Aula 3

Professores:

Carlos Bazílio Isabel Rosseti

Tipos, expressões e comandos de Java

Conteúdo:

- revisão da aula anterior
- motivação
- tipos básicos
- operadores
- comandos
- exercício



- classes
 - atributos
 - métodos

```
exemplo:
```

```
class Point{
  int x, y;
  void move(int dx, int dy){
    x = x + dx;
    y = y + dy;
  }
}
```



- classes
 - atributos
 - métodos
- objetos

```
exemplo:
```

```
class Point{
  int x, y;
  void move(int dx, int dy){
    x = x + dx;
   y = y + dy;
public static void main
(String[] args){
    Point p = new Point();
    p.move(9,3);
    p.x = 0;
```



- classes
 - atributos
 - métodos
- objetos
- membros de classe

```
exemplo:
```

```
class Produto {
   static int prox_id = 0;
   int id;
   Produto() {
      id = prox_id++;
   }

   public static void main
   (String[] args) {
      Produto l=new Produto();
      Produto c=new Produto();
   }
}
```



- classes
 - atributos
 - métodos
- objetos
- membros de classe
- primeiro programa: "Olá Mundo!"

```
exemplo:
```

```
class Mundo {
  public static void main
  (String[] args) {
    System.out.println("Olá
    Mundo!");
  }
}
```



- classes
 - atributos
 - métodos
- objetos
- membros de classe
- primeiro programa: "Olá Mundo!"
- compilação: **javac**
- execução: **java**

exemplo:

```
class Mundo {
  public static void main
  (String[] args) {
    System.out.println("Olá
    Mundo!");
  }
}
```

- javac Mundo.java
- ▶ java Mundo



Motivação

- primeiros passos para aprender uma linguagem de programação:
- conhecer seus tipos básicos e as operações válidas para cada tipo
- instruir-se em seus comandos
 - responsáveis por atualizar variáveis e controlar o fluxo de execução de um programa

esta aula é de fundamental importância no aprendizado de Java...



Tipos básicos de Java



Java é uma linguagem fortemente tipada

cada variável precisa ter um tipo declarado



existem 8 tipos primitivos em Java:

- 4 tipos de números inteiros;
- 2 tipos de números ponto flutuante;
- 1 tipo caracter;
- 1 tipo lógico.



Tipo básico: inteiro

byte: número com sinal (1 byte) cexemple

- **-- -128** a **127**
- short: número com sinal (2 bytes)
 - **-- -32.768** a **32.767**
- int: número com sinal (4 bytes)
 - pouco mais de 2 bilhões
- long: número com sinal (8 bytes)
 - ± 9.223.372.036.854.775.807

```
class Teste{
  public static void main
  (String[] args){
    byte b;
    short s;
    int i, j;
    long l;
  }
}
```



Tipo básico: ponto flutuante

float: número com sinal (4 bytes) exemplo

```
- ± 3.40282347E+38
```

double: número com sinal (8 bytes)

```
= ± 1.7976931348623E+308
```

```
class Teste{
  public static void main
  (String[] args){
    float fl;
    double d, e;
  }
}
```



Tipo básico: caracter

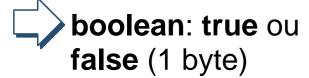
- **Char**: caracter UNICODE (2 bytes)
- tabela UNICODE:
 - → ASCII;
 - maioria dos caracteres das linguagens naturais existentes.

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
    char a;
    char c;
  }
}
```



Tipo básico: lógico



```
exemplo:

class Teste{
  public static void main
  (String[] args){
   boolean boo;
```



Operadores



Java oferece os seguintes tipos de operadores básicos:

- de atribuição;
- aritméticos;
- de incremento e de decremento;
- relacionais;
- bit a bit;
- condicional.



Operador de atribuição





após declaração de variável, deve-se inicializá-la com um valor válido.

