Aula 2

Anotações

- Criar pacote
  - Clicar com botão direito do mouse sobre a pasta “src” -> New -> Package -> digita o nome do pacote (“br.com.<nome do projeto>.<responsabilidade>”) -> Finish
- Criar classe principal
  - Clicar com botão direito do mouse sobre o pacote -> New -> Class -> digita o nome da classe e marca a opção (se aparecer) “`public static void main(String[] args)`” -> Finish
- Criar demais classes
  - Clicar com botão direito do mouse sobre o pacote -> New -> Class -> digita o nome da classe -> Finish

Anatomia do método main

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


                System.out.println("Olá Mundo!!");


        }
}
```

O método main é padrão para qualquer aplicação java. Por regra, todo método main deverá ser: Público, estático, sem retorno(void), com nome de "main", e deverá receber como argumento um array do tipo String.

- **public**: refere-se a visibilidade do método. Quando dizemos que o método é de visibilidade "public", estamos dizendo que este método poderá ser acessado por outras classes.
- **static**: nos garante que somente haverá uma, e não mais que uma, referência para nosso método main, ou seja, todas as instâncias da classe irão compartilhar a mesma cópia do método main.
- **void**: refere-se ao tipo de retorno que esse método terá. Nesse caso, como o tipo de retorno deve ser "void", ou seja , "vazio", esse método não retornará valor nenhum.
- **(String[] args)**: refere-se aos argumentos que serão passados para esse método, sendo obrigatório no caso do método main. Ele é uma forma do "mundo externo" comunicar-se com sua aplicação através de argumentos
- **{}**: as chaves indicam até onde certa classe ou método se estende (escopo). O código que queremos inserir neste método deverá ser escrito dentro do espaço das chaves.

Vamos praticar a

Alguns conceitos antes de testarmos outro exercício.

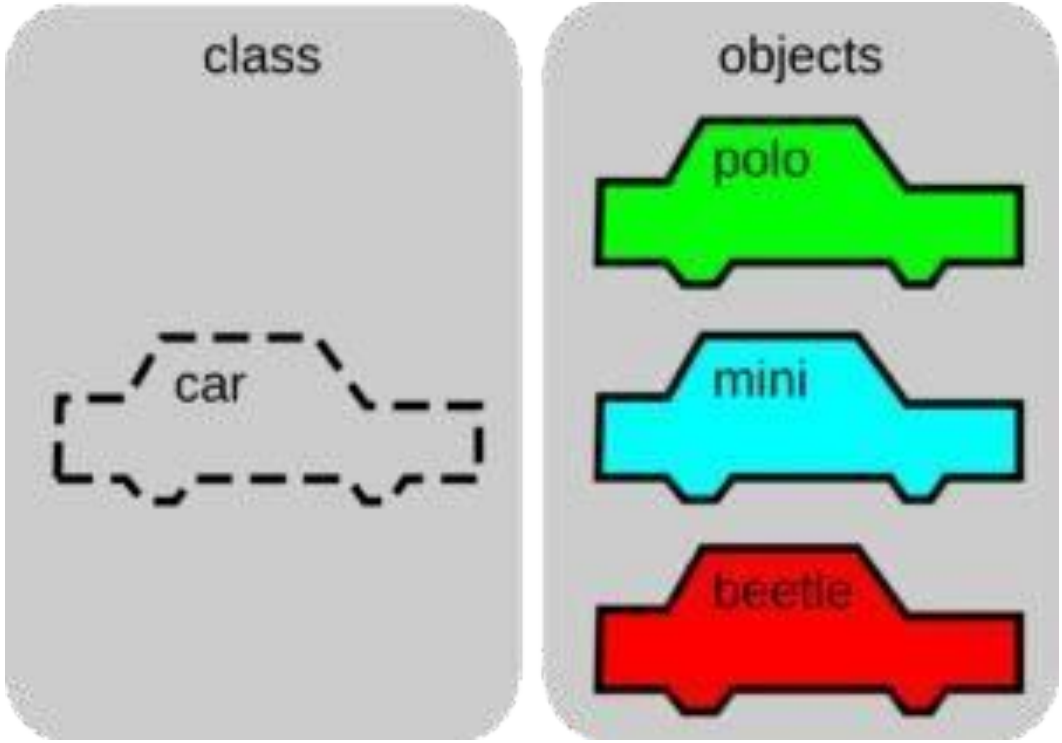
- O Java é uma linguagem Orientada a Objetos;
- A programação é feita a partir de classes;
- Cada classe representa o projeto de um objeto;
- A método main() é obrigatória e estará contida dentro de uma classe;
- O Java agrupa as classes em pacotes (utilizados para organizar as classes da sua aplicação e ajuda na reutilização de código);
- Para uma classeA usar uma classeB é preciso “importar” a classeB na classeA;

Vamos praticar a

Suas características são seus atributos (dados atrelados ao objeto) e seus comportamentos são suas ações, métodos!

Podemos dizer então que um carro específico é um objeto, ou seja, nada mais que uma instância dessa classe chamada "carro".

objeto -> instância - registro(linhas na tabela)
classe -> tabela - entidade
atributos -> variáveis - coluna



