NetBeans

Conhecendo Um Pouco da IDE

Programação de Computadores II

Professor: Edwar Saliba Júnior

Apresentação:

1) Abra a IDE (*Integrated Development Environment*), ou seja, o Ambiente Integrado de Desenvolvimento conhecido como “NetBeans”. Você verá a IDE, como na imagem a seguir (Figura 1):

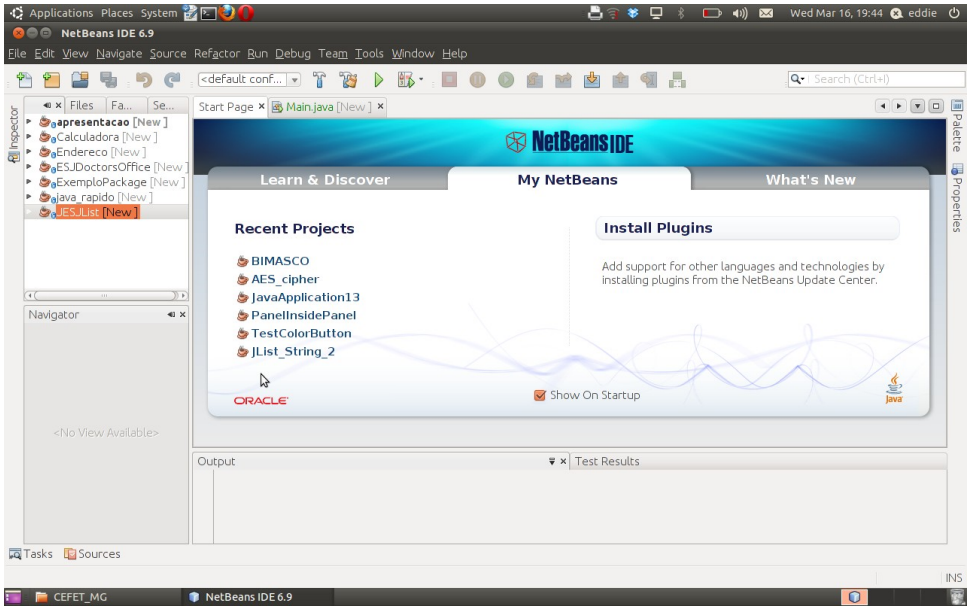

Figura 1: IDE NetBeans

Criando Um Novo Projeto de Software:

2) Vá em “*File* | *New Project*...” a seguinte tela aparecerá para você:

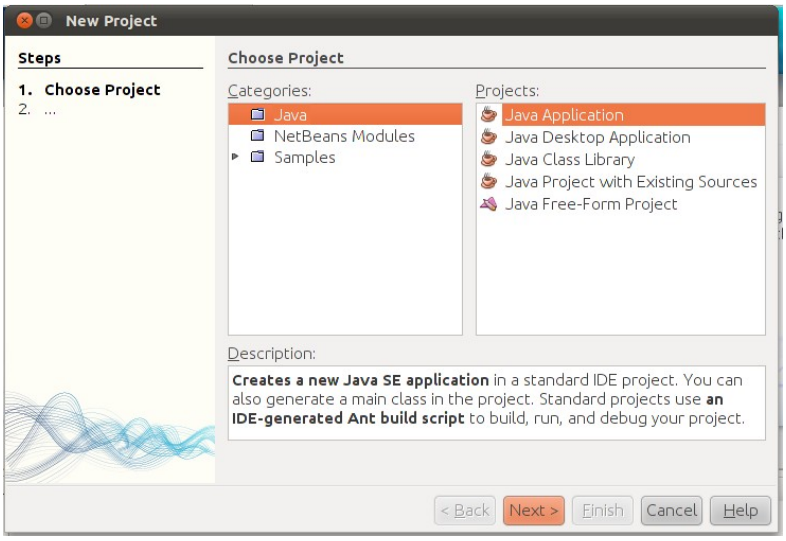

Figura 2: New Project

3) Em *Categories* escolha a opção "Java" em *Projects* escolha a opção "Java *Application*". (Figura 2), e em seguida aperte o botão *Next*. A seguinte tela aparecerá para você (Figura 3):

