ALGORITMOS Y PROGRAMACIÓN

Clase 5

Programación orientada a objetos

Universidad Nacional
ARTURO JAURETCHE

Temario

- Miembros de instancia y de clase
- Composición de objetos

Miembros de una clase.

- En POO hay dos tipos de miembros:
  - De instancia
  - De clase

- Los miembros de instancia son las variables de instancia y los métodos de instancia. Se utilizan para especificar atributos y comportamiento de las instancias u objetos de una clase.

- Los miembros de clase comprenden las variables de clase y los métodos de clase. Se utilizan para especificar características compartidas por todos los objetos de una clase y definir métodos para crear instancias y manipular las variables de clase.

Miembros de una clase.

- En C# si queremos declarar *miembros de clase* usamos la palabra reservada *static.*

```
public static <tipo> unaVariableDeClase;
public static void UnMetodoDeClase() { }
```

Si queremos que los *miembros sean de instancia* no ponemos nada

```
public <tipo> variableDeInstancia;
public void UnMetodoDeInstancia() { }
```

Miembros de instancia

- Los miembros de instancia, ya sean variables o métodos son utilizados cuando se trabaja con instancias u objetos.

```
public class Auto
   {
      private string marca;
      private int modelo;

      /*constructores*/
      public Auto(){  }
      public Auto(string marca, int modelo){
         this.marca = marca;
         this.modelo = modelo; }
      ……
      public void imprimir(){
      Console.WriteLine("Marca " + marca + " modelo " +
      modelo);
   }
}
```

Variables de instancia

Método de instancia

Miembros de instancia

Los *métodos de instancia* se usan para *modificar o consultar el valor de las variables de instancia de un objeto*.

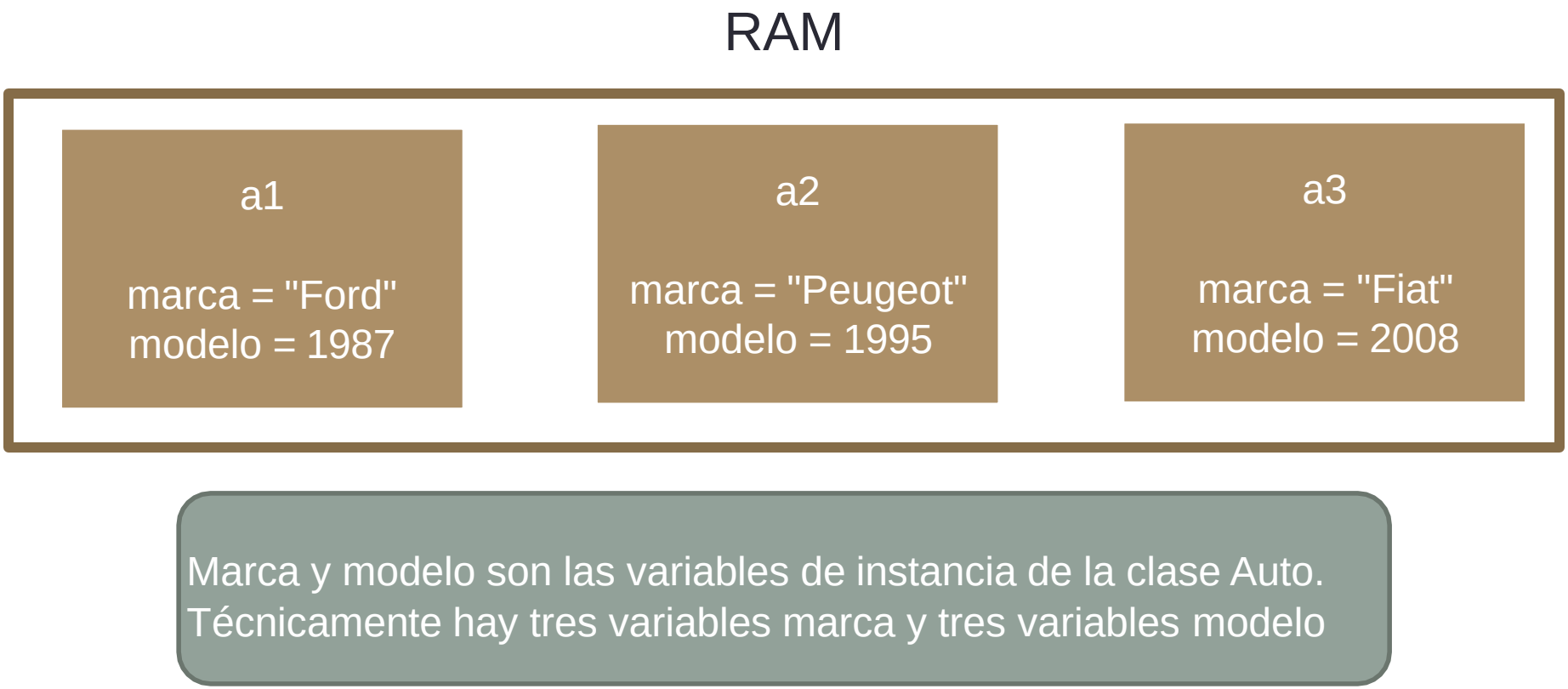
En el siguiente ejemplo creamos objetos o instancias a1,a2 y a3 de la clase Auto:

```
Auto a1 = new Auto("Ford", 1987);
Auto a2 = new Auto("Peugeot", 1995);
Auto a3 = new Auto("Fiat", 2008);

a1.imprimir();  /*se invoca al método de instancia*/
a2.imprimir();
a3.imprimir();
```

