CURSO: Ciência da Computação	**DATA: ____/____/ 2013**
PERÍODO: 4º.	**PROFESSOR: Andrey**
DISCIPLINA: Técnicas Alternativas de Programação	**AULA: 05**

APRESENTAÇÃO:

Nesta aula vamos ver como se faz a criação de Métodos em Java e como se usam objetos; criação de classes; Atributos e métodos (criação e utilização); Pacotes; visibilidade (encapsulamento); sobrecarga de métodos; Atributos e métodos static; métodos construtores; referência this.

DESENVOLVIMENTO:

À medida que os programas se tornam mais e mais complicados, passa a existir a necessidade de subdividir este programa em partes. Em Java este particionamento é feito a partir da criação de classes e métodos.

Métodos são trecho de programa que recebem um nome e podem ser chamados diversas vezes dentro de um programa. Pode ser uma funcionalidade que é necessária em várias ocasiões. Neste caso seu código estaria repetido em diversos lugares do programa. Para evitar esta repetição de código desnecessária, são usados os métodos. Ex:

```
import java.util.*;

class Exemplo0502
{
	public static void main (String args[])
	{
		Scanner teclado = new Scanner(System.in);
		String valor;
		String frase1="Esta é a primeira frase";
		String frase2="Esta é a segunda frase";
		String frase3="Esta é a terceira frase";
		limpaTela(); //chamada do método
		tela(frase1); //chamada do método
		valor = teclado.nextLine();
		limpaTela(); //chamada do método
		tela(frase2); //chamada do método
		valor = teclado.nextLine();
		limpaTela(); //chamada do método
		tela(frase3); //chamada do método
	}

	public static void tela(String S) //declaração do método
	{
		System.out.println(S);
	}

	public static void limpaTela() //declaração do método
	{
		for(int i=1;i<=25;i++)
		System.out.println();
	}
 }
```

Neste exemplo foram criados mais dois métodos para conter partes de código que seriam usadas mais de uma vez.

Declaração de métodos

```
qualificadores tipoDeRetorno nomeDoMetodo (tipo1 argumento1, tipo2 argumento2, ...)
{
      <corpo do método>
}
```

Exemplo

```
      public static void tela(String S) //declaração do método
      {
            System.out.println(S);
      }

      public static void limpaTela() //declaração do método
      {
            for(int i=1;i<=25;i++)
            System.out.println();
      }
```

Variáveis dos métodos

Argumentos são variáveis usadas para receber valores externos que serão usados pelos métodos. Estas variáveis são declaradas no cabeçalho do método. Estas variáveis só existirão enquanto o método estiver sendo executado. Ex. Variável S do método tela.

Variáveis locais são usadas pelos métodos para realizar operações normais dentro do método. Estas variáveis são declaradas dentro do método e só existirão enquanto o método estiver sendo executado. Ex. Variável i do método limpaTela.

Retorno de valores

Alguns métodos realizam operações para calcular um resultado e necessitam passar este resultado de volta para o programa. Esta passagem se chama retorno. Ex:

Um método que calcula o valor da comissão de um vendedor de um produto.

```
public static double comissao (double perc, int quantidade, double valorProduto)
{
      double valorComissao = (valorProduto * perc * quantidade)/100;
      return valorComissao;
}
```

Usando Métodos para melhorar seu programa

Quando você estiver escrevendo seu programa e notar que:
- uma determinada funcionalidade é utilizada várias vezes em seu programa;
- trechos de código se repetem;
- trechos de programas que você já escreveu são usados novamente

Você deve tentar escrever métodos que satisfaçam a funcionalidade ou operação que é repetida.