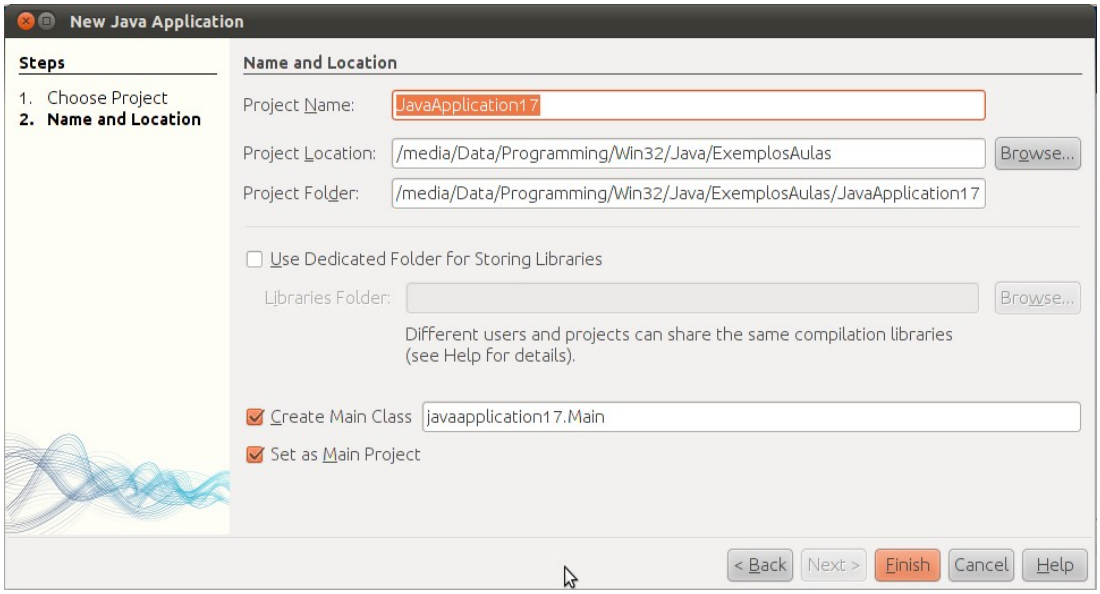

Figura 3: New Java Application

4) Na tela apresentada na Figura 3, dê um nome ao seu projeto, preenchendo o campo *Project Name* (para que você não tenha problemas com este tutorial mais adiante, sugere-se que você dê o nome deste primeiro projeto de: `apresentacao`). Lembre-se de não usar caracteres especiais. No campo *Project Location*, você poderá escolher uma pasta (diretório), onde você deseja que seu projeto seja criado. No campo *Project Folder*, você verá o caminho completo, onde estará localizado seu novo projeto. Neste campo você poderá observar também, que será criada uma pasta com o nome que você deu ao seu projeto, e esta pasta será criada dentro da última pasta do cominho que você escolheu no campo *Project Location* (campo imediatamente acima do campo *Project Folder*. E para finalizar, temos o campo *Create Main Class*, que possui um *Check Box* e, quando este estiver marcado, então a classe *Main* será criada com o nome que você escolher. Caso você não escolha um nome específico, será utilizado o nome que você deu ao projeto.

5) Tudo preenchido, então aperte o botão *Finish* (Figura 3). Será mostrado a você, uma tela com um novo projeto já iniciado (Figura 4).