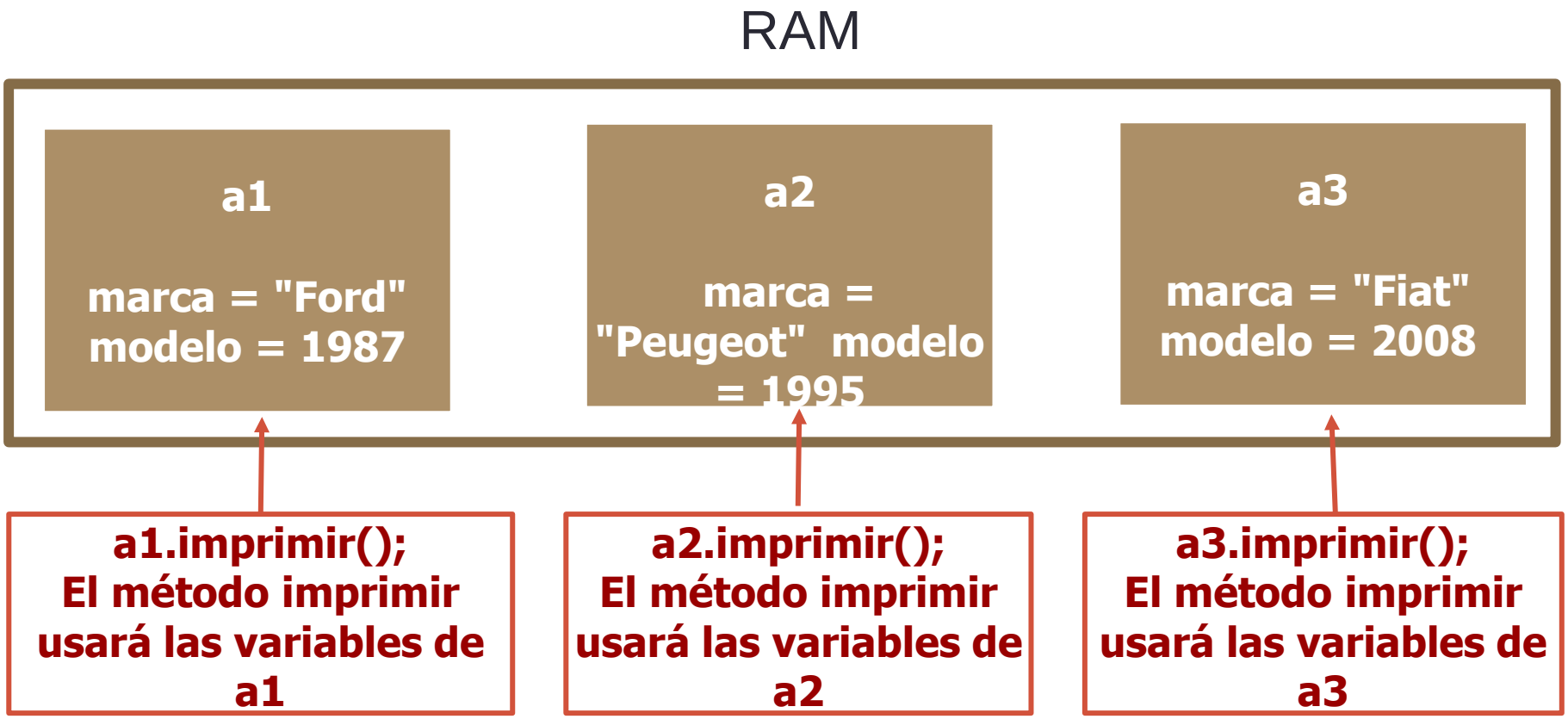
Miembros de instancia

- Cuando se instancia una clase, el compilador genera en memoria una "copia" de la clase para cada instancia creada.



Miembros de instancia

Los métodos de instancia son los que acceden a las variables de instancia, dependiendo de quién sea el objeto receptor del mensaje.



Puesta en común Ejercicio 3 – TP 4

Defina la clase Persona con 3 campos: Nombre, Edad y DNI.
Escriba un programa de aplicación (Main) que permita al usuario ingresar por consola una serie de datos de la forma

Nombre Documento Edad <ENTER> (datos separados por espacios en blanco)

El proceso de entrada finaliza con un string vacío.

Una vez finalizada la entrada de datos, el programa debe imprimir en la consola el listado con la forma:

Nro.) Nombre (Edad) DNI.

Por ejemplo:
1) Juan Perez (40) 2098745
2) José García (41) 1965412

Vamos por partes…

Definamos la clase Persona con 3 campos: nombre, edad y dni

1- Qué tipos de miembros son nombre, edad y dni?

2- ¿Dónde definimos la operación imprimir? Qué tipo de método es?

3- En la aplicación, qué estructura de datos utilizamos para almacenar a las personas?

