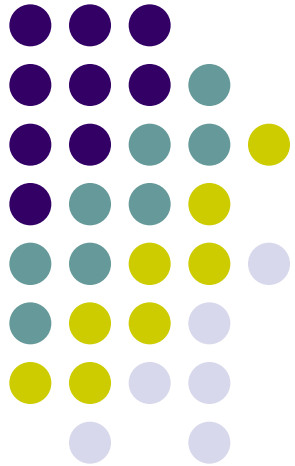
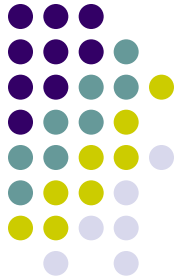
COM220
Aula 7: Programação em Java
Conceitos básicos
Parte II

Prof. Laércio Baldochi

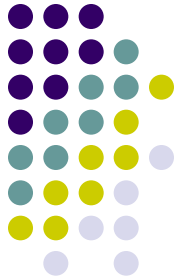


Java: Conceitos básicos



- Variáveis
  - Regiões de memória nas quais pode-se armazenar e recuperar valores
  - Todas as variáveis têm um nome, um tipo e um valor
  - Três categorias
    - variáveis de instância
    - variáveis de classe
    - variáveis locais (para métodos)

Java: Conceitos básicos



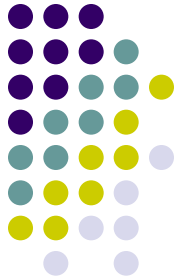
- Declaração de variáveis
  - Antes de usar uma variável é necessário declará-la.
  - São exemplos de declarações:

    int idade;

    String nome;

    boolean condição;

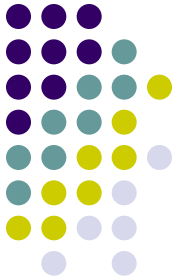
Java: Conceitos básicos



- Declaração de variáveis
  - São também declarações válidas:

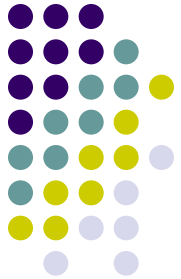
```
int x, y, z;
String nome, sobrenome;
String nome="Fulano";
int nr_identidade, idade=40;
int a=4, b=5, c=6;
```

Java: Conceitos básicos



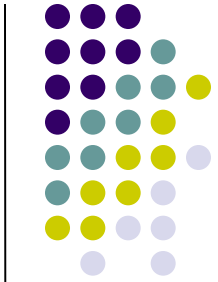
- Declaração de variáveis
  - Tipo => define que tipo de valores a variável pode armazenar.
    - Um dado primitivo
    - Um nome de classe
    - Um array

Java: Conceitos básicos



- Tipo de dado primitivo
  - Tipos de dados básicos são "embutidos" na linguagem Java (não são objetos) com o objetivo de facilitar seu uso.
  - São independentes de plataformas
    - todos os oito tipos de dado primitivo possuem o mesmo tamanho e formato em todas as plataformas

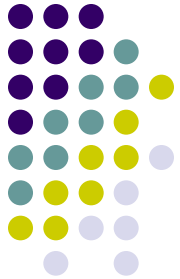
Java: Conceitos básicos



- Tipo de dado primitivo

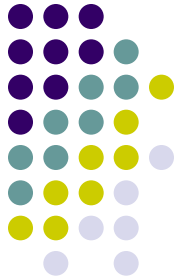
byte	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
char	2 bytes
boolean	true/false

Java: Conceitos básicos



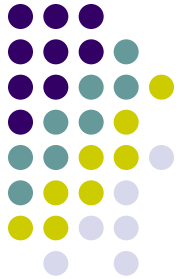
- Instância de classe
  - Além de ser declarada como um tipo de dado primitivo, uma variável pode ser declarada como instância de uma classe, como mostra os exemplos abaixo:
    - String nome;
    - Font fonte1;
    - OvalShape figura1;
    - Motocicleta moto1;

Variáveis de Instância X Classe



- Instância
  - Cada objeto tem seu próprio valor para a variável
    - Cada variável tem um slot individual para guardar seu valor
      - Vários objetos = vários slots
- Classe (static)
  - Todos objetos possuem o mesmo valor para a variável
    - Existe um único slot de memória para todos os objetos

