```
class Data
{
  ...
}
public class Compromisso
{
  private Data data;
  private int hora;
  private String descricao; /**java.lang.String*/
  public Compromisso()
  {
  }
  public Compromisso(Data data, int hora, String descricao)
  {
    this.data = data; //palavra reservada this
    this.hora = hora;
    this.descricao = descricao;
  }
  public void setData(Data d)
  {
    data = d;
  }
```

```
public void setHora(int h)
{
   hora = ((h>=0 && h<24)?h:0);
}
void setDescricao(String desc)
{
   descricao = desc;
}
Data getData()
{
   return data;
}
int getHora()
{
   return hora;
}
String getDescricao()
{
   return descricao;
}
```

```
  void apresenta() {
        System.out.printf("\nInformacoes sobre o compromisso:
%s\nHorario: %d", getDescricao() ,getHora());
    System.out.print ("\nData: ");
    data.apresenta();
  }
  static public void main(String args[])
  {
        Compromisso c = new Compromisso();
        Data d = new Data(01, 06, 2010);
        d.apresenta();
        Compromisso c2 = new Compromisso(d, 10,
        "Ministrar aula de POO");
        c2.apresenta();
  }
}
```

ALOCAÇÃO DINÂMICA

A melhor forma de entender a alocação dinâmica de memória em Java é através da análise de um exemplo.

```
import java.util.Scanner;
public class AlocacaoDinamica
{
  static public void main (String args[])
  {
    Scanner input = new Scanner(System.in);
    int vetor[] = null, numero;
    do
    {
      System.out.print("Forneca um natural para compor o vetor");
      System.out.print(" ou um negativo para finalizar o ");
      System.out.print("preenchimento do vetor: ");
      numero = input.nextInt();
```

```
        if (numero>=0) {
            if (vetor == null) { /**não é possível substituir esta expressão lógica
por "if (!vetor) {" pois vetor não é uma variável boolean*/
              vetor = new int[1];
              vetor[0] = numero;
            }
            else
            {
             int[] aux = new int[vetor.length+1];
             for (int i=0; i<vetor.length; i++)
               aux[i] = vetor[i];
             aux[aux.length-1] = numero;
             vetor = aux;
             System.gc(); // "pede" que a coleta de lixo ocorra neste ponto
          }
        }
      }while(numero>=0);
      System.out.println("\nOs elementos do vetor sao:");
      for (int aux:vetor)
        System.out.println(aux);
    }
}
```

VARIÁVEIS E MÉTODOS DE CLASSE

Assim como vimos na linguagem C++ na linguagem Java cada objeto de uma classe possui sua cópia das variáveis de instância definidas na classe.

Java também possibilita que, em certos casos, apenas uma cópia de certo atributo seja compartilhada por todos os objetos da classe.

Denominado este tipo de atributo como variável de classe e assim como em C++ o especificamos com a palavra reservada *static*.

Da mesma forma que podemos definir variáveis de classe também podemos definir métodos de classe.

Analisaremos agora um exemplo da declaração da classe Ponto2D utilizando-se de variáveis e métodos de classe.

```
public class Ponto2D
{
  private float x;
  private float y;
  private static int numeroDePontosInstanciados = 1; /** variável de
classe */
  /* ... */
  public static void setNumeroDePontosInstanciados(int
        novoNumeroDePontosInstanciados) /** método de classe */
  {
    numeroDePontosInstanciados =
        novoNumeroDePontosInstanciados;
  }
  public static int getNumeroDePontosInstanciados() /** método de
classe */
  {
    return numeroDePontosInstanciados;
  }
}
```