4- Cómo ingresamos los datos de entrada? (split)

Miembros de clase

- Los miembros de clase, ya sean variables de clase o métodos de clase, no pertenecen a ningún objeto en particular, *son comunes y compartidos por todas las instancias u objetos* que pertenecen a la clase.

- Se declaran con la palabra reservada **static**

- Para utilizar un miembro de clase se lo califica mediante el nombre de la clase seguido del nombre de la variable o del método:

<clase> . <miembro>

Miembros de clase

- Las variables de clase son variables únicas, globales, comunes a todos los objetos. Se almacenan en la clase, no en cada objeto.
- El valor que toma una variable de clase es único o sea es el mismo valor para todos los objetos.
- Los métodos de clase se utilizan para crear objetos, o bien para consultar y modificar variables de clase.
- Los métodos de clase solo pueden ser ejecutados por una clase, no por un objeto.

Miembros de Clase

```
public class Auto {

        private string marca;
        private int modelo;
        public static int impresiones = 0;

        public Auto(string marca, int modelo){
            this.marca = marca;
            this.modelo = modelo;
        }
        public void imprimir(){
              Console.WriteLine("Marca y modelo: {0} {1}", marca, modelo);
              impresiones++;
        }
}
```

Declaración de una variable de clase, pública

La variable es usada como una variable más. En este ejemplo la usamos para llevar la contabilidad de la cantidad de veces que se imprime un auto.

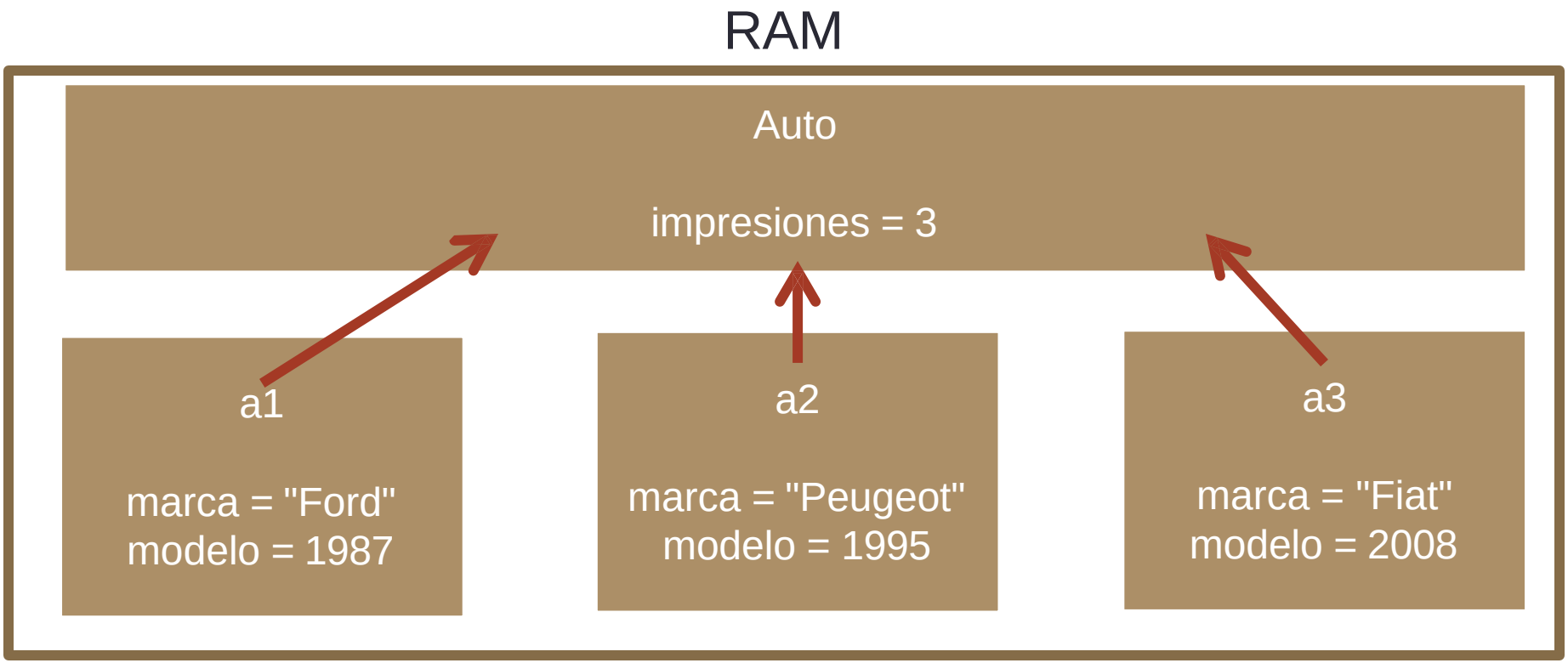
Miembros de clase

```
class Program
    {
        public static void Main(string[] args)
        {
            Auto a1 = new Auto("Ford", 1987);
            Auto a2 = new Auto("Peugeot", 1995
            Auto a3 = new Auto("Fiat", 2008);
            a1.imprimir();
            a2.imprimir();
            a3.imprimir();
            Console.WriteLine(Auto.impresiones);
        }
    }
```

Notar que para hacer referencia a una variable de clase en una aplicación, se la califica con el nombre de la clase.

