

```

import java.util.Scanner;
class Data { ... }
class Compromisso { ... }
public class Agenda {
    private Compromisso[] compromissos;
    private Compromisso[] realoqueCompromissos (Compromisso[] vetor,
int tamanhoDesejado) {
        if (vetor==null) {
            vetor = new Compromisso[tamanhoDesejado];
            return vetor;
        } else
            if (tamanhoDesejado==0)
                return null;
            else {
                Compromisso[] vetorAux = new
Compromisso[tamanhoDesejado];
                if (tamanhoDesejado>=vetor.length)
                    for (int i=0; i<vetor.length; i++)
                        vetorAux[i] = vetor[i];
                else
                    for (int i=0; i<tamanhoDesejado; i++)
                        vetorAux[i] = vetor[i];
                return vetorAux;
            }
    }
}

```

```

public Agenda()
{
    compromissos = null;
}
public int getNumeroDeCompromissos()
{
    if (compromissos==null)
        return 0;
    else
        return compromissos.length;
}
public void insereCompromisso(Compromisso c) {
    compromissos = realoqueCompromissos (compromissos,
getNumeroDeCompromissos() + 1);
    compromissos[compromissos.length-1] = c;
}
public boolean removeCompromisso(Data d, int h)
{
    if (getNumeroDeCompromissos(>0)
        for (int i=0; i<compromissos.length; i++)
            if ((compromissos[i].getData().getDia()==d.getDia() &&
compromissos[i].getData().getMes()==d.getMes() &&
compromissos[i].getData().getAno()==d.getAno()) &&
compromissos[i].getHora()==h)

```

```

    {
        compromissos[i]=compromissos[compromissos.length-1];
        compromissos = realoqueCompromissos(compromissos,
compromissos.length-1);
        if (compromissos==null && getNumeroDeCompromissos() >0)
        { /* exemplo da utilização da exit() */
            System.out.print ("\nERRO!\n");
            System.exit(1);
        }
        return true;
    }
    return false;
}
public boolean existeDisponibilidade(Data d, int h)
{
    if (getNumeroDeCompromissos())>0)
        for (int i=0; i<compromissos.length; i++)
            if ((compromissos[i].getData().getDia()==d.getDia()) &&
compromissos[i].getData().getMes()==d.getMes() &&
compromissos[i].getData().getAno()==d.getAno()) &&
compromissos[i].getHora()==h)
                return false;
    return true;
}

```

```

public boolean ocorrenciaDoCompromisso(String desc, Data d, int[] h)
{
    if (getNumeroDeCompromissos(>0)
        for (int i=0; i<compromissos.length; i++)
            if (compromissos[i].getDescricao() == desc) /** Comparação
inadequada, utilizar compromissos[i].getDescricao().equals(desc)*/
            {
                d = compromissos[i].getData();
                h[0] = compromissos[i].getHora();
                return true;
            }
    return false;
}
public void listeCompromissos()
{
    if (getNumeroDeCompromissos(>0)
        for (Compromisso c:compromissos)
        {
            c.apresenta();
        }
    else
        System.out.println("\nNenhum compromisso cadastrado.");
}
}

```

```

static private void menu()
{
    System.out.println ("\nEntre com sua opcao: ");
    System.out.println ("1- Inserir compromisso");
    System.out.println ("2- Excluir compromisso (com base em seu
horario e data de ocorrencia)");
    System.out.println ("3- Consultar a disponibilidade de um
determinado horario em um determinado dia");
    System.out.println ("4- Consultar o horario e a data da ocorrencia de
um compromisso");
    System.out.println ("5- Imprimir o conjunto de compromissos");
    System.out.println ("6- Sair do programa");
    System.out.println ("Opcao = ");
}
static public void main(String args[])
{
    Agenda agenda = new Agenda();
    int opcao;
    Scanner entrada = new Scanner(System.in);
    do
    {
        menu();
        opcao = entrada.nextInt();
    }
}

```

```

switch (opcao) {
    case 1:
    {
        String descricao;
        int aux1, aux2, aux3;
        System.out.println("\nEntre com as informacoes sobre o
compromisso:");
        System.out.print("Descricao: ");
        entrada.nextLine();
        descricao = entrada.nextLine();
        System.out.print("Data:\nDia -> ");
        aux1 = entrada.nextInt();
        System.out.print("Mes -> ");
        aux2 = entrada.nextInt();
        System.out.print("Ano -> ");
        aux3 = entrada.nextInt();
        Data data = new Data(aux1, aux2, aux3);
        System.out.print("Horario:\nHora -> ");
        aux1 = entrada.nextInt();
        Compromisso compromisso = new Compromisso(data, aux1,
descricao);
        agenda.insereCompromisso(compromisso);
        break;
    }
}

