UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

POO en Java

Instructor:

MC. Gerardo Gálvez Gámez

gerardo.galvez@uas.edu.mx

Diciembre de 2016

Programación Orientada a Objetos

La Sintaxis en el Lenguaje Java

Declaración de una clase

```
public class Circulo {
    //Atributos
   private float radio;
   //Constructores
   public Circulo(){// por defecto
      this.radio=0;
   }
   public Circulo(float pRadio) {//parametrizado
      this.radio=pRadio;
   }
   //Propiadades  (set y get )
   public void setRadio(float pRadio){
      radio=pRadio;
   }
   public float getRadio(){
      return radio;
   }
   //Definir los métodos
    public  double Area(){
          return Math.PI * Math.pow(this.radio, 2);
    }
}
```


Creación de objetos

- Paso 1: Asignación de memoria
  - Se usa **new** para asignar memoria
- Paso 2: Inicialización del objeto usando un constructor
  - Se usa el nombre de la clase seguido por paréntesis

```
Circulo Figura = new Circulo( );

Circulo Figura = new Circulo(12.5f);
```

Uso del constructor por defecto

- Características de un constructor por defecto:
  - Acceso público
  - Mismo nombre que la clase
  - No tiene tipo de retorno (ni siquiera **void**)
  - No recibe ningún argumento
  - Inicializa todos los campos a **cero**, **false** o **null**

- **Sintaxis del constructor:**

```
class NombreClase
 {
        public NombreClase( )
        {
                ...
        }
 }
```

Ejemplo:

```
class Circulo
{
        private float radio
        public Circulo( )
        {
                this.radio=0.0f;
        }
}
```


Sobrecarga de constructores

- Los constructores son métodos y pueden estar sobrecargados
  - Mismo ámbito, mismo nombre, distintos parámetros
  - Permite inicializar objetos de distintas maneras
- AVISO
  - Si se escribe un constructor para una clase, el compilador no creará un constructor por defecto

```
class Circulo
{
        private float radio
        public Circulo ( )
        {
            this.radio=0;
        }
        public Circulo (float pRadio )
        {
            this.radio=pRadio;
        }
}
```

Destrucción de Objetos

- Los objetos se destruyen por recolección de basura
- La liberación de memoria en Java es automática.
- Java posee un mecanismo denominado **Garbage Collection** mediante el cual, y de forma periódica, localiza objetos que no tengan ninguna referencia que les apunte para liberar la memoria que están utilizando

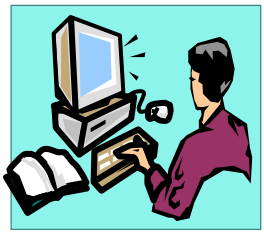
Mecanismos de accesibilidad

Accesibilidad	Acceso en la misma clase	Acceso en el mismo package	Acceso en el árbol de herencia	Acceso desde todas las clases
private	Si	No	No	No
default o package	Si	Si	No	No
protected	Si	Si	Si	No
public	Si	Si	Si	Si

- ***"package"***
  - –accesible por las clases del paquete, no accesible a los clientes del paquete
- ***public***
  - –accesible por todas las clases
- ***private***
  - –sólo accesible por los métodos de la clase
- ***protected***
  - –accesible por las clases del paquete y por las subclases

Práctica – Creación de objetos

Crear objetos en diferentes ámbitos y probar su accesibilidad.



Actividad #1

- Elaborar un proyecto, orientado a objetos, que permita realizar las siguientes operaciones sobre un Circulo:
  - Calcular su Área: Se utiliza la siguiente formula:
    - Area: PI * radio^2
  - Calcular su Longitud: Se emplea la siguiente formula:
    - Longitud: 2 * PI *radio

Diseño de clase propuesto

Circulo
-radio:float
+Circulo() +Circulo(pRadio:float) + Radio:float (get, set) +area():double +longitud():double

Actividad #2: Diseñar y programar la Clase, e instanciar objetos.

Elaborar un proyecto, orientado a objetos, que permita calcular el sueldo quincenal y mensual de un empleado.

El empleado cuenta con los siguientes datos: ***Nombre, Puesto:***

El sueldo Mensual, se calcula en base a la siguiente formula:

- Mensual=Sueldo Base – Retención – Cuota IMSS

Donde: La retención de calcula en base a:

- Para un SueldoBase de hasta de 2000 no hay retención,
- Para un SueldoBase mayor de 2000 y hasta 3000 el porcentaje de retención es de 10%.
- Para un SueldoBase mayor a 3000 el porcentaje de retención es el 15%.
- Nota: el sueldo quincenal es la mitad del sueldo mensual.