Qué imprime?

Miembros de clase



La variable impresiones es una variable de clase, es única (hay una sola copia guardada en la clase) y es común a todas las instancias. Por eso imprime el valor 3.

Ejemplos de uso

```
public static void Main(string[] args)
        {
            Auto a1 = new Auto("Ford", 1987);
            Auto a2 = new Auto("Peugeot", 1995);
            Auto a3 = new Auto("Fiat", 2008);
            a1.imprimir();
            a2.imprimir();
            a3.imprimir();
            Console.WriteLine(Auto.impresiones);
            a1.impresiones++;
            }
```

Esta sentencia produce error.
Los miembros de clase no pueden ser referenciados con instancias.
a1 es un objeto.

Puesta en común

Modifique la clase Hora agregándole el huso horario y escriba un programa de aplicación (Main) que permita imprimir por consola:

xx HORAS, yy MINUTOS Y zz SEGUNDOS – (GTM-3),

donde GTM-3 corresponde al huso horario de Buenos Aires - Argentina

- Pensemos: el huso horario es un valor único y común para toda la Argentina. En todas las provincias tenemos la misma hora correspondiente al mismo huso…..entonces, qué tipo de campo o variable de instancia será huso horario?

- Defina los métodos necesarios para consultar y modificar dicho campo. Qué tipos de métodos son?

```
public class Hora
  {
    /*variables de instancia*/
    public int hour;
    public int min;
    public int seg;

    /*variable de clase*/
    public static string husoHorario="GTM- 03:00  Bs.As";

    /*constructor con inicializacion de variables de instancia*/
    public Hora(int hour,int m, int s)
    {
       this.hour=hour;
       min=m;
       seg=s;
    }
```

Por qué no se inicializa en el constructor esta variable de clase?

```
  /*método de instancia*/
      public void imprimir()
    {
      Console.WriteLine("{0} HORAS, {1} MINUTOS Y {2} SEGUNDOS",hour,min,seg);
    }
  /*propiedades*/
    ..................

  /*metodo de clase para imprimir el huso horario*/
    public static void verHuso()
    {
      Console.WriteLine(husoHorario);
    }
/*metodo de clase para modificar el huso horario*/
    public static void asignarHuso(string nuevo)
    {
       husoHorario=nuevo;
    }
}
```

Composición de objetos

- Cuando se programa con objetos es usual crear muchas clases de objetos, donde cada una tiene su estado y comportamiento específico.

- Muchas veces cuando un objeto es "muy grande", es decir tiene muchos estados y mucho comportamiento, es preferible crear varios objetos más chicos y luego un objeto que los englobe a todos juntos.

- La composición de objetos implica que un objeto está formado por otros.

Composición de objetos

Supongamos que queremos definir una clase llamada Dueño que tenga un campo privado llamado nombre, otro apellido y otro capaz de almacenar una mascota.

Cómo sería la implementación de la clase?

Estaríamos definiendo un objeto Dueño que tiene como atributo un objeto de la clase Mascota.

Mostraría composición de clases!

```
public class Dueño {
        private string nombre, apellido;
        private Mascota suMascota;


        /*constructor*/
        public Dueño(string nombre, string apellido){
                this.nombre = nombre;
                this.apellido = apellido;
         }
        public Dueño(string nombre, string apellido, Mascota mas){
                this.nombre = nombre;
                this.apellido = apellido;
                suMAscota=mas;
       }
```

Composición de objetos. En un atributo de dueño interviene un objeto de la clase Mascota.

```
/*propiedades*/
public string Nombre{
        set {
                nombre = value;
            }
        get {
                return nombre;
            }
        }

public string Apellido{
        set {
                apellido = value;
        }
        get {
                return apellido;
        }
}

public Mascota SuMascota
        {
                set {
                        suMascota=value;
                    }
                get {
                        return suMascota;
                    }
         }
}
```

Cómo lo usamos ahora en un programa de aplicación, que instancie un dueño, una mascota y los relacione (le asigne la mascota a su dueño).

```
Dueño per = new Dueño("Juan", "Diaz");
Mascota animal = new Mascota(“Fido”,"perro",”Juan”,2);

per.SuMascota=animal;          //se usa la propiedad set
```

Esta sentencia genera la composición de objetos. La instancia dueño "conoce" a la instancia mascota.

Veamos otro ejemplo más complejo

- Supongamos tener que modelar una Veterinaria, que guarda la información de todas las mascotas que atiende.

- Para modelar cada una de las mascotas usaremos la clase ya definida.

- Vemos que un objeto de clase Veterinaria está compuesto por varios objetos de la clase Mascota.