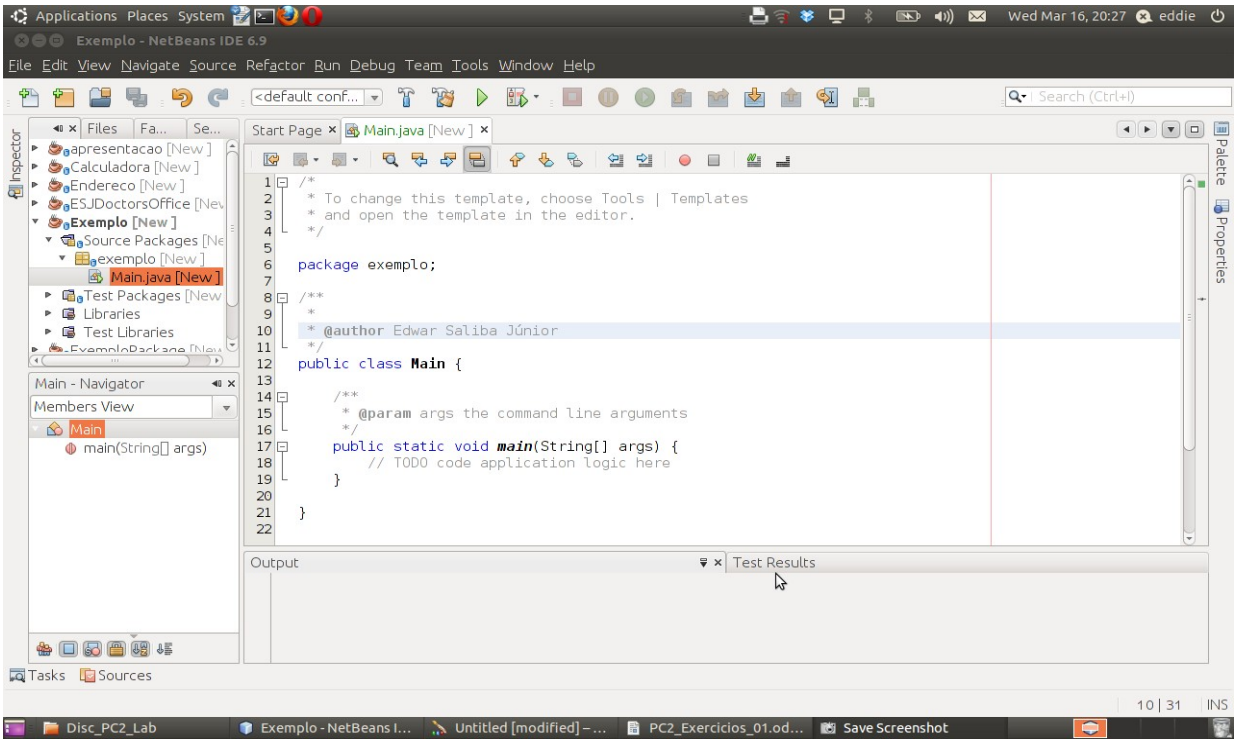

Figura 4: Projeto criado.

6) Você notará que a IDE, cria um projeto com o nome que você escolheu. Cria também um *package* com o nome que você escolheu para o projeto. Cria ainda, uma classe chamada *Main*, isto se você não desmarcou e tampouco alterou o nome no campo *Create Main Class* (Figura 3). E para finalizar, a IDE cria também um método (função) especial, chamado "*main*" (Figura 4).

7) Agora, para continuarmos nossa aula, complete o código-fonte criado automaticamente pela IDE, com o seguinte código-fonte (Figura 5):

```
6   package apresentacao;
7
8
9   import java.util.Scanner;
10
11  /**
12   * @author Edwar Saliba Júnior
13   */
14  public class Main {
15
16      /**
17       * @param args the command line arguments
18       */
19      public static void main(String[] args) {
20          int idAtual, id2028, anoAtual, anoNasc;
21          Scanner input = new Scanner(System.in);
22
23          System.out.print("Digite o ano atual: ");
24          anoAtual = input.nextInt();
25          System.out.print("Digite o ano de nascimento: ");
26          anoNasc = input.nextInt();
27
28          idAtual = anoAtual - anoNasc;
29          id2028 = 2028 - anoNasc;
30
31          System.out.printf("\nSua idade é: %d", idAtual);
32          System.out.printf("\nSua idade em 2028 é: %d\n\n", id2028);
33      }
34  }
```

Figura 5: Código-fonte "apresentacao".

8) Após ter escrito o código-fonte da Figura 5, então compile-o e execute-o. Para compilação e execução em sequencia, basta pressionar a tecla F6.