Diseño de Clases (jerarquía de clases)

Empleado	
-nombre:string -puesto:string -sueldobase:float -cuotaIMSS:float	Atributos(campos)
+Empleado() +Empleado(pNombre:string, pPuesto:string, pSueldoBase:float, pCuotaIMSS:float)	Constructores
+Nombre:string(get,set) +Puesto:string(get,set) +SueldoBase:float(get,set) +CuotaIMSS:float(get,set)	Propiedades
-Retencion():float +SueldoMensual():float +SueldoQuincenal():float	Métodos

Construcción del *Algoritmo (Pseudocódigo)*

Objetivo: Determinar retención correspondiente a un empleado.
Programador: MC. Gálvez Gámez Gerardo
Fecha: __/diciembre/2015

```
//atributo de clase
 SueldoBase

REAL  Retencion()
INICIO
     //Definición de Variables y Constantes
      CONST REALPorcentajeRetencion1=10, PorcentajeRetencion2=15
      CONST REAL ValorRetencion1=2000, ValorRetencion2=3000
      REAL Retencion
      //Proceso determinar el tipo de número
    SI SueldoBase <ValorRetencion1 ENTONCES
            Retencion=0
    SI_NO
        SI SueldoBase <ValorRetencion2 ENTONCES
            Retencion=SueldoBase * (PorcentajeRetencion1/100)
       SI_NO
            Retencion=SueldoBase * (PorcentajeRetencion2/100)
       FIN_SI
     FIN_SI
     //(salida)
     REGRESAR Retencion
FIN
```

Herencia en Java

En Java solo se permite Herencia Simple

Sintaxis de clase

[ClassModifiers] **`class`** ClassName
[**`extends`** SuperClass]
[**`implements`** Interface1, Interface2 ...] **{**
ClassMemberDeclarations
}

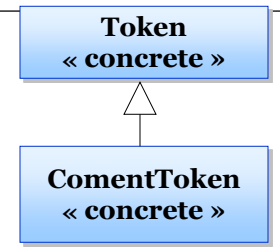
Extensión de clases base

- Sintaxis para derivar una clase desde una clase base

```
class Token
{
    ...
}
class ComentToken extends Token
{
    ...
}
```

Clase derivada

Clase base



- Una clase derivada hereda la mayor parte de los elementos de su clase base
- Una clase derivada no puede ser más accesible que su clase base

Acceso a miembros de la clase base

```
class Token
{   ...
    protected string name;
}
class ComentToken extends Token
{   ...
    public string Name( )
    {
        return name;
    }                          }
}
```

✓

```
class Outside
{
 void Fails(Token t)
 {
    ...
   t.name
    ...
 }
```

✗

- Los miembros heredados con protección están implícitamente protegidos en la clase derivada
- Los miembros de una clase derivada sólo pueden acceder a sus miembros heredados con protección

Llamadas a constructores de la clase base

- Las declaraciones de constructores deben usar la palabra base
- Una clase derivada no puede acceder a un constructor privado de la clase base
- Se usa la palabra base para habilitar el ámbito del identificador

```
class Token
{
   protected Token(string name) { ... }
   ...
}
class ComentToken extends Token
{
   public ComentToken(string name) {
      super(name)
   }
   ...
}
```

Observaciones

- Para referirse a la instancia actual usamos **this** y
- Para referirnos a nuestra clase padre usamos **super**

Polimorfismo (@override)

- Sintaxis: Se usa la palabra reservada @override

```
class Token
{   ...
    public virtual string Name( ) { ... }
}
class ComentToken extends Token
{   ...
    @override
    public string Name( ) { ... }
}
```

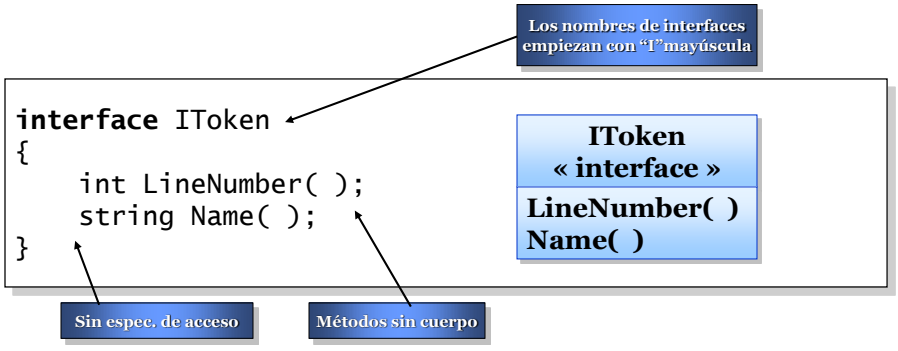
Uso de clases selladas

- Ninguna clase puede derivar de una clase sellada
- Las clases selladas sirven para optimizar operaciones en tiempo de ejecución
- **Sintaxis:** Se usa la palabra reservada **final**

```
namespace System
{
    public final class String
    {
        ...
    }
}
namespace Mine
{
    class FancyString extends String {✘.. }
}
```


