CLASSES (tipos por referência)

Como os conceitos da OO já foram trabalhados em UML e implementados na linguagem C++ otimizaremos nosso estudo de Java, apenas demonstrando como os conceitos OO são implementados em Java e tratando de algumas particularidades desta linguagem.

Vamos analisar o exemplo da declaração da classe Ponto2D:

```
import java.util.Scanner;
public class Ponto2D /** classe que dará nome ao arquivo*/
{
  private float x; /** variável de instância*/
  private float y; /** variável de instância*/
  public void setX (float novoX) /** método de instância*/
  {
    x = novoX;
  }
  public float getX () /** método de instância*/
  {
    return x;
  }
  public void setY (float novoY) /** método de instância*/
  {
    y = novoY;
  }
  public float getY () /** método de instância*/
  {
    return y;
  }
  public void apresentaCoordenadas() /** método de instância*/
  {
    System.out.printf("\n(%f, %f)\n", getX(), getY());
  }
```

```
    public static void main(String args[])
    {
      Scanner input = new Scanner(System.in);
      float x, y;
      Ponto2D p = new Ponto2D(); //instanciação de um objeto
      System.out.println("Coordenadas iniciais do ponto:");
      p.apresentaCoordenadas(); //invocação de método
      System.out.print("Forneca a coordenada x do ponto: ");
      x = input.nextFloat();
      p.setX(x); //invocação de método
      System.out.print("Forneca a coordenada y do ponto: ");
      y = input.nextFloat();
      p.setY(y); //invocação de método
      p.apresentaCoordenadas(); //invocação de método
    }
}
```

```
public class Ponto2D /** classe Ponto2D com construtor */
{
   private float x;
   private float y;
   public Ponto2D() /** método construtor */
   {
      x = 1;
      y = 1;
   }
   public void setX (float novoX)
   {
      x = novoX;
   }
   public float getX ()
   {
      return x;
   }
   public void setY (float novoY)
   {
      y = novoY;
   }
```

```
  public float getY ()
  {
    return y;
  }
  public void apresentaCoordenadas() /** utilização do print para
imprimir valores não literais */
  {
    System.out.print ("\n(");
    System.out.print (getX());
    System.out.print (", ");
    System.out.print (getY());
    System.out.println (")");
  }
  public void apresentaCoordenadas2() /** utilização do print para
imprimir valores não literais */
  {
    System.out.println ("\n(" + getX() + ", " + getY() + ")");
  }
}
```

```
import java.util.Scanner;
public class Ponto2D
{
   private float x;
   private float y;
   public Ponto2D(float novoX, float novoY) /** método construtor */
   {
      x = novoX;
      y = novoY;
   }
   public void setX (float novoX)
   {/*...*/}
   public float getX ()
   {/*...*/}
   public void setY (float novoY)
   {/*...*/}
   public float getY ()
   {/*...*/}
   public void apresentaCoordenadas()
   {
      System.out.printf("\n(%f, %f)\n", getX(), getY());
   }
```

```
  public static void main(String args[])
  {
    Scanner input = new Scanner(System.in);
    float x, y;
    Ponto2D p = new Ponto2D(); /* ERRO! Instanciação de um objeto inadequada */
    Ponto2D p = new Ponto2D(5.2f, 7.3f); //instanciação de um objeto
    System.out.println("Coordenadas iniciais do ponto:");
    p.apresentaCoordenadas(); //invocação de método
    System.out.print("Forneca a coordenada x do ponto: ");
    x = input.nextFloat();
    p.setX(x); //invocação de método
    System.out.print("Forneca a coordenada y do ponto: ");
    y = input.nextFloat();
    p.setY(y); //invocação de método
    p.apresentaCoordenadas(); //invocação de método
  }
}
```

Exercício

Declare uma classe Data que possua como variáveis de instância três inteiros representado o dia, o mês e o ano. Implemente métodos de instância que manipulem adequadamente as variáveis de instância existentes.

```
public class Data
{
   private int dia;
   private int mes;
   private int ano;
   public Data(int novoDia, int novoMes, int novoAno)
   {
            setAno(novoAno);
            setMes(novoMes);
            setDia(novoDia);
   }
   public Data() // sobrecarga de método
   {
   }
   public int getDia ()
   {
     return dia;
   }
   public int getMes ()
   {
     return mes;
   }
}
```

```
public int getAno ()
{
   return ano;
}

public void setDia(int d)
{
   dia = verificaDia(d);
}
public void setMes(int m)
{
   if (m>0 && m<=12)
      mes = m;
   else
   {
      mes = 1;
      System.out.println ("Mes invalido (" + m + ") setado para 1.");
   }
}
```

```
public void setAno(int a)
{
  if (a>=1900 && a<2017)
    ano = a;
  else{
    ano = 1900;
    System.out.println ("\nAno invalido (" + a + ") setado para 1900.");
  }
}
int verificaDia(int diaTeste)
{
  final int diasPorMes[] = {0,31,28,31,30,31,30,31,31,30,31,30,31}; //vetor
  if (diaTeste>0 && diaTeste <= diasPorMes[getMes()])
    return diaTeste;
  if (getMes()==2 && diaTeste==29 && (getAno()%400==0 ||
(getAno()%4==0 && getAno()%100!=0)))
    return diaTeste;
  System.out.println ("\nDia invalido (" + diaTeste + ")setado para 1.");
  return 1;
}
```

```
  public static void main(String args[])
  {
    Data d = new Data(11, 06, 2010);
    System.out.printf("\n%d/%d/%d\n",d.getDia(),d.getMes(),d.getAno());
    Data d2 = new Data(25, 12, 2000);
   System.out.println("\n"+d2.getDia()+"/"+d2.getMes()+"/"+d2.getAno());
  }
}
```

VETORES

Como vimos um exemplo de sintaxe para declaração de um vetor e observamos que um vetor pode ser inicializado na declaração.

Porém, também podemos declarar um vetor da seguinte forma:

```
int v[];
float v2[] = new float[10];
v = new int[2];
char[] v3;
```

Um vetor é um objeto e possui uma referência para um elemento de vetor (*array*). Cada vetor possui uma variável de instância pública denominada length que contém o comprimento do vetor.

VETORES

Um detalhe sutil com relação à declaração de vetores é que em

```
int v[], v2, v3;
```

temos apenas o vetor v, e em

```
char[] v4, v5, v6;
```

temos três vetores.

Em Java existe uma forma peculiar da estrutura for, o exemplo a seguir demonstra sua sintaxe e utilização.

```
public class Teste
{
   static public void main (String args[])
   {
      char []v = {'M','a','r','c','e','l','o'};
      System.out.println(v.length);
      for (char aux:v)
         System.out.print(aux);
   }
}
```

Exercício

Declare uma classe Compromisso que possua como variáveis de instância um inteiro representando a hora do compromisso, uma string representando a descrição do compromisso e um objeto da classe Data. Implemente métodos de instância que manipulem adequadamente as variáveis de instância existentes.

```
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);
    }
}
```