Composición de objetos

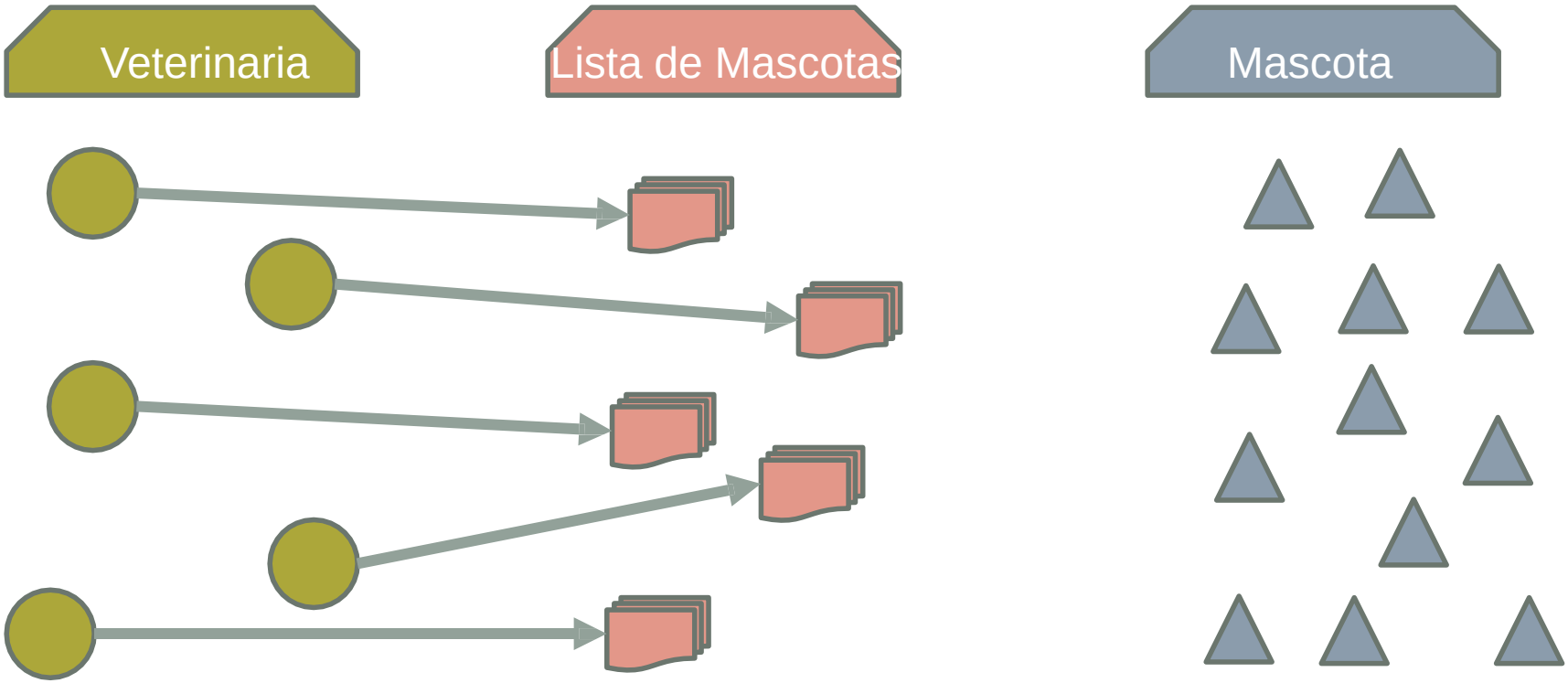
- Atributos
  - nombreVete
  - listaMascotas
  - cantidadMascotas
- Comportamiento
  - agregarMascota(…..)
  - atenderMascota(……)
  - verDatosDeUnaMascota(…..)
  - eliminarMascota(…..)
  - totalMascotas()

De qué clase será esta variable?

Es un conjunto de objetos de clase Mascota (Array o ArrayList)