Depurando Um Código-fonte:

9) Após executado o programa, vamos agora aprender a executá-lo passo a passo. Vamos colocar um *break point* na linha 28, ou seja, na linha do comando de atribuição da variável ”idAtual”. Para tanto, dê um clique com o ponteiro o *mouse* sobre o número da linha (caso este esteja visível), caso não esteja visível, basta clicar próximo à borda da parte cinza onde são apresentados os números de linhas. Você verá uma linha rosa, algo como mostrado na Figura 6.

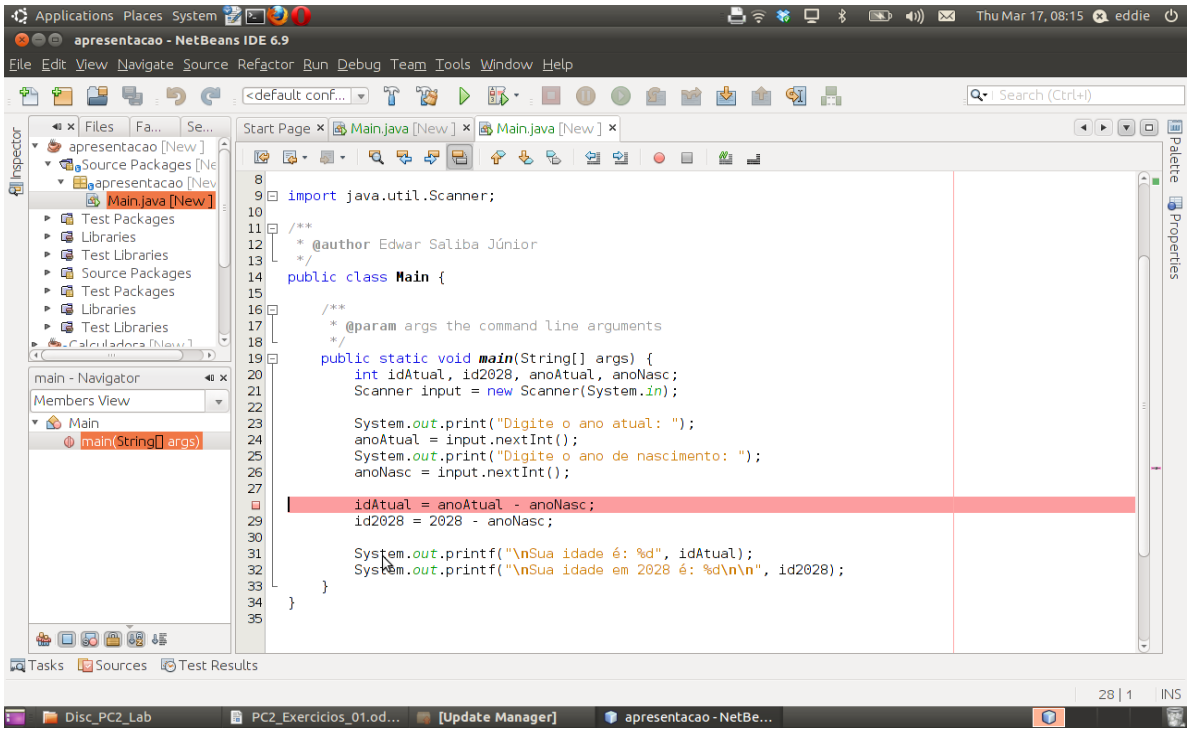

Figura 6: Break point.

10) Agora aperte as teclas “Ctrl + F5” (*Debug Main Project*). O *software* será executado, solicitando normalmente que você entre com o Ano Atual e o Ano Nascimento, após a segunda entrada de dados, o compilador encontrará o seu *break point* e ficará parado sobre ele. Isto quer dizer que a linha onde está o *break point* ainda não foi executada. Você poderá observar, que a linha do *break point* ficou verde (Figura 7).

11) Agora adicione as variáveis: idAtual, anoAtual, anoNasc e id2028 a uma tela de *Watches* (Observador). Para isto, basta clicar com o botão direito do *mouse* sobre a variável e escolher a opção *New Watch* (Figura 8).

12) Feito isto, vamos a depuração do programa.