Vamos praticar a

Residente	Classes/Tabelas	Disciplina
nome sobrenome email cpf	Atributos/ Colunas	nome professor nível
nome = Lucas sobrenome = Silva email = s@gmail cpf = 123456	Objetos/ Registros	nome = POO professora = Déby nível = Hard

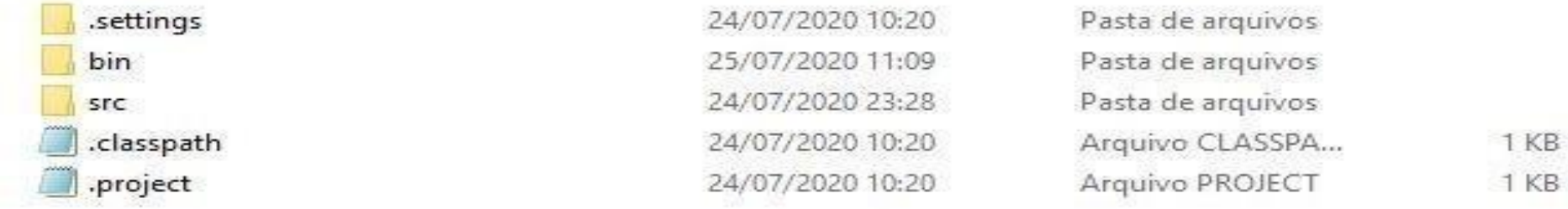
Escopo de

- O que define um escopo?
  - Basicamente o uso de chaves { }
- No Java, podemos declarar variáveis a qualquer momento. Porém, dependendo de onde você as declarou, ela vai valer de um determinado ponto a outro;
- Escopo da variável é o nome dado ao trecho de código em que aquela variável existe e onde é possível acessá-la;
- Quando abrimos um novo bloco com as chaves, as variáveis declaradas ou inicializadas ali dentro só valem até o fim daquele bloco.

UTILIZANDO O ECLIPSE - ALGO MAIS...

Pastas do Projeto

Dentro do diretório como o nome do projeto onde o Workspace foi criado temos a estrutura de pastas. abaixo:



- A pasta .settings armazena configurações de bibliotecas do projeto.
- A pasta bin contém os arquivos .class
- A pasta src os arquivos .java
- O arquivo .classpath serve para informar onde serão armazenados os arquivos .class e .java
- O arquivo .project é utilizado pelo eclipse para configurações referente ao projeto.

Para visualizarmos a estrutura de pastas no Eclipse pressione CTRL+3 digite **navigator**

BOAS PRÁTICAS E CONVENÇÕES

- **Pacotes:** eles devem ser escritos de forma semelhante a um endereço web, só que de trás para frente e ao final, indicamos um nome (ou um conjunto de nome), que classifica as classes agrupadas. (Ex.: “br.com.poo.ecommerce”, ‘br.com.poo.ecommerce.entities”)

- **Classes e Interfaces:** nomes das classes e interfaces iniciam com letra maiúscula, sendo simples e descritivo. Caso seja nome composto utiliza-se o padrão *PascalCase.* (Ex.: “Usuario”, “ContaCorrente”)

- **Métodos:** os métodos seguem o padrão *CamelCase*. Como os métodos executam alguma ação, procure usar verbos para seu nome. (Ex.: “imprimirValor”, “listarProdutos”, “calcularMedia”)

- **Variáveis:** a convenção é a mesma adotada para métodos, com nomes curtos e significativos (ex.: “nome”, “nota”, “mediaAluno”). Evitar variáveis com apenas um caracter, a não ser que seja índice em repetições ou vetores (Ex.: “x”, “y”, “i”). Em constantes todas as letras deve estar em maiúsculas e separadas por “_” (Ex.: “JUROS”, “DATA_CORTE”).

Novidade, nem tanto!

O que iremos aprender

- Variáveis Primitivas e Controle de Fluxo:
  - Declarando e usando variáveis;
  - Tipos primitivos e valores;
  - Casting;
  - O if e else (se, senão);
  - O while (enquanto);
  - O for (para… até… faça);
  - Escopo de variáveis;

Type

O que acontece com o seguinte trecho de código?