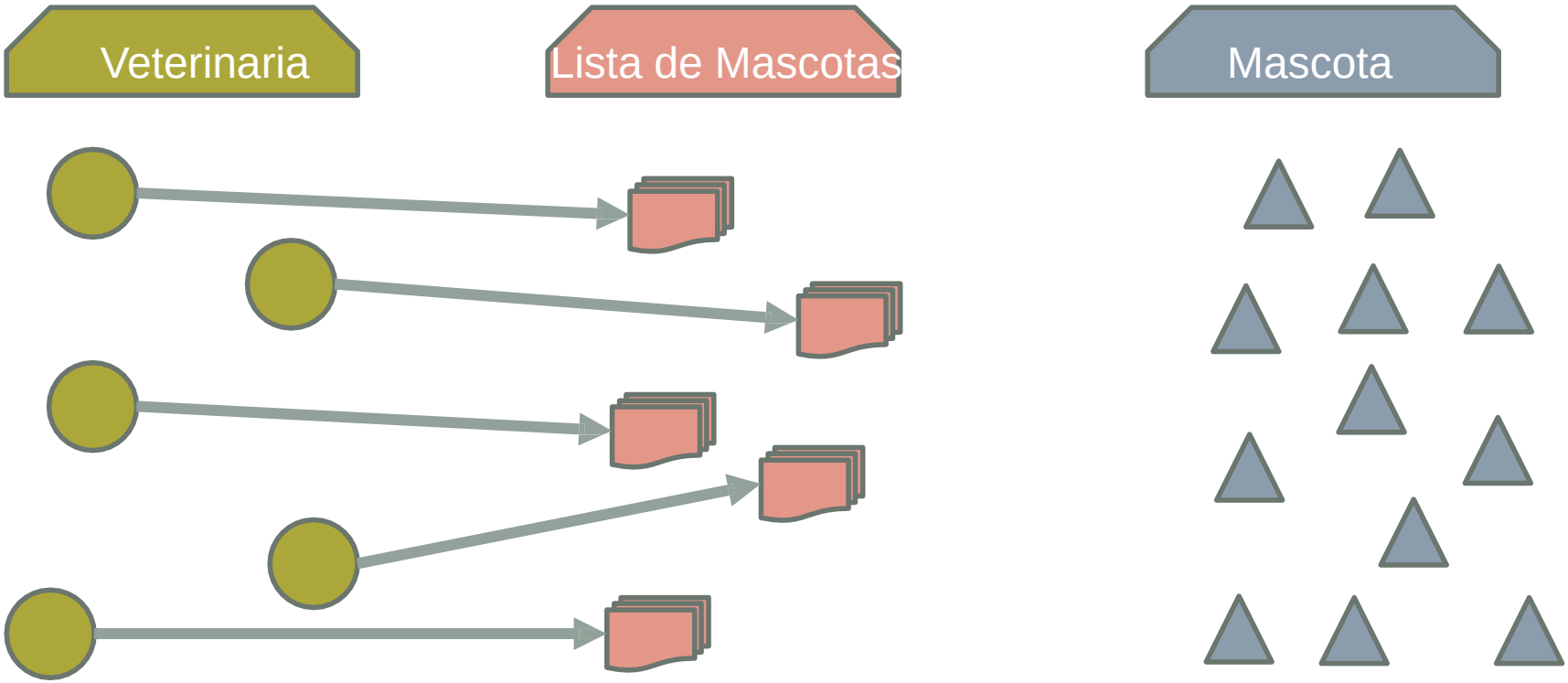
Composición de objetos

- Cada veterinaria (instancia) tiene su propia lista de mascotas



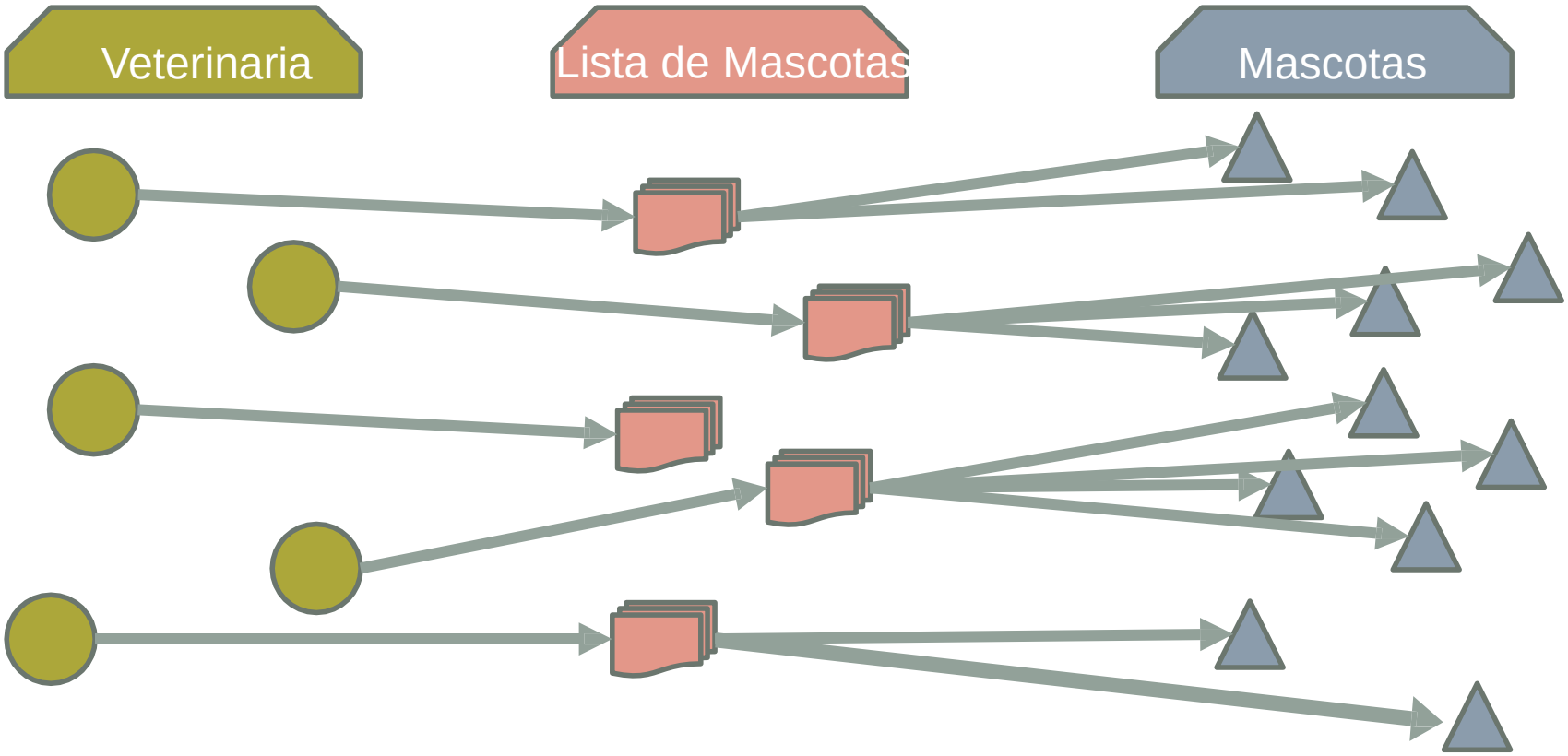
Composición de objetos

- Cada veterinaria (instancia) tiene su propia lista de mascotas



Composición de objetos

- Cada lista de mascotas puede tener 0 o más mascotas



Composición de objetos

```
public class Veterinaria
{
        private string nombreVete;
        private Mascota [] listaMascotas;
        private int cantidadMascotas;
        ……
        ……..
        ……..

}
```