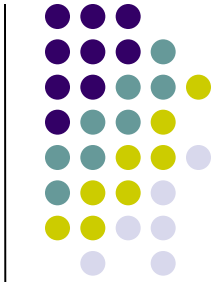
Métodos de Instância X Classe



- Instância
  - Trabalham com as variáveis de instância
    - Só podem ser chamados pelas instâncias das classes
- Classe (static)
  - Trabalham com as variáveis de classe ou servem como métodos utilitários
    - Geralmente manipulam valores passados como parâmetro ao método
    - Exemplo:

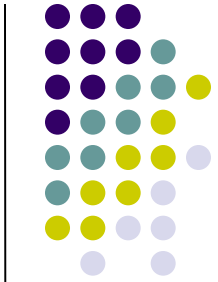
      ```
      int a = Integer.parseInt(args[0]);
      ```

Modificadores de acesso



- private
- public
- default
- protected

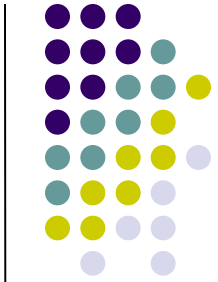
private



- Se uma classe **Carro** declara uma variável private chamada **marca**, a única classe que poderá acessar tal atributo é a própria classe **Carro**.

```
public class Carro {
    private String marca;
}
```

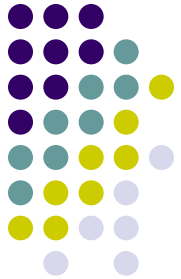
public



- Uma variável declarada com o modificador public pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence

```
public class Carro {
	public String marca;
}
```

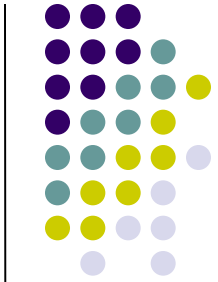
Default (sem modificador)



- Caso uma classe **Carro** declare uma variável com o nível *default*, essa variável só será acessível para as classes que estejam no mesmo pacote que a classe **Carro**.

```
public class Carro {
        String marca;
}
```

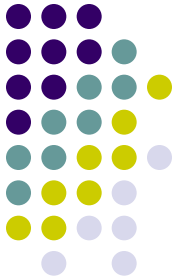
protected



- Parecido com o default
  - Se a classe Carro declarar a classe carro com o modificador *protected*, esta só será visível para classes que estejam no mesmo pacote que a classe **Carro** e para as subclasses de **Carro**.

```
public class Carro {
      protected String marca;
 }
```

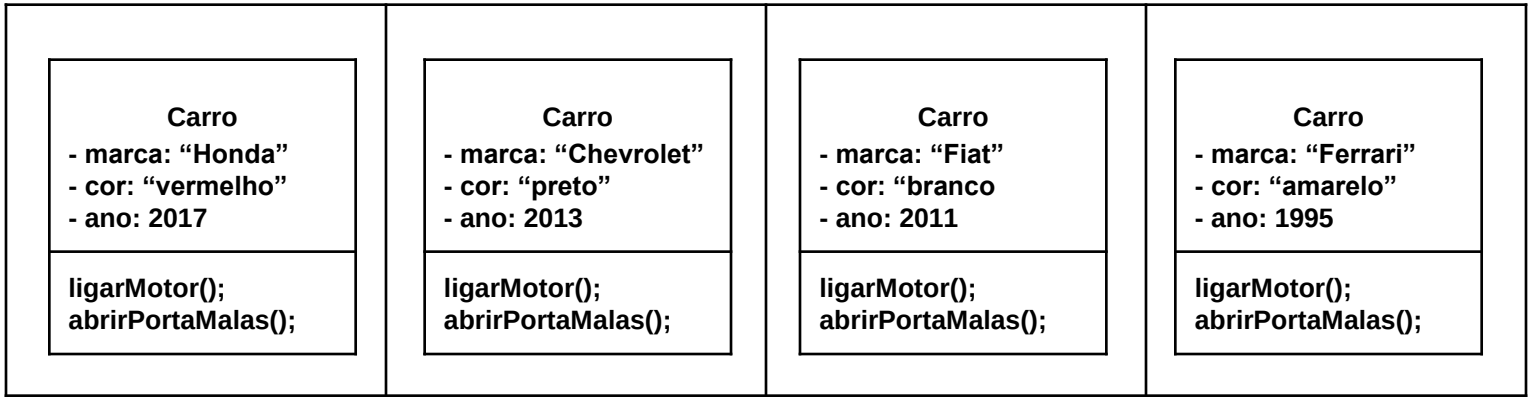
Java: Conceitos básicos



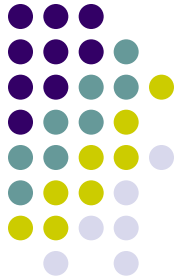
- Array

Arrays podem conter tipos primitivos (int, float, char, ...) ou instâncias de uma classe.