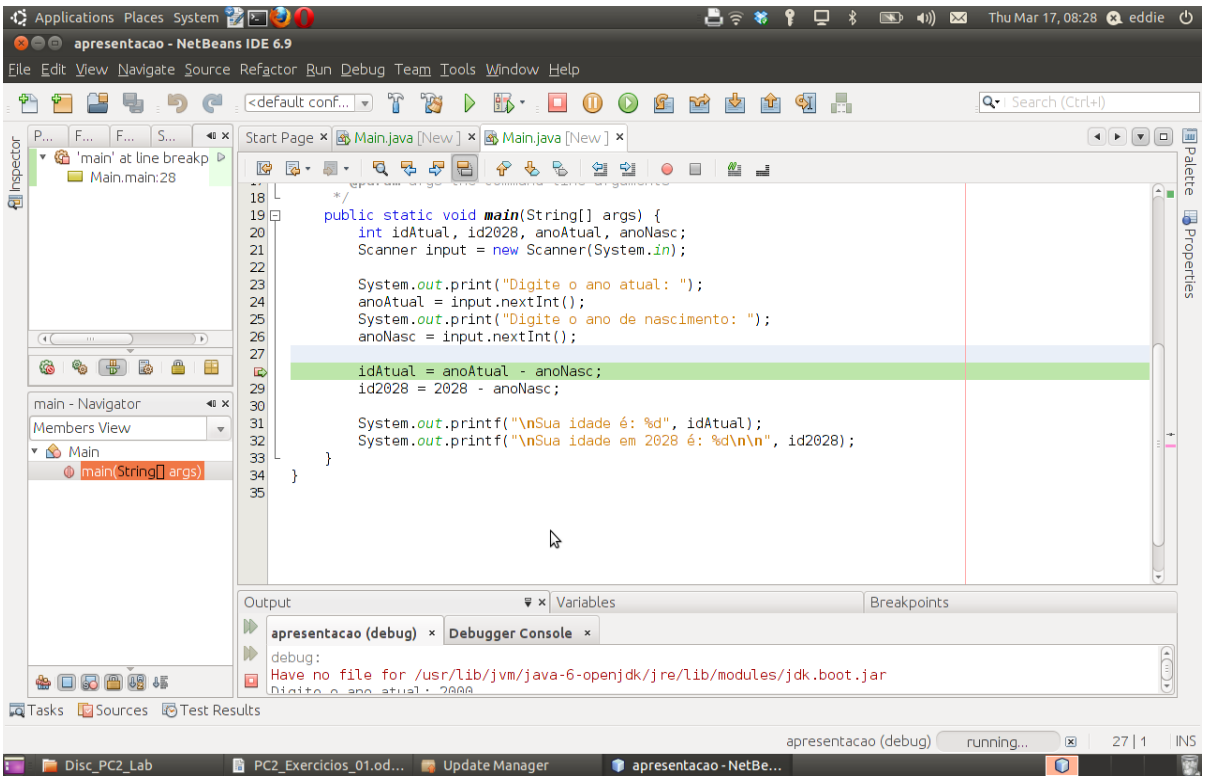

Figura 7: Depuração

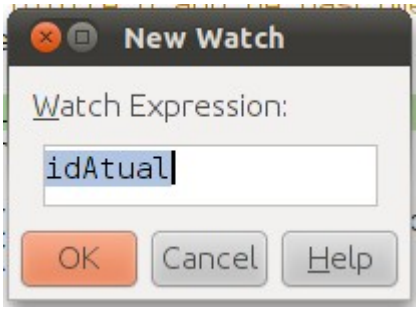

Figura 8: Add Watch.

13) Ao adicionar as variáveis na tela de *Watches* (observação), toda e qualquer mudança ocorrida nos valores destas variáveis, poderão ser observadas na parte inferior da IDE (Figura 9 - seta vermelha).

14) Pressionando a tecla F8, pode-se ver a execução (linha com sombreamento verde claro) linha a linha do código-fonte em questão. E, automaticamente, as mudanças nos valores das variáveis, através da tela *Watches* (Figura 9 - seta vermelha).