Construa um programa que calcule e imprima o valor das comissões relativas às vendas de televisores de 30 funcionários. O valor da comissão é dado pela tabela abaixo:

Quantidade de televisores vendidos	***Porcentagem de comissão por aparelho***
Menor ou igual a 10	10%

Quantidade de televisores vendidos	*Porcentagem de comissão por aparelho*
Entre 11 e 29	12%
Maior ou igual a 30	15%

O programa deverá ler a quantidade vendida de cada funcionário e calcular e imprimir o valor devido.

```
import java.util.*
public class Vendas
{
	public static void main(String args[])
	{
		int qtde;
		double percentagem;
		double produto = 100.0;
		double valcomissao;
		Scanner teclado = new Scanner(System.in);
		for (int i = 1; i <30; i++)
		{
			System.out.println("digite quantidade vendida");
			qtde = teclado.nextInt();
			if (qtde < 10)
				valcomissao = (produto * 10 * qtde)/100;
			else if (qtde >=30)
				valcomissao = (produto * 15 * qtde)/100;
			else
				valcomissao = (produto * 12 * qtde)/100;
		}
		System.out.println("O valor da comissao e "+valcomissao);
	}
}
```

Note que o calculo da comissão é repetido diversas vezes. Vamos supor que seja preciso mudar o calculo. Será necessário mudar em todas as vezes. Mas se transformarmos este calculo em um método, quando precisarmos mudar é só mudar o método.

```
import java.util.*
public class Vendas
{
	public static void main(String args[])
	{
		int qtde;
		double percentagem;
		double produto = 100.0;
		double valcomissao;
		Scanner teclado = new Scanner(System.in);
		for (int i = 1; i <30; i++)
		{
			System.out.println("digite quantidade vendida");
			qtde = teclado.nextInt();
			if (qtde < 10)
				valcomissao = comissao(10, qtde, produto);
			else if (qtde >=30)
				valcomissao = comissao(15, qtde, produto);
			else
				valcomissao = comissao(12, qtde, produto);
		}
		System.out.println("O valor da comissao e "+valcomissao);
	}

	public static double comissao (double perc, int quantidade, double valorProduto)
	{
		double valorComissao = (valorProduto * perc * quantidade)/100;
		return valorComissao;
	}
}
```

Usando objetos.

Até agora vimos como criar métodos da mesma classe que continha o programa principal. Agora vamos ver como iremos usar métodos de outras classes no nosso programa.

Vamos fazer um programa que precise ler vários números do teclado. Um Inteiro, um double e um float. Para isso vamos precisar de métodos que nos ajudem nesta leitura. A classe Leitor, possui os seguintes métodos

public float leNumeroFloat()
public int leNumeroInt()
public double leNumeroDouble()
public String leString()

Esta classe será usada para resolver nosso problema desta forma:

```
class Exemplo0508
{
    public static void main (String args[])
    {
        System.out.println("Entre com uma nota de prova");
        float n;
        Leitor leitor = new Leitor();

        n=leitor.leNumeroFloat();
        System.out.println("Nota digitada = "+ n);
        System.out.println("Entre com uma idade");
        int i;
        i=leitor.leNumeroInt();
        System.out.println("Idade digitada = "+ i);

        System.out.println("Entre com um nome");
        System.out.println(leitor.leString());
    }
}
```

Usando métodos de outras classes através de objetos.

Temos uma classe chamada SimpleDateFormat da API (Application Programmer Interface) do Java

Esta classe possui alguns métodos, entre eles:

public String format(Date date)

Como o método não é static, para chamar este método é necessário a criação de um objeto da classe e a chamada através de um objeto. Ex:

```
import java.text.*;
import java.util.*;
public class Hora
{
   public static void main (String args[])
   {
      SimpleDateFormat formatter = new SimpleDateFormat ();
      String dateString = formatter.format(new Date());
      System.out.println("a hora é "+ dateString);
   }
}
```

Objetos

Um objeto é um conceito fundamental na programação em Java. Basicamente, um objeto é uma representação de entidades do mundo real no seu programa. No mundo real temos coisas como cadeiras, automóveis, computadores e também temos pessoas.