Declaración de interfaces

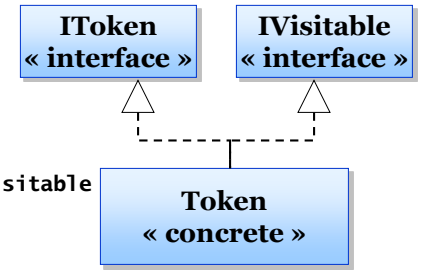
- Sintaxis: Para declarar métodos se usa la palabra reservada interface



Implementación de varias interfaces

- Una clase puede implementar cero o más interfaces

```
interface IToken
{
    string Name( );
}
interface IVisitable
{
    void Accept(IVisitante v);
}
class Token implements IToken, IVisitable
{ ...
}
```



- Una interfaz puede extender cero o más interfaces
- Una clase puede ser más accesible que sus interfaces base
- Una interfaz no puede ser más accesible que su interfaz base
- Una clase implementa todos los métodos de interfaz heredados

Implementación de métodos de interfaz

- El método que implementa debe ser igual que el método de interfaz
- El método que implementa puede ser virtual o no virtual

```
class Token implements IToken, IVisitable
{
    public virtual string Name( )
    { ...
    }
    public void Accept(IVisitante v)
    { ...
    }
}
```

Mismo acceso
Mismo retorno
Mismo nombre
Mismos parámetros

Declaración de clases abstractas

- Se usa la palabra reservada abstract

```
public abstract class Token
{
    ...
}
public class Test
{
    static void Main( )
    {
        new Token( );
    }
}
```

Token
{ abstract }

No se pueden crear instancias de una clase abstracta

Uso de clases abstractas en una jerarquía de clases

Ejemplo 1 Las clases abstractas se emplean a menudo para la implementación parcial de interfaces.

```
interface IToken
{
    string Name( );
}

abstract class Token implements IToken
{
    string IToken.Name( )
    {   ...
    }
    ...
}

class ComentToken extends Token
{   ...
}
class KeywordToken extends Token
{   ...
}
```

IToken
« interface »

Token
{ abstract }

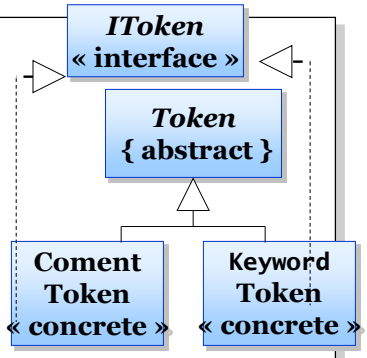
Coment Token « concrete »

Keyword Token « concrete »

Uso de clases abstractas en una jerarquía de clases *(cont.)*

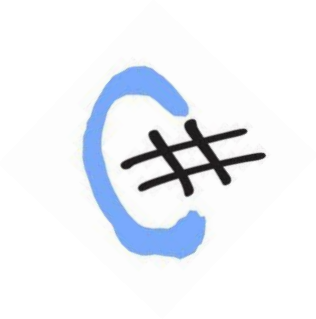
- **Ejemplo 2**

```
interface IToken
{
    string Name( );
}

abstract class Token
{
    public virtual string Name( )
    {   ...
    }
    ...
}


class ComentToken extends Token implements IToken
{   ...
}
class KeywordToken extends Token implements IToken
{   ...
}
```



Implementación de métodos abstractos

```
Public abstract class Token
{
    public string Name( ) { ... }
    public abstract int Longitud( );
}
class ComentToken extends Token
{
    @override
    public string Name( ) { ... }
    @override
    public int Longitud( ) { ... }
}
```

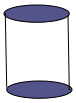


Figuras Geométricas (Luis joyanes Aguilar)