```
class Teste{
  public static void main
  (String[] args){
    int i = 2, j = 3;
    float fl;
    fl = 7.5F;
    double d = 3.9;
    char c = 'A';
    boolean boo = true;
  }
}
```



Operadores aritméticos

- válidos para inteiros e ponto flutuante:

 - subtração: -
 - divisão: /
 - inteira
 - ponto flutuante
 - resto: % (inteiros)
 - com atribuição: +=, -=, *=,
 /= e %=

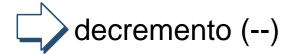
exemplo:



Observação: A variável w é float

Operadores de incremento e de decremento

```
incremento (++)
```



```
exemplo:
```



Operadores relacionais

- igualdade: ==
- desigualdade: !=
- menor que: <
- menor ou igual a: <=
- maior que: >
- maior ou igual a: >=

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
    int x = 5, y = 7;
    boolean b, c;
    b = (x == y); //false
    c = (x < y); // true
    b = (x != y); // true
    c = (y >= x); // true
}
```



Operadores lógicos

```
__>"e" lógico: &&
```

ou" lógico: ||



```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
    int x = 5, y = 5;
    boolean k = true, b;
    k = x == y;
    b = !k;
    k = ((x==5)&&(y==5));
    k = ((x==0)||(y==x));
}
```



Operadores bit a bit

- "e": &
- **"ou":** |
- "ou exclusivo": ^
- "não": ~
- deslocar bits:
 - para a direita: >>
 - para a esquerda: <<</p>
 - para a direita, pondo zeros nos bits mais significativos: >>>

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
    int r, j = 10, c = 2;
    r = j & c; // r = 2
    r = j | c; // r = 10
    r = j ^ c; // r = 8
    r = j << c; // r = 40
    r = j >> c; // r = 2
}
```



Operador condicional



representação: ?:



equivale ao comando **if-else**

exemplo:

```
class Teste{
  public static void main
  (String[] args) {
    int m, i = 15, j = 0;
    m = i > j ? i : j;
    /* m = 15 */
  }
}
```



Precedência de operadores

```
(conversão_tipo)
              %
<<
              >>>
       <=
&
Λ
&&
                                   %=
                            /=
```



Conversões entre tipos numéricos

- conversões são possíveis em Java, mas informações podem ser perdidas...
- conversões entre diferentes tipos que não precisam de cast:
 - byte → short→ int→ long → float → double

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
    double x = 9.9997;
    int y;
    y = (int) x; // y = 9
    x = 9.5F;
    float f = 0.0F;
    long l = y;
  }
}
```



Classes pré-definidas

```
textos ou "strings"
```

vetores

```
class Teste{
  public static void main(String[] args){
    String texto = "Exemplo";
    int[] lista = {1, 2, 3, 4, 5};
    String[] nomes = {"João", "Maria"};
    System.out.println(nomes[0]); // Imprime "João"
  }
}
```



Comandos



instruções do programa cujo objetivo é atualizar as variáveis ou controlar o fluxo de execução



tipos de comandos em Java:

- expressão de atribuição;
- formas pré-fixadas ou pós-fixadas de ++ e --;
- chamada de métodos;
- criação de objetos;
- blocos;
- comandos de controle de fluxo.



Bloco



é qualquer lista de instruções delimitada por um par de chaves



assim:

bloco = { de comandos> }



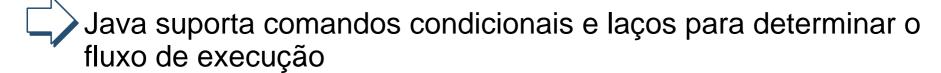
exemplo:

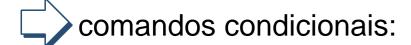
Observações:

- O bloco começa na chaves antes do int **n**
- Tem um outro bloco dentro do método main



Comandos de controle de fluxo





- ─ if-else
- switch-case-default



- while
- do-while
- for



break

comando de saída de método:

- return



Comando condicional: if-else





```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
  int a = 5, b = 3, m;
  if ((a > 0) && (b > 0))
    m = (a + b) / 2;
  else{
    System.out.println
    ("numero negativo");
    m = 0;
  }
}
```



Comando condicional: switch-case-default



- tipo char;
- todos os tipos inteiros, exceto long.
- execução começa no case correspondente
- termina no primeiro
 break seguinte a este
 case
- a cláusula **default** é opcional

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
  int a = 5;
  switch (a)
    case 5:
     System.out.println(5);
     break;
    case 0:
     System.out.println(0);
     break;
    default:
     System.out.println("erro");
```



Comando de repetição: while



while (teste) bloco



somente executa o bloco quando **teste** for verdadeiro



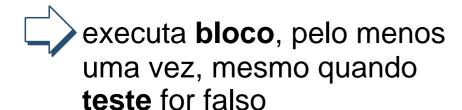
exemplo:

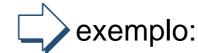
```
class Teste{
  public static void main
  (String[] args){
  int i = 0;
  while (i<10)
  {
    System.out.println(i);
    i++;
  }
}</pre>
```



Comando de repetição: do-while