```

```

case 2:
{
    int aux1, aux2, aux3;
    System.out.println("Entre com as informacoes sobre o
compromisso a ser excluido:");
    System.out.print("Data: \nDia -> ");
    aux1 = entrada.nextInt();
    System.out.print("Mes -> ");
    aux2 = entrada.nextInt();
    System.out.print("Ano -> ");
    aux3 = entrada.nextInt();
    Data data = new Data(aux1, aux2, aux3);
    System.out.print("Horario: \nHora -> ");
    aux1 = entrada.nextInt();
    if (agenda.removeCompromisso(data, aux1))
        System.out.print("\nCompromisso excluido.");
    else
        System.out.print("\nCompromisso nao existente.");
    break;
}
case 3:
{
    int aux1, aux2, aux3;

```

```

    System.out.println("Entre com as informacoes necessarias para
efetuar a operacao:");
    System.out.print("Data: \nDia -> ");
    aux1 = entrada.nextInt();
    System.out.print("Mes -> ");
    aux2 = entrada.nextInt();
    System.out.print("Ano -> ");
    aux3 = entrada.nextInt();
    Data data = new Data(aux1, aux2, aux3);
    System.out.print("Horario: \nHora -> ");
    aux1 = entrada.nextInt();
    if (agenda.existeDisponibilidade(data, aux1))
        System.out.println("\nHorario disponivel.");
    else
        System.out.println("\nHorario indisponivel.");
    break;
}
case 4:
{
    String descricao;
    Data data = new Data(); /* ou apenas "Data data;" */
    int[] hora = new int[1];
    System.out.println("Entre com a descricao do compromisso:");

```



```

descricao = entrada.nextLine();
if (agenda.ocorrenciaDoCompromisso(descricao, data, hora)) {
    System.out.print("O compromisso ocorrera em ");
    data.apresenta();
    System.out.printf(" as %d.", hora);
}
else
    System.out.println("\nCompromisso nao agendado.");
break;
}
case 5: {
    agenda.listeCompromissos();
    break;
}
case 6: {
    System.out.println("\nObrigado por utilizar nosso software.");
    break;
}
default:
    System.out.println("\nOpcao invalida!");
}
}while (opcao!=6);

```

Linguagem de Programação Java

Recebendo argumentos da linha de comando

Assim como em outras linguagens Java possibilita o recebimento de argumentos através da linha de comando.

Isto ocorre através do parâmetro da função main. Ou seja, através do vetor de String que até o momento apenas inserimos como parâmetro da função estática main.

Vamos analisar um exemplo, para obtermos uma compreensão melhor.

```

public class VetorDeInteiros
{
    public static void main(String args[])
    {
        if (args.length != 3)/* verifica número de argumentos de linha de
comando */
            System.out.println( "Erro: Por favor execute novamente o " +
" programa e forneça tres valores inteiros");
        else
        {
            int tamanhoVetor = Integer.parseInt(args[0]); /* obtém o tamanho do
vetor a partir do primeiro argumento da linha de comando */
            int vetor[] = new int[tamanhoVetor]; // cria o vetor
            int valorInicial = Integer.parseInt(args[1]); /* obtém o valor inicial e
o incremento a partir dos argumentos da linha de comando */
            int incremento = Integer.parseInt(args[2]);
            for (int contador=0; contador<vetor.length; contador++) /* calcula o
valor de cada elemento do vetor */
                vetor[contador] = valorInicial + incremento * contador;
            System.out.printf( "%s%8s\n", "Index", "Value" );
            for (int contador=0; contador<vetor.length; contador++) /* exhibe o
valor e o índice do vetor */
                System.out.printf( "%5d%8d\n", contador, vetor[contador]);
        }
    }
}

```

Linguagem de Programação Java

Definição de pacotes

Visando tornar uma classe reutilizável, ou seja, possibilitar que a mesma seja importada por outros aplicativos, devemos colocá-la em um pacote.

Para declararmos um pacote utilizamos a instrução ***package***.

Visando possibilitar a definição de nomes únicos para cada pacote, a Sun especificou uma convenção para a nomeação de pacotes.