```
double d = 3.1415;
int i = d;
System.out.println("O valor de i é = " + i);
```

```
Exception in thread "main"
java.lang.Error: Unresolved compilation
problem:
Type mismatch: cannot convert from double to
int
```

Type

O que acontece com o seguinte trecho de código?

```
double d = 3.1415;
int i = (int)d;
System.out.println("O valor de i é " + i);
```

```
Saida : O valor de i é 3
```

Type

Casting possíveis

- A indicação Impl. quer dizer que o cast é implícito e automático, ou seja, você não precisa indicar o cast explicitamente.
- O tipo boolean não pode ser convertido para outro tipo.

PARA: / DE:	byte	short	char	int	long	float	double
byte	----	*Impl.*	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
short	(byte)	----	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
char	(byte)	(short)	----	*Impl.*	*Impl.*	*Impl.*	*Impl.*
int	(byte)	(short)	(char)	----	*Impl.*	*Impl.*	*Impl.*
long	(byte)	(short)	(char)	(int)	----	*Impl.*	*Impl.*
float	(byte)	(short)	(char)	(int)	(long)	----	*Impl.*
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

Lidando com Datas e

Date

- Podemos criar uma variável date assim:

```
Date date = new Date();
System.out.println(date);
```

Saída: Sat Aug 08 16:48:42 BRT 2020

- Na criação do objeto Date, é capturada a Data/Hora do relógio do computador;
- Mas como colocar uma data digitada pelo usuário em uma variável Date?
- Temos que usar conversores pois a data pode estar em vários formatos;
- A classe SimpleDateFormat nos servirá.

Lidando com Datas e

SimpleDateFormat

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
Date date = new Date();
System.out.println("Date antes de formatar: " + date);
System.out.println("Date depois de formatar: " + sdf.format(date));
```

O contrário também é importante: mostrar a data em forma de texto

```
//Podemos usar outros formatos no construtor do conversor
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String stx = "2024-08-30";
Date dataUsuario = sdf.parse(stx);
System.out.println("Date digitado e formatado: " + dataUsuario);
```

Lidando com Datas e

O formato do formatter

dd: dia do mês com dois dígitos (01 à 31)

MM: mês do ano com dois dígitos (01 à 12)

yyyy: ano com quatro dígitos (exemplo: 2018)

HH: horas com dois dígitos (00 à 23)

mm: minutos com dois dígitos (00 à 59)

ss: segundos com dois dígitos (00 à 59)

hh: horas (até 12) com dois dígitos (00 à 12)

Lidando com Datas e

Gregorian Calendar

- A classe Date é bem simples e serve para armazenar datas apenas.
- Ainda temos que manipular as datas não é?
- Por exemplo acrescentar 10 dias, verificar qual o dia da semana
- Para isso vamos usar a classe GregorianCalendar.
- Assim como o Date, quando instanciado é capturada a Data/Hora do relógio do computador;

Lidando com Datas e

```
//Instância
GregorianCalendar calendar = new GregorianCalendar();

//Mostra qual o dia da semana 1 = domingo, 2=segunda, etc
System.out.println(gc.get(gc.DAY_OF_WEEK));


//Adiciona 2 meses à data atual
System.out.println( gc.add(gc.MONTH, 2));

//Imprime falso. 2009 não é bissexto.
System.out.println(gc.isLeapYear(2009));


//Atribui a data do GregorianCalendar à uma variável Date
Date d1 = gc.getTime();

//Armazena a data de d1 para o GregorianCalendar gc
gc.setTime(d1);
```

Lidando com Datas e

LocalDate

- O LocalDate usa outra classe como formatador o DateTimeFormatter

```
LocalDate hoje = LocalDate.now();

System.out.println("LocalDate antes de formatar: " + hoje);

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

String hojeFormatado = hoje.format(formatter);
System.out.println("LocalDate depois de formatar: " + hojeFormatado);
```