```
class Teste{
  public static void main
  (String[] args){
  int i = 0;
  do{
    System.out.println(i);
    i++;
  }
  while (i<10);
}</pre>
```



Comando de repetição: for

- expressão inicial é avaliada uma vez
- 2) expressão booleana é avaliada
 - se for verdadeira, o bloco é executado
 - senão, pára
- 3) expressão de incremento é feita. Volta-se para 2).

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
  for(int i=0;i<10;i++)
     System.out.println(i);
  }
}</pre>
```



Comando de desvio: break



comando usado em:

- switch-case-default;
- laços.

quando executado em um laço, termina a execução do mesmo

```
exemplo:
```

```
class Teste{
  public static void main
  (String[] args){
    int i = 0;
  while (true)
   if (i==10) break;
   i++;
   System.out.println(i);
```



Comando de saída de método: return

- quando executa-se este comando, abandona-se o método
- valor de retorno do método
 - é o resultado da expressão que segue o comando

```
exemplo:
```

```
class Teste{
  static int média(int a,
   int b){
  return (a+b)/2;
  }

  public static void
  main(String[] args){
  int a = 9, b = 5, c;
   c = média(a,b);
   System.out.println(c);
  }
}
```



Exercício: enunciado

Projete e implemente um sistema que modele um banco. Seu projeto deve permitir a criação de vários bancos e várias contas para cada banco. Para um dado banco deve ser possível: obter seu nome, obter seu código, criar uma nova conta e obter uma conta a partir de um código.

Para cada conta corrente criada deve ser possível: obter o nome do correntista, obter o banco a qual a conta pertence, obter seu saldo, fazer uma aplicação e efetuar um débito.

Faça com que cada banco tenha um código próprio, o mesmo vale para as contas. Permita que contas de bancos diferentes tenham o mesmo número. Escreva um programa de teste que crie dois bancos e duas contas, uma para cada banco. Efetue as operações possíveis para as duas classes.



Exercício: solução (classe Banco)

```
class Banco{
  static int prox_banco = 1;
  final int MAX_CONTAS = 10;
  String nome;
  int codigo,prox_conta,ind_array;
  Conta[] contas;
```