Cada nome de pacote deve começar com seu nome de domínio Internet na ordem inversa.

Como exemplo, podemos utilizar o nome de domínio de nossa universidade, ou seja, nossos pacotes começariam com edu.univasf.

Depois que o nome de domínio é invertido pode-se determinar qualquer nome para o pacote.

Linguagem de Programação Java

Definição de pacotes

Para compilar uma classe empacotada utilizamos, por exemplo, a linha

```
javac -d . Exemplo.java
```

Vamos ver um exemplo:

```
package edu.univasf.poo;  
public class Horário2  
{  
    private int hora;  
    private int minuto;  
    private int segundo;  
    public Horário2()  
    {  
        this( 0, 0, 0 );  
    }  
    public Horário2( int h, int m, int s )  
    {  
        setHorario( h, m, s );  
    }  
}
```

```

public Horario2( Horario2 horario )
{
    this( horario.getHora(), horario.getMinuto(), horario.getSegundo() );
}
public void setHorario( int h, int m, int s )
{
    hora = ( ( h >= 0 && h < 24 ) ? h : 0 );
    minuto = ( ( m >= 0 && m < 60 ) ? m : 0 );
    segundo = ( ( s >= 0 && s < 60 ) ? s : 0 );
}
public int getHora()
{
    return hora;
}
public int getMinuto()
{
    return minuto;
}
public int getSegundo()
{
    return segundo;
}
public String toString()
{
    return String.format( "\n%02d:%02d:%02d", hora, minuto, segundo );
}
}

```

Linguagem de Programação Java

Observações relevantes sobre o exemplo anterior

Só é possível invocar um construtor sobrecarregado através do *this()* em outro construtor. Sendo que, esta invocação deve ser a primeira instrução do construtor.

Quando um objeto de uma classe contém uma referência a outro objeto da mesma classe, o primeiro objeto pode acessar todos os dados e métodos do segundo objeto (incluindo aqueles que são *private*).

Logo, poderíamos reescrever o método construtor `Horario2(Horario2)` da seguinte maneira:

```
public Horario2( Horario2 horario )  
{  
    this(horario.hora, horario.minuto, horario.segundo);  
}
```

Linguagem de Programação Java

Observações relevantes sobre o exemplo anterior

Outro detalhe importante é a possibilidade de se converter os atributos a serem impressos na saída para uma `String` com o `String.format(stringDeControle, listaDeParametros)`; em um método denominado `toString` e assim poder passar o objeto diretamente para o método `System.out.print()`; ou `System.out.println()`;

Um detalhe importante, refere-se ao fato de em Java se não especificamos um modificador de tipo de acesso para uma variável ou método, este terá acesso de pacote, ou seja, será acessível a todas as classes do pacote.

Linguagem de Programação Java

Por fim, vamos tratar da importação de uma classe reutilizável.

A importação de pacote é necessária quando a classe que vai se utilizar da outra está em um pacote (diretório) diferente.

Se tivermos a intenção de utilizar a classe `Horario2` em um aplicativo cujo arquivo não encontra-se no mesmo diretório que o da classe `Horario2` devemos inserir a seguinte linha de código no início do arquivo do aplicativo em questão:

```
import edu.univasf.poo.Horario2;
```

No caso em que exista a intenção de se utilizar mais de uma classe do pacote `edu.univasf.poo`, devemos utilizar a seguinte linha:

```
import edu.univasf.poo.*;
```

```
import edu.univasf.poo.Horario2;

public class TesteHorario2
{
    public static void main(String args[])
    {
        Horario2 h = new Horario2(12,20,3);

        System.out.print(h);

    }
}
```

Linguagem de Programação Java

Exercício: O serviço de correio expresso como o SEDEX®, oferece uma opção de entrega de pacotes. Explore os conceitos estudados na aula de hoje, declarando uma classe Pacote que inclua variáveis de instância para representar nome, endereço, cidade e CEP tanto do remetente quanto do destinatário do pacote, além de uma variável de instância que armazene o peso (em quilos) do pacote e da variável de classe que armazene o custo por quilo para entrega do pacote. Assegure que o peso e o custo por quilo contêm valores positivos. A classe Pacote deve fornecer um método de instância calculaCusto que retorna um double indicando o custo associado com a entrega do pacote. A função calculaCusto de Pacote deve determinar o custo multiplicando o peso pelo custo (em quilos). Construa uma classe que se utilize da classe Pacote e que esteja armazenada fisicamente em um diretório distinto.