Para representarmos estas entidades do mundo real vamos ter que definir um modelo para estes objetos com os detalhes que nos interessam. No caso de uma pessoa podemos querer representar dados como nome, endereço, telefone, e-mail, e podemos não querer representar outros dados como cor dos olhos, cor dos cabelos, idade e tipo sanguíneo.
Após termos determinado os detalhes que nos interessam vamos definir o modelo dos objetos pessoas, que é a Classe Pessoa.

```
class Pessoa
{
   String nome;
   String endereco;
   String telefone;
   String email;
}
```

Nome, endereço, telefone e email são as características das pessoas. Por exemplo:

A Maria terá
nome = “Maria”
endereco = “rua XV, 123”
telefone = “234-5678”
email = “maria@spet.br”

O João terá valores diferentes para as características:

nome = “João”
endereco = “av. trinta, 987”
telefone = “222-2222”
email = “joao@spet.br”

Ambos João e Maria são pessoas. Então dizemos que Maria e João são instancias da classe pessoa e nome, endereco, email e telefone são atributos dos objetos da classe pessoa.

Vamos usar os objetos da classe pessoa.

```
Public class Exemplo
{
       public static void main(String args[])
       {
              Pessoa professor = new Pessoa();
              professor.nome = “Andrey”;
              professor.endereco = “rua X, 1234”;
              professor.email = “andrey@spet.br”;
              professor.telefone = “333-4900”;

              System.out.println(professor.nome);
              System.out.println(professor.endereco);
              System.out.println(professor.email);
              System.out.println (professor.telefone);
       }
}
```

Objetos são variáveis que você pode usar no seu programa. Você deve declarar e instanciar (criar) o objeto.

Pessoa professor = new Pessoa();

e usando o nome da variável (objeto) e o nome do atributo você consegue acessar o valor das características do objeto, tanto para gravação quanto para leitura.

professor.email = “andrey@spet.br”;

Criação de classes

As classes são modelos a partir dos quais podem ser criados objetos. Os objetos são compostos basicamente de:

Atributos (variáveis de instância)
Métodos (operações)

Na aula passada vimos uma classe (Pessoa) que só possuia atributos. Existem classes que só possuem metodos. Vamos ver uma classe que possue métodos e atributos.

```
class Produto
{
   int codigo;
   String nome;
   int quantidade;
   double preco;

   public void mostraPreco()
   {
      System.out.println(preco);
   }

   public void atualizaPreco(double perc)
   {
      preco += preco*perc/100;
   }
}
```

Para usar a classe Produto temos que criar alguns objetos.

```
class Usaproduto2
{
   public static void main (String args[])
   {
      Produto A,B,C;
      A = new Produto();
      B = new Produto();
      C = new Produto();

      A.codigo = 1;
      System.out.println("Valor de codigo em A : " + A.codigo);
      System.out.println("Valor de codigo em B : " + B.codigo);
      B.codigo = 2;
      System.out.println("Valor de codigo em A : " + A.codigo);
      System.out.println("Valor de codigo em B : " + B.codigo);
      System.out.println("Valor de codigo em C : " + C.codigo);
   }
}
```

Encapsulamento

Quando se usa uma classe, devemos ter o cuidado de não acessar diretamente os atributos de um objeto. No programa acima o comando:

```
A.codigo = 1;
```

pode ser muito prejudicial. Por exemplo, se precisarmos mudar o atributo código para String (para adicionar um dígito verificador) teremos que mudar todas as partes do programa que acessam esta variável. A saída é criarmos métodos de acesso aos atributos e não permitirmos mais o acesso direto aos atributos. São os métodos get e set.

```
class Produto
{
   int codigo;
   String nome;
   int quantidade;
   double preco;

   public void setCodigo(int codigo)
   {
      this.codigo = codigo;
   }

   public int getCodigo()
   {
      return this.codigo;
   }

   public void mostraPreco()
   {
      System.out.println(preco);
   }

   public void atualizaPreco(double perc)
   {
      preco += preco*perc/100;
   }
}
```