ALOCAÇÃO DINÂMICA - Retomada

Para uma visualização mais profunda de como ocorre o processo de coleta de lixo analisaremos mais um exemplo.

```
public class Classe
{
  private static int numeroDeObjetosInstanciados = 0; /** variável de classe */
  public static void setNumeroDeObjetosInstanciados(int novoNumeroDeObjetosInstanciados) /** método de classe */
  {
    numeroDeObjetosInstanciados = novoNumeroDeObjetosInstanciados;
  }
  public static int getNumeroDeObjetosInstanciados() /** método de classe */
  {
    return numeroDeObjetosInstanciados;
  }
```

```
  public Classe()
  {

setNumeroDeObjetosInstanciados(getNumeroDeObjetosInstanciado
s()+1);
  }
  protected void finalize() /**Método utilizado pela coleta de lixo*/
  {

setNumeroDeObjetosInstanciados(getNumeroDeObjetosInstanciado
s()-1);
            System.out.println("O metodo finalize() foi executado.");
  }
```

```
  static public void main(String args[])
  {
    System.out.println("Numero inicial de objetos: " +
Classe.getNumeroDeObjetosInstanciados());
        Classe c1 = new Classe();
        System.out.println("Numero atual de objetos: " +
Classe.getNumeroDeObjetosInstanciados());
        c1 = null;
        System.gc();
        System.out.println("Numero atual de objetos: " +
Classe.getNumeroDeObjetosInstanciados());
  }
}
```

VETORES MULTIDIMENSIONAIS

Java não suporta vetores multidimensionais diretamente, mas permite que os programadores especifiquem vetores unidimensionais cujos elementos também são vetores unidimensionais.

Esta característica permite que sejam declarados/instanciados vetores bidimensionais cujas linhas possuem números de colunas distintos.

Por exemplo:

```
int matriz1[][], matriz2[][];
matriz1 = new int[4][7]; /* cria um vetor bidimensional com 4 linhas e 7 colunas por linha */
matriz2 = new int[2][]; //cria duas linhas
matriz2[0] = new int[5]; //determina cinco colunas
matriz2[1] = new int[3]; //determina três colunas
```

VETORES MULTIDIMENSIONAIS

Analisaremos agora um exemplo mais completo:

```
public class InicializandoVetores {
   public static void main( String args[] ) {
      int vetor1[][] = { { 1, 2, 3 }, { 4, 5, 6 } };
      int vetor2[][] = { { 1, 2 }, { 3 }, { 4, 5, 6 } };
      System.out.println( "Valores dos elementos do vetor1:" );
      apresentaVetor( vetor1 );
      System.out.println( "\nValores dos elementos do vetor2:" );
      apresentaVetor( vetor2 );
   }
   public static void apresentaVetor(int vetor[][]) {
      for ( int linha = 0; linha < vetor.length; linha++ ) {
         for ( int coluna = 0; coluna < vetor[linha].length; coluna++ )
            System.out.printf( "%d  ", vetor[linha][coluna] );
         System.out.println();
      }
   }
}
```

CLASSE Math

A classe Math disponibiliza vários métodos matemáticos como:

Método	Descrição
abs(x)	valor absoluto de x
ceil(x)	arredonda x para o menor inteiro não menor que x
cos(x)	cosseno trigonométrico de x (x em radianos)
exp(x)	método exponencial e^x
floor(x)	arredonda x para o maior inteiro não maior que x
log(x)	logaritmo natural de x (base e)
max(x, y)	maior valor de x e y

CLASSE Math

Método	Descrição
min(x, y)	menor valor de x e y
pow(x, y)	x elevado à potência de y (isto é, x^y)
sin(x)	seno trigonométrico de x (x em radianos)
sqrt(x)	raiz quadrada de x
tan(x)	tangente trigonométrica de x (x em radianos)

A classe também declara duas constantes matemáticas:

Math.PI (3,14159265358979323846)

Math.E (2,718281828459045235 4), base para logaritmos naturais.

Exercício: Com base no que vimos declare uma classe Agenda que será uma composição de Compromisso.

Implemente um método que instancie um objeto da classe Agenda e explore seus comportamentos, através de um menu, exibido ao usuário, com as seguintes funcionalidades:

- Inserir compromisso;
- Excluir compromisso (com base em sua hora e data de ocorrência);
- Consultar a disponibilidade de uma determinada hora em um determinado dia;
- Consultar a hora e a data da ocorrência de um compromisso;
- Apresentar o conjunto de compromissos

Observação: Considerar que os compromissos agendados têm duração constante de uma hora. Caso julgue necessário, efetue as devidas adaptações nas classes Data e Compromisso definidas anteriormente.