C206 - Programação Orientada a Objetos
Fabio de Moura Camargo Neto
github.com/bio353

Primitivas e Fluxo de Código

Código em Java (`*.java`)

Classe e Método Principal

Em Java, nós utilizamos de um método principal para executar a aplicação. Para criá-lo, devemos adicioná-lo dentro de uma classe principal que, para fins de convenção, iremos chamá-la de `Main`.

```
// Classe Principal
public class Main {
    // Método Principal
    public static void main(String[] args) {
        // Código ...
    }
}
```

Entrada e Saída de Dados

Para realizarmos uma operação de saída de dados, mostrando valores em um terminal, utilizamos do método `System.out.print()` ou `System.out.println()`.

```
import java.util.Scanner;

// Classe Principal
public class Main {
    // Método Principal
    public static void main(String[] args) {
        // Saída de Dados
        System.out.println("Oi, mundo!");
    }
}
```

Para entrada de dados através do terminal de comando, entretanto, devemos criar uma instância da classe `Scanner` e, através dele, chamar a função `nextLine()`.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Saída de Dados
        System.out.print("Qual o seu nome?");
        // Entrada de Dados
        Scanner scanner = new Scanner(System.in);
        String name = scanner.nextLine();
        // Saída de Dados
        System.out.println("Oi, " + name + "!");
    }
}
```

Obs.: É interessante notar que `System`, `Scanner` e `String` são classes! Diferentemente dos tipos primitivos, elas possuem vários métodos que ajudam na realização de certas operações.

Condicionais (`if-else`)

Condicionais são expressões que utilizam de valores *booleanos* (`verdadeiro` e `falso`) para controlarem o fluxo do código.

```
 import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.print("Qual a sua idade? ");
        Scanner scanner = new Scanner(System.in);
        int age = scanner.nextInt();
        if (age < 18) {
            System.out.println("Muito novo pra beber...");
        } else {
            System.out.println("Pode beber, mas tenha responsabilidade.");
        }
    }
}
```

Laços de Repetição (`for`, `while` e `do-while`)

Muitas vezes, queremos executar o mesmo comando várias vezes, com algumas leves diferenças entre cada vez que o executamos. Para isso, podemos criar *loops* com `for`, `while` e `do-while`, principalmente, para repetir um trecho de código um determinado número de vezes.

```
public class Main {
    public static void main(String[] args) {
```

```
        System.out.println("Vamos contar de 0 a 10?");
        // Loop de 0 a 10 (for)
        for (int i = 0; i <= 10; i++) {
            System.out.println(i + "...");
        }
        System.out.println("\nVamos contar de 10 a 0?");
        // Loop de 10 a 0 (for)
        for (int i = 10; i >= 0; i--) {
            System.out.println(i + "...");
        }
    }
}
```

É importante observar que utilizamos de um condicional como condição de parada. Mas, como um condicional retorna um valor *booleano*, podemos utilizar um valor diretamente, como `true`, por exemplo, assim fazendo um *loop* indeterminado.

```
import java.util.Scanner;

public class Main {
    public static void main(String[]args) {
        Scanner scanner = new Scanner(System.in);
        // Loop "infinito"
        while (true) {
            System.out.print("Por favor, entre com zero: ");
            int num = scanner.nextInt();
            if (num == 0) {
                break;
            }
            System.out.println("Sinto muito, mas precisa entrar com zero!");
        }
        System.out.println("Finalmente...");
    }
}
```

Só para não falarem que não coloquei nenhum exemplo de `do-while`...

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Vamos contar de 0 a 10?");
        int i = 0;
        // Loop de 0 a 10 (do while)
        do {
            System.out.println(i + "...");
            i++;
        } while (i <= 10);
    }
}
```

Funções ou Métodos

Para criação de funções (ou métodos), devemos ter em mente duas coisas:

1. toda função retorna um valor ou é `void`;
2. toda função tem um espaço reservado para passagem de parâmetros.

```
public class Main {
    static void sayHello() {
        System.out.println("Oi, mundo!");
    }

    static int calculateSum(int a, int b) {
        return a + b;
    }

    public static void main(String[] args) {
        sayHello();
        int num1 = 18;
        int num2 = 3;
        int sum = calculateSum(num1, num2);
        System.out.println(num1 + " + " + num2 + " = " + sum);
    }
}
```

Quanto ao `static`, ele só está aqui por conta da função `main` ser estática. Falaremos mais a respeito disso na próxima aula, então não se preocupem com o significado de modificadores como `static` ou `public` agora. 😉

Exercícios Práticos

Questão N.º 1

A fórmula para calcular a área de uma circunferência é: $área = \pi \cdot raio^2$. Considerando para este problema que $\pi = 3.14$, efetue o cálculo da área, elevando o valor de raio ao quadrado e multiplicando por π. A entrada deverá conter o valor do raio, em ponto flutuante. Para a saída, apresente uma mensagem com a área calculada.

Questão N.º 2

Leia 3 valores, representando três notas de um aluno. A seguir, calcule a média do aluno, sabendo que a primeira nota tem peso 2, a segunda tem peso 3 e a terceira tem peso 5. Considere que cada nota pode ir de 0.0 até 10.0, sempre com uma

casa decimal. A entrada deverá conter 3 valores com uma casa decimal. Imprima, no fim, a uma mensagem mostrando a média deste mesmo aluno.

Questão N.º 3

Leia um valor inteiro maior do que 1 e menor do que 1000 e mostre os números ímpares de 1 até este mesmo valor, inclusive, se for o caso.