Exercício: solução (classe Banco)

```
class Banco{
  static int prox_banco = 1;
  final int MAX_CONTAS = 10;
  String nome;
  int codigo,prox_conta,ind_array;
  Conta[] contas;

Banco(String n){
  nome = n;
  codigo = prox_banco++;
  prox_conta = 1;
  contas = new Conta[MAX_CONTAS];
  ind_array = 0;
}
```



```
class Banco{
 static int prox banco = 1;
 final int MAX CONTAS = 10;
 String nome;
 int codigo,prox_conta,ind_array;
Conta[] contas;
Banco(String n){
 nome = n;
  codigo = prox_banco++;
 prox conta = 1;
  contas = new Conta[MAX_CONTAS];
  ind_array = 0;
 int pegaCodB(){return codigo;}
```



```
class Banco{
 static int prox banco = 1;
 final int MAX CONTAS = 10;
 String nome;
 int codigo,prox_conta,ind_array;
Conta[] contas;
Banco(String n){
 nome = n;
  codigo = prox_banco++;
 prox conta = 1;
  contas = new Conta[MAX_CONTAS];
  ind_array = 0;
 int pegaCodB(){return codigo;}
 String pegaNomeB(){return nome;}
```



```
class Banco{
 static int prox banco = 1;
 final int MAX CONTAS = 10;
 String nome;
 int codigo,prox_conta,ind_array;
Conta[] contas;
Banco(String n){
 nome = n;
  codigo = prox_banco++;
 prox conta = 1;
  contas = new Conta[MAX_CONTAS];
  ind array = 0;
 int pegaCodB(){return codigo;}
 String pegaNomeB(){return nome;}
```

```
Conta criaConta(String nome) {
  Conta c;
  if(prox_conta==MAX_CONTAS)
    c=null;
  else{
    c = new Conta(nome,
    prox_conta++, this);

  contas[ind_array++] = c;
  }
  return c;
}
```



```
class Banco{
 static int prox banco = 1;
 final int MAX CONTAS = 10;
 String nome;
 int codigo, prox_conta, ind_array;
Conta[] contas;
Banco(String n){
  nome = n;
  codigo = prox_banco++;
 prox_conta = 1;
  contas = new Conta[MAX_CONTAS];
  ind array = 0;
 int pegaCodB(){return codigo;}
 String pegaNomeB(){return nome;}
```

```
Conta criaConta(String nome){
 Conta ci
 if(prox conta==MAX CONTAS)
  c=null;
 else{
  c = new Conta(nome,
  prox_conta++, this);
  contas[ind_array++] = c;
 return c;
Conta buscaConta(int cod){
 int i;
 for (i=0; i< ind array; i++)
 if(contas[i].pegaCodigo()==cod
  return contas[i];
 return null;
```

```
class Conta{
  String nome;
  int codigo;
  Banco banco;
  float saldo;
```



```
class Conta{
  String nome;
  int codigo;
  Banco banco;
  float saldo;

Conta(String n, int c, Banco b){
   nome = n;
   codigo = c;
   banco = b;
   saldo = OF;
}
```



```
class Conta{
 String nome;
 int codigo;
 Banco banco;
 float saldo;
Conta(String n, int c, Banco b){
  nome = n;
  codigo = c;
  banco = bi
  saldo = 0F;
 Banco pegaBanco() { return banco; }
```



```
class Conta{
 String nome;
 int codigo;
 Banco banco;
 float saldo;
Conta(String n, int c, Banco b){
  nome = n;
  codigo = c;
  banco = bi
  saldo = 0F;
 Banco pegaBanco() { return banco; }
 String pegaNome(){return nome;}
```



```
class Conta{
 String nome;
 int codigo;
 Banco banco;
 float saldo;
Conta(String n, int c, Banco b){
  nome = n;
  codigo = c;
  banco = bi
  saldo = 0F;
 Banco pegaBanco() { return banco; }
 String pegaNome(){return nome;}
```

```
int pegaCodigo(){
  return codigo;
}
```



```
class Conta{
 String nome;
 int codigo;
 Banco banco;
 float saldo;
Conta(String n, int c, Banco b){
  nome = n;
  codigo = c;
  banco = bi
  saldo = 0F;
 Banco pegaBanco() { return banco; }
 String pegaNome(){return nome;}
```

```
int pegaCodigo(){
  return codigo;
}

float pegaSaldo(){
  return saldo;
}
```



```
class Conta{
 String nome;
 int codigo;
 Banco banco;
 float saldo;
Conta(String n, int c, Banco b){
  nome = n;
  codigo = c;
  banco = bi
  saldo = OF;
 Banco pegaBanco() { return banco; }
 String pegaNome(){return nome;}
```

```
int pegaCodigo(){
  return codigo;
}

float pegaSaldo(){
  return saldo;
}

void aplica(float soma){
  saldo += soma;
}
```



```
class Conta{
 String nome;
 int codigo;
 Banco banco;
 float saldo;
Conta(String n, int c, Banco b){
  nome = n;
  codigo = c;
  banco = bi
  saldo = 0F;
 Banco pegaBanco() { return banco; }
 String pegaNome(){return nome;}
```

```
int pegaCodigo(){
 return codiqo;
float pegaSaldo(){
 return saldo;
void aplica(float soma){
 saldo += soma;
void retira(float soma){
 saldo -= soma;
```



Exercício: solução (classe Teste)

```
class Teste
 public static void main(String[] args){
    Banco itau = new Banco("Itau");
    Conta b, maria = itau.criaConta("Maria");
    System.out.println(itau.pegaCodB());
    System.out.println(itau.pegaNomeB());
    b = itau.buscaConta(1);
    b = itau.buscaConta(2);
    Banco bb = new Banco("Banco do Brasil");
    Conta jose = bb.criaConta("Jose");
    System.out.println(jose.pegaNome());
    System.out.println(jose.pegaCodigo());
    System.out.println(jose.pegaSaldo());
    jose.aplica(100.0F);
    jose.retira(30.5F);
    System.out.println(jose.pegaSaldo());
```