11	22	33	44



Java: Conceitos básicos



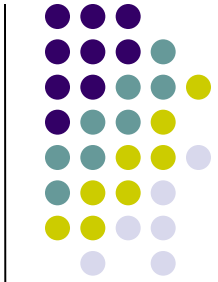
- Expressões e operadores
  - Expressões
    - Comandos que retornam valor
  - Operadores
    - Símbolos usados nas expressões

x = y + 10 => expressão

+ operador aritmético

= operador de atribuição

Java: Conceitos básicos



- Operadores

  +

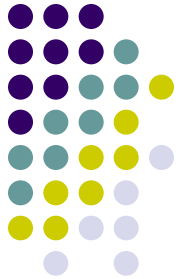
  -

  /

  *

  % => módulo

Java: Conceitos básicos



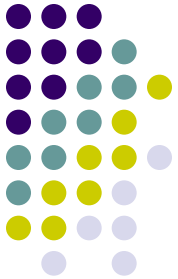
- Atribuição
  - atribuição múltipla

    x = y = z = 0;

  - Operadores de atribuição

x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y

Java: Conceitos básicos



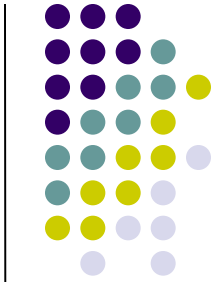
- Incremento e decremento

x = x + 1	x++	++x
x = x - 1	x--	--x

Qual é a diferença entre o prefixo e o posfixo?

Suponha o código a seguir...

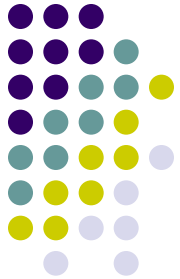
Java: Conceitos básicos



- Posfixo e prefixo

```
int x;
int y = 1;

x = ++y;
system.out.println(x);
x= y++;
system.out.println(x);
```

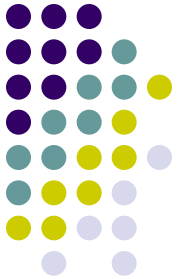
Java: Conceitos básicos



- Operadores de comparação

==	igual
!=	diferente (não igual)
<	menor que
>	maior que
<=	menor ou igual que
>=	maior ou igual que

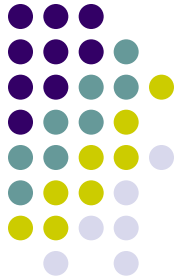
Java: Conceitos básicos



- Operadores lógicos

&&	AND
||	OR
!	NOT
...	

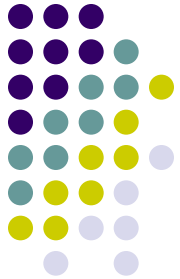
Java: Conceitos básicos



- Aritmética de Strings
  - A concatenação de Strings é feita através do sinal "+". Suponha o seguinte código

```
String nome = "Laércio"
String sobrenome = "Baldochi"
System.out.printl (nome + " " + sobrenome);
```

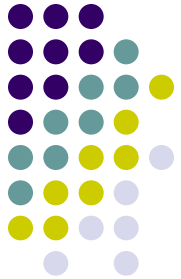
Java: Conceitos básicos



- Aritmética de Strings
  - O operador "+=" também funciona com Strings. Considere o código abaixo:

```
sobrenome += " Jr";
System.out.println(sobrenome);
```

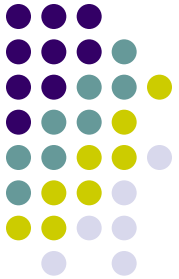
Java: Trabalhando com Objetos



- O operador new
  - Para criar a instância de uma classe usa-se sempre o operador new seguido do nome da classe, com um parênteses no final. Ex:

```
String str = new String();
Random r = new Random();
Motorcicleta m = new Motorcicleta();
Date dt = new Date(90, 4, 1, 4, 30);
```

Java: Trabalhando com Objetos

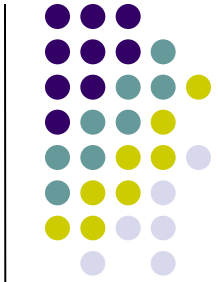


- Considere o exemplo da Data:

  Date dt = new Date(90, 4, 1, 4, 30);

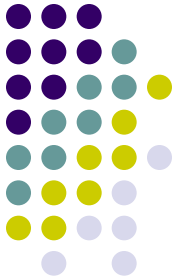
  - O número e o tipo dos argumentos usados no parênteses para a criação de uma instância é definido pela própria classe usando um método especial chamado **construtor**.
  - Para "clarear" esta idéia, considere o seguinte exemplo:

Java: Trabalhando com Objetos



```
import java.util.Date;
class CriaDatas {
    public static void main(String args[]) {
        Date d1, d2, d3;
        d1 = new Date();
        System.out.println("Data 1: " + d1);
        d2 = new Date(71, 8, 1, 7, 30);
        System.out.println("Data 2: " + d2);
        d3 = new Date("April 3 1993 3:24 PM");
        System.out.println("Data 3: " + d3);
    }
}
```

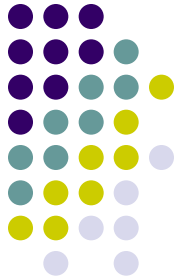
Java: Trabalhando com Objetos



- O Programa anterior produzirá a seguinte saída:

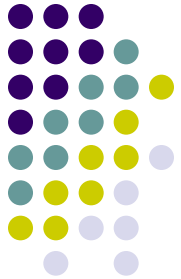
  Data 1: data e hora locais
  Data 2: Sun Aug 01 07:30:00 PDT 1971
  Data 3: Sat Apr 03 15:24:00 PST 1993

Exercício 1



- O programa anterior usa a classe Date, que está desatualizada (deprecated)
- Modifique esse programa, substituindo a classe Date por DateFormat
  - Utilize, junto com DateFormat, a classe Locale
  - Imprima a data atual nos formatos utilizados no Brasil, nos EUA, na Rússia, em Israel e na Islândia

Exercício 1



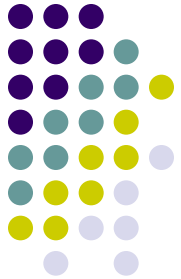
```
import java.util.*;
import java.text.*;

public class TestaData {

    public static void main(String[] args) {
            Date hoje = new Date();
            Locale brasil = new Locale ("pt","BR");
            DateFormat df = DateFormat.getDateInstance(DateFormat.LONG, brasil);
            System.out.println("Hoje no Brasil: "+ df.format(hoje));

            df = DateFormat.getDateInstance(DateFormat.LONG, Locale.FRANCE);
            System.out.println("Aujourdhui au France: "+ df.format(hoje));
    }
}
```

Exercício 2

- Considere o código a seguir

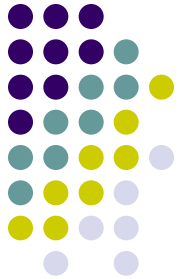
```
public class Ponto {
    public int x = 0;
    public int y = 0;

    //construtor
    public Ponto(int a, int b) {
        x = a;
        y = b;
    }
}
```

Exercício 2



```
public class Retangulo {

    public int largura = 0;
    public int altura = 0;
    public Ponto origem;

    // quatro construtores
    public Retangulo() {
        origem = new Ponto(0, 0);
    }

    public Retangulo(Ponto p) {
        origem = p;
    }

    public Retangulo(int w, int h) {
        origem = new Ponto(0, 0);
        largura = w;
        altura = h;
    }
```

Exercício 2



```
    public Retangulo(Ponto p, int w, int h) {
        origem = p;
        largura = w;
        altura = h;
    }

    // método para movimentar o retângulo
    public void move(int x, int y) {
        origem.x = x;
        origem.y = y;
    }

    // método para computar a área do retângulo
    public int getArea() {
        return largura * altura;
    }
}
```

Exercício 2

- Crie a classe TestaRetangulo
  - Instancie 4 retângulos usando o construtor apropriado
    - Retângulo invisível na origem do plano cartesiano
    - Retângulo invisível no ponto (50, 100)
    - Retântulo de tamanho (100, 150) na origem do plano cartesiano
    - Retântulo de tamanho (200, 300) no ponto (150, 200)
  - Movimente os retângulos de tal forma que nenhum fique posicionado na origem – ponto (0,0)
  - Informe a área dos retângulos criados