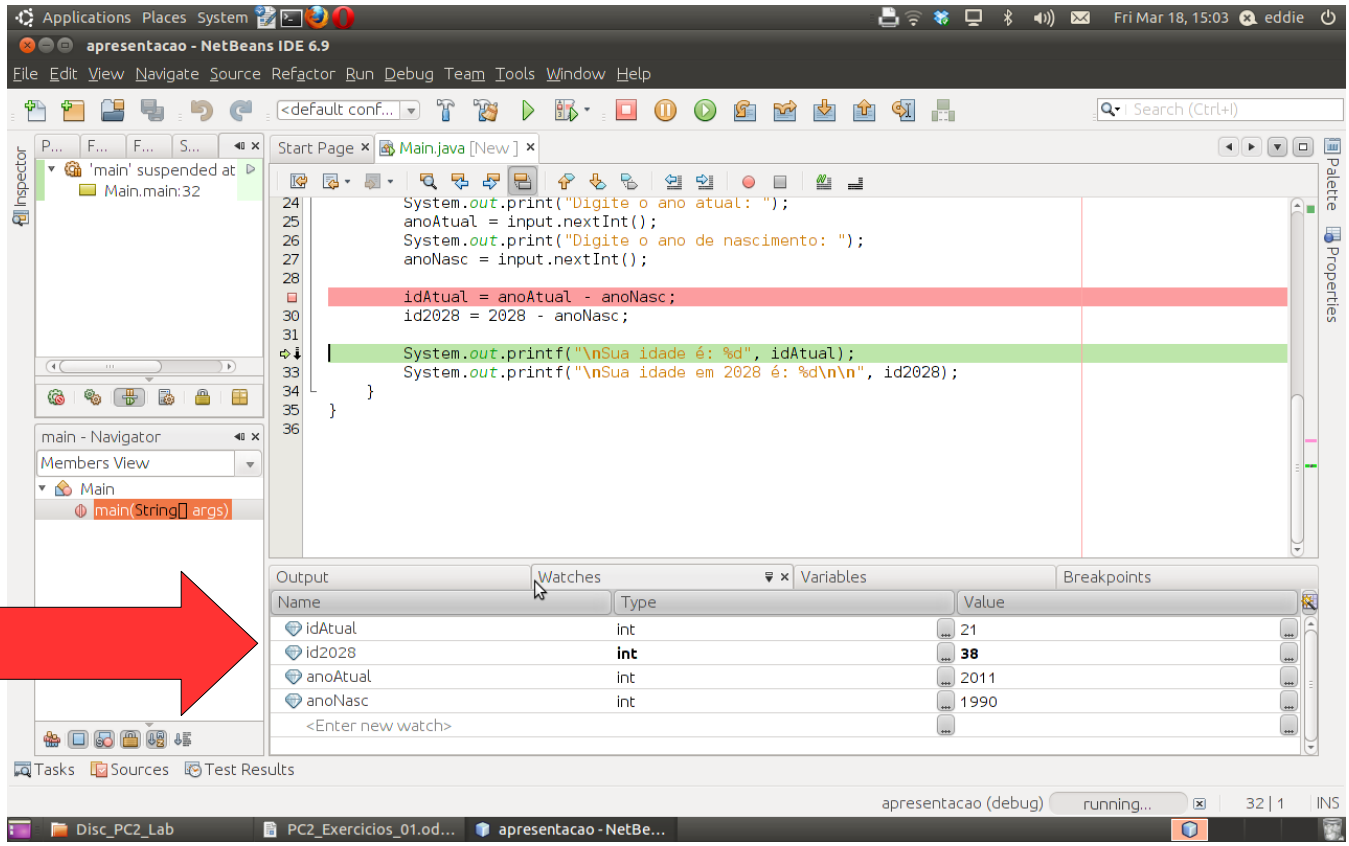

Figura 9: Watches

15) A medida que pressiona-se a tecla F8 e vai-se navegando pelo código-fonte, você poderá observar o comportamento das variáveis que adicionou na tela *Watches*. Desta forma, você poderá depurar o seu *software*, e eventualmente concertar todos os erros de lógica que encontrar.