Esto es composición de objetos. En el estado de un objeto interviene/n otro/s objeto/s.

Composición de objetos

```
public class Veterinaria
{
        private string nombreVete;
        private Mascota [] listaMascotas;
        private int cantidadMascotas
        /*constructor*/
        public Veterinaria (string nom) {
            nombreVete=nom;
            listaMascotas = new Mascota [100];
            cantidadMascotas=0;
        }
}
```

El constructor es el lugar para instanciar la lista de mascotas. Allí se hace el **new**.
NO se recibe NUNCA como parámetro del constructor la lista.

```
/*propiedades*/
public string NombreVete {
    set { nombreVete=value;}
    get { return nombreVete;}
  }

public int CantidadMascotas {
    set { cantidadMascotas=value;}
    get { return cantidadMascotas;}
    }

 public Mascota [] ListaMascotas {
    get { return listaMascotas;}  /*se define solo propiedad de lectura/*
    }
```

```
/*métodos de instancia*/
  public void agregarMascota(Mascota unaM)
    {
       int pos=cantidadMascotas; /*obtiene última posición libre*/
       listaMascotas[pos]=unaM; /* asigna en esa posición*/
       cantidadMascotas+=1;
     }

 public Mascota verDatosDeUnaMascota( int pos)
    {
     return  listaMascotas[pos];
    }
```

```
public void eliminarMascota(Mascota unaM)
  {     //busca la posición de la mascota a borrar
        int pos=Array.IndexOf(listaMascotas,unaM);
        //en un Array no se pueden borrar elementos, hacemos corrimiento de elementos
           for ( int k=pos; k < cantidadMascotas; k++)
                 listaMascotas[k]= listaMascotas[k+1];

        listaMascotas[cantidadMascotas]=null;  //limpio la última posición
        cantidadMascotas=cantidadMascotas - 1;
  }
```

```
public void atenderMascota(string nameM, string nameDue, string nuevodiag)
    {/*dada una mascota y dueño, la busca y le pone su nuevo diagnóstico*/
        for (int j=0; j < cantidadMascotas; j++)
       {
           Mascota m= (Mascota) listaMascotas[j];  /*toma una Mascota*/
           if ((m.Nombre==nameM) && (m.NombreDelDueño==nameDue) )
                 {  m.Diagnostico=nuevodiag;
                    break;}
           }
     }
 ....
 ....
}
```

Nombre, NombreDelDueño y Diagnostico son propiedades de los objetos de la clase Mascota y por lo tanto las entienden sus instancias.

Como Veterinaria es un **objeto compuesto**, al acceder a una mascota, deben aplicarse las propiedades de dicha clase. No se pueden usar las variables de instancia de mascota directamente.

¿Cómo usamos la clase Veterinaria?

Vamos a realizar una aplicación que:

- cree una instancia de la clase Veterinaria de nombre “Huesitos” en donde se carguen varias mascotas.
- luego simule la atención de alguna mascota.
- indique la cantidad total de mascotas atendidas al final del proceso.
- imprima el listado de mascotas de mas de 5 años

```
//MAIN
Veterinaria  vete;
 string sigue;
 vete= new Veterinaria("Huesitos");      /*creamos la veterinaria*/

Console.WriteLine("Desea cargar una mascota?s/n");
sigue= Console.ReadLine();
while (sigue=="s")
  {
      string nom, nomdue, esp;
      int edad;
      Console.WriteLine("ingrese nombre, dueño, especie y edad de una mascota");
      nom=Console.ReadLine();
      nomdue=Console.ReadLine();
      esp=Console.ReadLine();
      edad=int.Parse(Console.ReadLine());
      Mascota mas=new Mascota (nom,esp,nomdue,edad);  /*se crea una mascota*/
      vete.agregarMascota(mas);                       /*se agrega a la veterinaria*/
      Console.WriteLine("ingresa otra mascota?");
      sigue=Console.ReadLine();
}
```

