Volvo
```

Comprimento da matriz

Para descobrir quantos elementos uma matriz tem, use a propriedade:length

Exemplo

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars.length);

// Outputs 4
```

Loop através de uma matriz

Você pode fazer loop através dos elementos de matriz com o loop e usar a propriedade para especificar quantas vezes o loop deve ser executado.forlength

O exemplo a seguir produz todos os elementos da matriz de carros:

Exemplo

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (int i = 0; i < cars.length; i++) {

  System.out.println(cars[i]);
```

}

Loop Through an Array with For-Each

Há também um loop "for-each", que é usado exclusivamente para loop através de elementos em matrizes:

Sintaxe

```
for (type variable : arrayname) {
  ...
}
```

O exemplo a seguir produz todos os elementos da matriz de carros, usando um loop "para cada" :

Exemplo

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

O exemplo acima pode ser lido assim: para cada elemento (chamado i - como em index) em carros, imprima o valor de i.String

Se você comparar o loop e o loop para cada loop, verá que o método para cada um é mais fácil de escrever, ele não requer um contador (usando a propriedade de comprimento) e é mais legível.for

Matrizes Multidimensionais

Uma matriz multidimensional é uma matriz que contém uma ou mais matrizes.

Para criar uma matriz bidimensional, adicione cada matriz dentro de seu próprio conjunto de chaves:

Exemplo

int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };

myNumbers é agora uma matriz com duas matrizes como seus elementos.

Para acessar os elementos da matriz myNumbers, especifique dois índices: um para a matriz e outro para o elemento dentro dessa matriz. Este exemplo acessa o terceiro elemento (2) na segunda matriz (1) dos meus Números:

Exemplo

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
int x = myNumbers[1][2];
System.out.println(x); // Outputs 7
```

Também podemos usar um outro interior para obter os elementos de uma matriz bidimensional (ainda temos que apontar para os dois índices):for loopfor loop

Exemplo

```
public class MyClass {
  public static void main(String[] args) {
    int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
    for (int i = 0; i < myNumbers.length; ++i) {
      for(int j = 0; j < myNumbers[i].length; ++j) {
        System.out.println(myNumbers[i][j]);
      }
    }
  }
}
```

Video

https://www.youtube.com/watch?v=CtM7o2rsTic

Exercicio

1] Quais são as verdadeiras sobre uma classe aninhada estática? (Escolha todas as opções aplicáveis.)

A. Você deve ter uma referência a uma instância da classe envolvente para instanciá-la

B. Ele não tem acesso a membros não estáticos da classe envolvente

C. Suas variáveis e métodos devem ser estáticos

D. Se a classe externa for chamada MyOuter, e a classe aninhada for chamada MyInner, pode ser

instanciado usando new MyOuter.MyInner ();

E. Deve estender a classe envolvente

2) class Boo {

Boo (string s) {}

Vaia() { }

}

class Bar extends Boo {

Barra() { }

Barra (String s) {super (s);}

void zoo () {

// insira o código aqui

}

}

2) Quais criam uma classe interna anônima de dentro da barra de classe? (Escolha todas as opções aplicáveis.)

A. Boo f = novo Boo (24) {};

B. Boo f = nova barra () {};

692 Capítulo 8: Classes internas

C. Boo f = new Boo () {String s; };

D. Barra f = novo Boo (String s) {};

E. Boo f = new Boo.Bar (String s) {};

3) Given:

1. public class HorseTest {

2. public static void main(String[] args) {

3. class Horse {

4. public String name;

5. public Horse(String s) {

6. name = s;

7. }

8. }

9. Object obj = new Horse("Zippo");

10. System.out.println(obj.name);

11. }

12. }

694 Chapter 8: Inner Classes

What is the result?

A. An exception occurs at runtime at line 10

B. Zippo

C. Compilation fails because of an error on line 3

D. Compilation fails because of an error on line 9

E. Compilation fails because of an error on line 10

Prova

---

1) Dado:

class Plane {

String estática s = "-";

public static void main (String [] args) {

novo Plano (). s1 ();

Respostas do autoteste 411

System.out.println (s);

}

void s1 () {

tente {s2 (); }

catch (exceção e) {s + = "c"; }

}

void s2 () lança Exception {

s3 (); s + = "2";

s3 (); s + = "2b";

}

void s3 () lança Exception {

lance new Exception ();

}}

Qual é o resultado?

A. -

B. -c

C. -c2

D. -2c

E. -c22b

F. -2c2b

G. -2c2bc

H. A compilação falha

2) Dado:

tente {int x = Integer.parseInt ("dois"); }

Qual poderia ser usado para criar um bloco catch apropriado? (Escolha todas as opções aplicáveis.)

A. ClassCastException

B. IllegalStateException

C. NumberFormatException

E. ExceptionInInitializerError

3)Dado:

class Emu {

String estática s = "-";

public static void main (String [] args) {

tentar {

lance new Exception ();

} catch (exceção e) {

```
 tentar {
 tente {lançar uma nova exceção ();
 } catch (Exceção ex) {s + = "ic"; }
 lance new Exception (); }
 catch (exceção x) {s + = "mc"; }
 finalmente {s + = "mf"; }
 } finalmente {s + = "de"; }
 System.out.println (s);
}}
```

Qual é o resultado?

A. -ic de

B. -mf de

C. -mc mf

D. -ic mf de

E. -ic mc mf de