DR1

Disciplina Regular 1 Fundamentos do Desenvolvimento Java

Graduação em Engenharia de Software - 2020

DR1

Etapa 2 Aula 1

Estruturas de Controle

DR1

Competências Trabalhadas Nesta Etapa

- Compreender os fundamentos da lógica de programação utilizando o Java e o Netbeans IDE:
  - Conhecer a estrutura básica de um programa Java.
  - Compilar um programa.
  - Corrigir erros no código.
  - Executar um programa.
- Escrever programas em Java utilizando variáveis, estruturas condicionais, loops e vetores:
  - Implementar a estrutura básica de um programa em Java.
  - Implementar o uso de váriáveis.
  - Implementar o uso de estruturas condicionais.
  - Implementar o uso de estruturas de repetição.
  - Implementar o uso de vetores.

DR1

Estruturas de Controle

Estruturas Condicionais

DR1

- Executam um grupo de comandos, dependendo do valor de uma condição.
- Caso haja uma instrução, as chaves podem ser omitidas (mas não devem).

```
if (condição) comandos [else comandos]
```

```
if (condição)
  instrução ou bloco de instruções cercados por chaves
[else if (condição-n)
  instrução ou bloco de instruções cercados por chaves]
[else
  instrução ou bloco de instruções cercados por chaves]
```

Estruturas Condicionais

DR1

```
double imposto = 0;
if (salario >= 1000 && salario < 2499)
   imposto = salario * 0.1;
else
   if (salario >= 2500 && salario < 5000)
      imposto = salario * 0.2;
   else
      if(salario >= 5000)
         imposto = salario * 0.25;
```

Estruturas Condicionais

DR1

```
double imposto = 0;
if (salario >= 1000 && salario < 2499){
   imposto = salario * 0.1;
} else if (salario >= 2500 && salario < 5000) {
   imposto = salario * 0.2;
} else if(salario >= 5000) {
   imposto = salario * 0.25;
}
```

Estruturas Condicionais

DR1

- O Operador Ternário é uma alternativa ao comando if.
- O resultado deste operador é o retorno de um valor de qualquer tipo.

```
expressão lógica ? retorno se true : retorno se false
```

```
int total = salario < 1000 ? salario : salario - IR;
```

Seletores

DR1

- Executam um determinado grupo de comandos, dependendo do valor de uma expressão.

```
switch (expressão_teste) {
[case expressão constante 1:
    comandos]
[default expressão constante n:
    comandos]
}
```

- expressão_teste é:
  - byte, short, int, char ou String.

Seletores

DR1

```
switch (diaSemana) {
    case 1:
        System.out.println("Hoje é segunda-feira");
        break;
    case 2:
        System.out.println("Hoje é terça-feira");
        break;
    ...
    case 7:
        System.out.println("Hoje é domingo");
        break;
    default:
        System.out.println("Opção inválida!!!");
        break;
}
```

Seletores

DR1

```
switch (diaSemana) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        System.out.println("Hoje é dia de trabalho!!");
        break;
    case 6:
    case 7:
        System.out.println("Hoje é fim de semana!!!");
        break;

    default:
        System.out.println("Opção inválida!!!");
        break;
}
```

Repetição Condicional

DR1

- Repete um bloco de comandos enquanto uma condição for **verdadeira**.

```
while (condição) {
    comandos
}
```

Repetição Condicional

DR1

- Repete um bloco de comandos enquanto uma condição for **verdadeira**.

```
do{
    instrução ou bloco de instruções;
}while (condição);
```

Repetição Finita

DR1

- Repete um grupo de comandos durante um número específico de vezes.

```
for (inicialização; condição de término; passo){
   instrução ou bloco de instruções
}
```

- **inicialização** – declara-se uma variável de controle do loop.
- **condição de término** – é uma expressão lógica envolvendo a variável de controle e o valor final desta.
- **passo** – expressão de incremento ou decremento envolvendo a variável de controle.

Repetição Finita

DR1

```
public class Tabuada {
  public static void main(String args[]) {
    for (int i = 1; i <=10 ; i++) {
      System.out.println("Tabuada do " + i);
      for (int j = 2; j <= 9; j++)
        System.out.println(i + " * " + j + " = " + (i*j));
    }//for i
  }//main
}//class
```

Desvios de Fluxo

DR1

```
 1.  for (int i = 1; i <= limite; i++) {
 2.
 3.    System.out.println("Loop passo " + i);
 4.    if(teste_condicao) {
 5.
 6.      break; //vai para a linha 12
 7.    } else {
 8.
 9.      continue; //vai para a linha 1
10.    }
11.  }
12.  System.out.println("Fim do Loop");
```

DR1

Arrays

Definições

DR1

- Características principais dos Arrays (vetores):
  - Tamanho fixo.
  - Todos os elementos possuem o mesmo tipo de dados e são acessados com o mesmo identificador.
  - Cada elemento possui um índice.
  - Cada posição (índice) só pode guardar um valor.

Declaração (passo 1 de 3)

DR1

- Declaração de arrays:

```
String [] args;
float notas[];
double [] salario;
String nome, documento[]; //só documento é array
String [] time, titulo; // time e titulo são arrays
```

Dimensionamento (passo 2 de 3)

DR1

- Uma vez declarados faz-se a criação do array em memória (dimensionamento) com o operador new seguido do tipo e do tamanho:

```
//arrays tem o primeiro índice igual a zero. Na instrução
//abaixo os elementos do vetor serão: notas[0]...notas[3]
notas = new float[4];
salario = new double[1000];

int numeroTimes = 64;
time = new String[numeroTimes];

int qualquer = umMetodo();//método umMetodo() retorna um int
titulo = new String[qualquer];

float f[] = new float[4]; //declaração e criação feitas na mesma linha
```

Inicialização (passo 3 de 3)

DR1

- Para arrays de tipos primitivos essa fase não é obrigatória.
- Para arrays de objetos é necessária a inicialização antes da utilização dos objetos.
- Enquanto não forem inicializados, seus valores serão null.

Inicialização (passo 3 de 3)

DR1

- Caso se queira inicializar os elementos de um array com outros valores que não considerados padrão, podem ser usadas duas formas. A primeira, através da atribuição individual:

```
notas[0] = 10;
notas[1] = 8.5;
notas[2] = 4;
notas[3] = 7;
```

- Outra forma: através de inicializadores de array.

```
float[] notas = {10, 8.5f, 4, 7};
```

Inicialização (passo 3 de 3)

```
public class TestaArray{
   public static void main(String args[]){
      int numeros[] = new int[4];
      System.out.println("numeros[2] = " + numeros[2]);

      float[] notas = {10, 8.5f, 4, 7};
      System.out.println("\nnotas[2] = " + notas[2]);

      String cores[] = new String[10];
      System.out.println("\ncores[3] = " + cores[3]);
      cores[0] = "Amarelo”;
      cores[1] = "Verde";
      for (int i = 2 ; i < cores.length; i++){
         cores[i] = new String();
      }
      for (int i = 0 ; i < cores. length; i++){
         System.out.println("cores[" + i + "] = " +cores[i]);
      }
   }
}
```

numeros[2] = 0

notas[2] = 4.0

cores[3] = null
cores[0] = Amarelo
cores[1] = Verde
cores[2] =
cores[3] =
cores[4] =
cores[5] =
cores[6] =
cores[7] =
cores[8] =
cores[9] =