Lidando com Datas e

LocalDateTime

O LocalDateTime usa o mesmo formatador do LocalDate

```
LocalDateTime agora = LocalDateTime.now();

System.out.println("LocalDateTime antes de formatar: " + agora);

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");

String agoraFormatado = agora.format(formatter);
System.out.println("LocalDateTime depois de formatar: " + agoraFormatado);
```

Recordar é

Desvios condicionais

- No nivelamento vimos que podemos desviar a execução de nosso código utilizando se... senao.

```
se (condicao) {
    // Execute uma parte de código
}
senao {
    // Execute outra parte de código
}
```

Recordar é

Desvios condicionais

- Em Java para desviar a execução basicamente trocamos o se por if e o senao por else.

```
if (hora == “22h”) {
    System.out.println(“FUI”);
}
else {
    System.out.println(“AQUI”);
}
//Operador ternário
(condicao) ? <se for verdade> : <se for falso>;
(hora == “22h”) ? System.out.println(“FUI”) : System.out.println(“AQUI”);
```

Só isso !?

Adicional: Switch - case

Sintaxe switch-case, conhecido em LP como escolha-caso:

```
int opcao = 5;
switch(opcao) {
    case 1:
        //bloco de código para esta opcao
        break;
    case 2:
        //bloco de código para esta opcao
        break;
    default:
        //bloco de código para esta opcao
        break;
}
```

Recordar é

Operadores lógicos

- A notação dos operadores lógicos que aprendemos anteriormente também muda:

- O operador e é representado por &&;
- O operador ou é representado por ||;
- O operador nao é representado por !;

```
int a=1, b=1, c=2, d=2;  boolean e = true;
if ( b == 10 || !(c == 2)) {
    // Execute uma parte de código
}
```

Recordar é

Laços de repetição - faca enquanto

- Vimos a estrutura do faca... enquanto no nivelamento.

```
faca {
    // Execute uma parte de código
}
enquanto (condicao)
```

Recordar é

Laços de repetição - do while

- Em Java basicamente trocamos o faca por do e o enquanto por while.

```
do {
    // Execute uma parte de código
}
while (condicao)
```

Recordar é

Laços de repetição - enquanto

- Vimos a estrutura do enquanto no nivelamento.

```
enquanto (condicao) {
    // Execute uma parte de código
}
```

Recordar é

Laços de repetição - enquanto

- Em Java basicamente trocamos o enquanto por while.

```
while (condicao) {
    // Execute uma parte de código
}
```

Recordar é

Laços de repetição - para

- Vimos a estrutura do para no nivelamento.

```
para (inicializacao; condicao; incremento)
{
    // Execute uma parte de código
}
```

Recordar é

Laços de repetição - for

- Em Java basicamente trocamos o para por for.

```
for (inicializacao; condicao; incremento)
{
    // Execute uma parte de código
}
```

- Exemplo

```
//definir uma lista nomes
//foreach
for(String nome : nomes) {
    if(nome == “Fulano”) {
    }
}
```

```
nomes.stream()
        .filter(nome -> "Fulano".equals(nome))
        .forEach(nome -> {
            //Execute uma parte de código
        });
```

Não é possível!
Só isso mesmo !?

Exercício 1

Testando castings

- 1) Declare duas variáveis do tipo int e realize sua soma. Em seguida, realize o casting destes dois inteiros para double para realizar sua divisão.

- 2) Declare dois caracteres : ‘A’ e ‘Z’, depois realize sua soma e armazene em uma variável do tipo int.
  - Qual é o resultado apresentado?
  - Por que você acha que esse foi o resultado apresentado?

Exercício 2

Em nossa empresa, há tabelas com o quanto foi gasto em cada mês. Para fechar o balanço do primeiro trimestre, precisamos somar o gasto total. Sabendo que, em Janeiro, foram gastos R$ 15000, em Fevereiro, R$ 23000, e em Março, R$ 17000, faça um programa que calcule e imprima o gasto total no trimestre. Siga os passos:

- Crie uma classe chamada BalancoTrimestral com um bloco main, como nos exemplos anteriores;
- Dentro do main (o miolo do programa), declare uma variável inteira chamada gastosJaneiro e inicialize-a com 15000;
- Crie também as variáveis gastosFevereiro e gastosMarco, inicializando-as com 23000 e 17000, respectivamente, e utilize uma linha para cada declaração;
- Crie uma variável chamada gastosTrimestre e inicialize-a com a soma das outras 3 variáveis;
- Imprima a variável gastosTrimestre.
    - Fazer com Integer, Double e BigDecimal.