No programa que usa a classe Produto o acesso aos atributos ficaria assim:

em vez de **`A.codigo = 1;`** ficaria **`A.setCodigo(1);`**

e em vez de **`System.out.println("Valor de codigo em A : " + A.codigo);`**
ficaria **`System.out.println("Valor de codigo em A : " + A.getCodigo());`**

Visibilidade

A visibilidade dos atributos e métodos de uma classe pode ser alterada. A visibilidade é um modificador que restringe o acesso das outras classes sobre o atributo ou método. Os tipos de visibilidade em Java são:

public	todas as classes podem acessar e modificar o atributo e executar o método.
protected	Apenas as subclasses e as classes do pacote podem acessar e modificar o atributo e executar o método.
private	Apenas a própria classe pode acessar e modificar o atributo e executar o método.

Para favorecer o Encapsulamento e deixar as classes menos acopladas (dependentes), uma boa prática é sempre deixar os atributos **private** (ou no máximo **protected**), **nunca public**.

Referencia This

As vezes quando estamos construindo um método usamos o mesmo nome de um atributo da classe para uma variável local. Para diferenciar as duas usamos a referencia this. This é uma referencia para o próprio objeto.

```
public void setCodigo(int codigo)
{
   this.codigo = codigo;
}
```

Sobrecarga de métodos

Uma classe pode métodos com o mesmo nome. Por exemplo:

```
public void setProduto(int codigo)
public void setProduto(int codigo, String nome)
public void setProduto(int codigo, double preco)
```

O nome pode ser o mesmo, mas a assinatura não pode ser totalmente igual. No caso acima os argumentos são diferentes. Podemos também mudar o tipo de retorno. Escrever mais de um método com o mesmo nome é chamado de **sobrecarga de método.**

Métodos Construtores

Quando nós criamos um objeto precisamos chamar o operador **new** . Este operador chama o construtor padrão das classes do java. Este construtor é responsável por alocar espaço na memória para o objeto e criar os atributos com valor nulo.

Se quizermos que, quando o objeto for criado ele já possua valores nos seus atributos podemos criar outros construtores (sobrecarga). Ex: Para a classe Produto

```
public Produto()
{
}

public Produto(int codigo, String nome, int quantidade, double preco)
{
   this.codigo= codigo;
   this.nome = nome;
   this.quantidade = quantidade;
   this.preco = perco;
}
```

Quando criamos um outro construtor precisamos declarar, também o construtor padrão.

INTEGRAÇÃO:

1) Usando as classes Leitor e Pessoa, apresentadas nesta unidade construa um programa que leia os dados de 4 pessoas guarde nos objetos e os imprima.

2) Usando a classe Arquivo e o método public void gravar(String frase, String nomeArquivo), construa um programa que leia os dados de 4 pessoas e grave em um arquivo texto.

3) Construa um método que receba uma temperatura em graus Celsius e calcule e retorne o valor da temperatura equivalente em graus Farenheit. Celsius = 5.0/9.0 (farenheit -32).

4) Modifique o programa do cálculo das comissões para que se o valor da comissão ultrapassar 1000 reais o vendedor receba mais 2% sobre o valor total.

5) Construa um método que receba uma String e imprima esta string invertida.

6) Escreva os métodos get e set para os outros atributos da classe Produto, e coloque a visibilidade apropriada para os atributos.

7) Usando a classe Arquivo e o método public void gravar(String frase, String nomeArquivo), escreva para a classe Pessoa os métodos de acesso (get e set) além de um método para gravar os dados da pessoa. Altere o programa que leia os dados de 4 pessoas e grave em um arquivo texto.

8) Crie construtores para as classes Pessoa e Produto.

BIBLIOGRAFIA:

FURGERI, SÉRGIO. *Java 2 Ensino Didático. Desenvolvendo e Implementando Aplicações*. ed. Érica. São Paulo, 2002.
DEITEL, H. M. e DEITEL, P. J.. Java, como Programar. Ed. Bookman. Porto Alegre. 2001.
ARNOLD Ken, GOSLING James: "The Java Programming Language Second Edition", Addison-Wesley, 1997.