Al trabajar con objetos compuestos, se deben usar los distintos niveles de abstracción, es decir, los métodos correspondientes a cada clase.

```
int cant=0;
 Console.WriteLine(“hay mascota a atender?s/n”);
 sigue=Console.ReadLine();
  while (sigue==“s”)
  {
    Console.WriteLine(“ingrese nombre de la mascota y del dueño”);
    nom=Console.ReadLine();
    nomdue=Console.ReadLine();
    Console.WriteLine(“ingrese diagnostico”);
    string diag=Console.ReadLine();
    vete.atenderMascota(nom,nomdue,diag);          /*atiende a una mascota*/

    cant+=1;                    /*cuenta la mascota atendida*/

    Console.WriteLine(“hay otra mascota a atender?s/n”);
    sigue=Console.ReadLine();
  }

 Console.WriteLine(“El total de mascotas atendidas es {0}”, cant);
 Console.Readkey(true);
```

```
/*imprimir el listado de mascotas con más de 5 años de edad*/

 for (int j=0; j < vete.CantidadMascotas; j++)
      {
       Mascota mas= vete.verDatosDeUnaMascota(j);   /*toma una mascota*/
       if (mas.Edad > 5)
          Console.WriteLine(mas.Nombre);
       }
  Console.Readkey(true);
}
 ************************************************** El for se puede reemplazar por:
foreach( Mascota m in vete.ListaMascotas)
  {
         if (m.Edad > 5)
         Console.WriteLine(m.Nombre);
    }
```

Combina el uso de métodos/propiedades de la clase Veterinaria con los/las de la clase Mascota, según corresponda.

Puesta en común Ejercicio 1 – TP 5

Defina la clase Horario, que almacena la hora de inicio, el día de la semana y nombre de la materia que se cursa en dicho horario.

Cuántos getters y setters se necesitan en este caso?

```
public class Horario
   {
      /*variables de instancia privadas*/
      private string dia, horaIn, materia;

      /*constructor*/
      public Horario(string nomdia, string hora, string nombreM)
      {
         dia= nomdia;
         horaIn=hora;
         materia=nombreM;
      }

    /*propiedades*/
      public string Dia
      {
         set{ dia=value;}
         get{ return dia;}
      }
      public string HoraIn
      {
         set{ horaIn=value;}
         get{ return horaIn;}
      }
      public string Materia
      {
         set{ materia=value;}
         get{ return materia;}
      }
 }
```

Puesta en común Ejercicio 1 – TP 5

Defina la clase Alumno. Considere cómo representaría los horarios de las materias que un alumno cursa.

1- Qué atributos podemos definirle a un alumno?

2- Cómo se guardan los horarios?

3- La clase Alumno es una clase compuesta? Justifique…

4- Qué operaciones podemos definir para manejar los horarios?

```
public class Alumno
   {
      private string nombre;
      private int dni,legajo;
      private ArrayList listahorarios;  //composición


      /*constructor*/
      public Alumno(string name, int docu, int leg)
      {
         nombre=name;
         dni=docu;
         legajo=leg;
         listahorarios= new ArrayList();    /*se crea la lista de horarios vacía
para que se vayan agregando a medida que el alumno se inscribe a una
materia*/
      }
```

Recordar que NUNCA se recibe la lista como argumento del constructor. Se crea con **new** dentro del constructor.

Operaciones posibles:

- Agregar materia (agrega un horario con la nueva materia que cursa el alumno. No debe estar ocupado ese horario con otra materia)
- Borrar materia (elimina el horario de dicha materia)
- Listado de horarios (imprime los horarios y las materias que cursa en dichos horarios)
- Listado de materias (imprime el nombre de las materias que cursa el alumno)
- Cantidad total de materias
- Consultar una materia (dado el nombre de la materia muestra el horario en que la cursa)
- Consultar un horario (dado un día y hora retorna el nombre de la materia que cursa; o avisa que está libre ese horario)

  ……..

```
public void agregarMat(string dia, string hora, string nomMat)
    {
      /*busco si ya esta ocupado el horario recibido*/
      bool ocupado=false;

      foreach (Horario elem in listahorarios)
      {
        if (dia==elem.Dia)
        {
          if (hora==elem.HoraIn)      /*horario ocupado por otra materia*/
          {ocupado=true;
            break;
          }
        }
      }
      if (!ocupado)              /*si el horario esta libre agrego la materia al alumno*/
      {
        Horario h=new Horario(dia,hora,nomMat);      /*creo un objeto horario con los datos*/
        listahorarios.Add(h);                        /*lo agrego a la lista de horarios*/
      }
      else
      {   Console.WriteLine("{0} {1} horario ocupado", dia, hora);
        Console.ReadKey();
      }
    }
```

```
public void listadoMaterias()
      {

          ArrayList mat=new ArrayList();            /*defino una lista auxiliar donde guardar los nombres de
las materias sin repeticiones ya que puede cursar la misma materia dos días de la semana*/

          foreach (Horario elem in listahorarios)
          {
             if (!mat.Contains(elem.Materia))        /*si la materia no existe en la lista auxiliar, la agrego */
                mat.Add(elem.Materia);

          }

          Console.WriteLine("\n Materias");
          Console.WriteLine(" ********");
          foreach (string nom in mat)
          {
             Console.WriteLine( " {0}",nom);

          }
      }

…….
……..
……..
```

```
public void listadoHorariosyMaterias()
    {

        Console.WriteLine("\n Horarios de Materias");
        Console.WriteLine(" ********");

       foreach (Horario elem in listahorarios)
       {
           Console.WriteLine(" \nDía {0} ,  hora {1}   Materia: {2} ", elem.Dia, elem.HoraIn, elem.Materia);

       }


    }
```

…….
……..
……..

Muchas gracias

Universidad Nacional
ARTURO JAURETCHE