16) Observação: a tecla F8 (*Step Over* – Passar Sobre) serve para executarmos uma função sem entrarmos em seu código-fonte. Caso seja necessário entrar no código-fonte da função, então, ao chegar na linha de código da respectiva função, deve-se apertar a tecla F7 (*Step Into* – Passar Dentro).

Entendendo o Código-fonte:

```
6   package apresentacao;
7
8
9   import java.util.Scanner;
10
11  /**
12   * @author Edwar Saliba Júnior
13   */
14  public class Main {
15
16      /**
17       * @param args the command line arguments
18       */
19      public static void main(String[] args) {
20          int idAtual, id2028, anoAtual, anoNasc;
21          Scanner input = new Scanner(System.in);
22
23          System.out.print("Digite o ano atual: ");
24          anoAtual = input.nextInt();
25          System.out.print("Digite o ano de nascimento: ");
26          anoNasc = input.nextInt();
27
28          idAtual = anoAtual - anoNasc;
29          id2028 = 2028 - anoNasc;
30
31          System.out.printf("\nSua idade é: %d", idAtual);
32          System.out.printf("\nSua idade em 2028 é: %d\n\n", id2028);
33      }
34  }
```

Figura 10: Código-fonte "apresentacao".

package – Mecanismo para organização de classes Java, dentro de um mesmo *namespace*. Esta organização pode-se dar de diversas formas: similaridade, funcionalidade e etc.

import – Semelhante ao comando *#include* da linguagem C. Através do comando *import*, pode-se utilizar outras classes e seus métodos, dentro da classe que está sendo construída.

public – Modificador de acesso, refere-se a acessibilidade que se pode ter em relação a um método, classe ou variável.

Acessibilidade de variáveis e métodos: depende dos modificadores de acesso, que se coloca (ou não) diante da variável em sua declaração, método ou classe em sua definição:

- *public*: Dá acessibilidade completa à variável. Esta pode ser acessada a partir da própria classe e também por outras classes, estando estas classes no mesmo *package* ou não.
- *protected*: Dá acessibilidade para todas as classes que estão no mesmo *package*.
- *private*: Só se pode acessar desde a própria classe.

static – O modificador *static* nos garante que somente haverá uma, e não mais que uma, referência para determinada variável ou método disponível em memória. Em outras palavras, declarando alguma coisa como *static*, todas as instâncias da classe compartilharão a mesma cópia da variável ou método. Declarar algo como *static* também permite você acessar métodos e atributos diretamente, ou seja, sem precisar criar uma instância da classe.

void – Palavra-chave utilizada na declaração de funções que não precisam retornar um valor.

int – Palavra-chave utilizada na declaração/criação de variáveis do tipo inteiro, ou seja, criação de espaços na memória do computador, capazes de armazenar valores do tipo inteiro.

public static void main(String[] args) - Assinatura do método principal.

{ - Início do corpo do método (função).

} - Fim do corpo do método.

"**int idAtual**" - Criação de variável do tipo inteiro. Ou seja, criação de um espaço na memória do computador, onde pode-se guardar valores do tipo inteiro.

// Este é um comentário de uma só linha .

/* Este é um comentário de uma ou mais linhas. *****/***

/** Este é um comentário de documentação. *****/***

Comentário - O que estiver como comentário não é executado pelo compilador. Utilizado para documentar o *software*.

System.out.print - Comando para impressão de *strings* na tela do computador.

System.out.printf - Comando para impressão de *strings*, associadas a valores em variáveis, na tela do computador.

new – Comando utilizado para instanciação de objetos.

Scanner – Classe utilizada para captação de dados via teclado.

Dica de Padronização em Java:

Em Java, o:

- nome de toda classe começa com letra maiúscula. Exemplo:
  - `public class Carro {};`
- nome de todo atributo e método, começam com letra minúscula. E quando este é formado por mais de uma palavra, então as demais palavras que compõem o nome, com exceção da primeira, iniciarão com letra maiúscula. Exemplo:
  - `int meuAtributoNovo;`
  - `void meuMetodoNovo( ){};`