Circulo

Area: PI * radio²

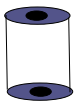
Longitud: 2 * PI *radio



Cilindro

Volumen: PI * radio² *altura

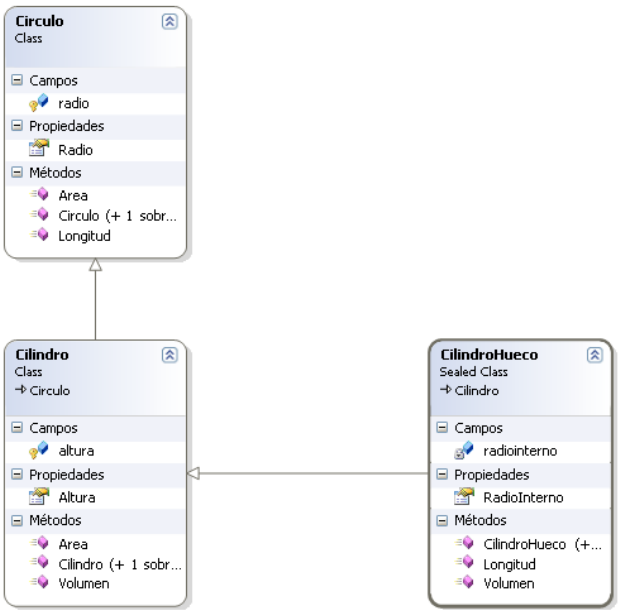
Area: 2 * PI * radio *altura + 2 * PI *radio²



Cilindro Hueco

Longitud: 2 * PI *(radio² - radioInterno²) + 2 * PI * radio * altura + 2*PI*altura*radioInterno

Volumen: PI *(radio² - radioInterno²) * altura



Elaboración de un proyecto Orientado a Objetos

- Utilizando como lenguaje a Java, con metodología orientada a objetos, que permita dar solución al problema planteado, utilizando las herramientas de **NetBeans.**
- Se desarrollaran los siguientes puntos:
  - Diseño de clases
  - Manipulación de objetos

Descripción general del problema:

En una empresa de Reparación automotriz de la localidad hoy dos tipos de empleados: ***administrativos y Mecánicos***.

Para ambos empleados se tienen los datos: ***RFC, Nombre y Puesto***. En particular para el ***empleado Mecánico***, su *sueldo Mensual* se calcula en base a la siguiente formula:

Retención $= \frac{Sueldo\ Base}{5} - \$40 * (Numero\ de\ Hijos - 2)$

Si el empleado tiene 1 o ningún hijo la retención será:

Retención $= \frac{Sueldo\ Base}{5}$

Final mente: el

Sueldo Mensual=Sueldo Base – Retención – Cuota IMSS

Para el *empleado Administrativo*

El sueldo mensual se determina en base a la siguiente política:

Se hace una retención de su sueldo Base:

- Para un sueldo de hasta de 1500 no hay retención,
- Para un sueldo mayor de 1500 y hasta 3000 el porcentaje de retención es de 5%.
- Para un sueldo mayor a 3000 el porcentaje de retención es el 8%.

Finalmente el sueldo mensual:

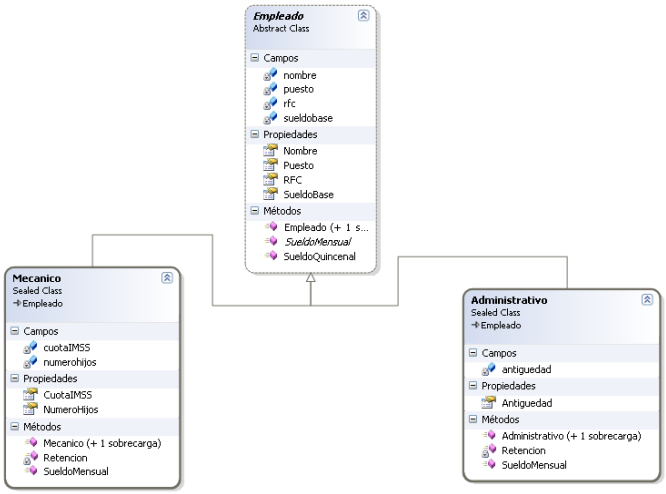
- Sueldo Mensual =Sueldo Base – retención + Antigüedad

(La antigüedad es el 5% de su sueldo Base, solo para aquellos empleados con una antigüedad mayor o igual a 5 años, en caso contrario es cero).

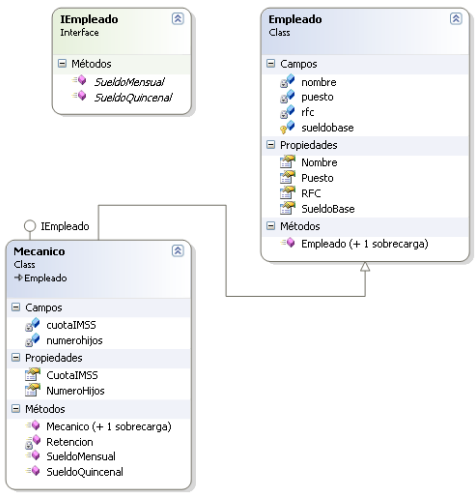
Nota: El salario quincenal de ambos empleados es la mitad de su salario Mensual.

La aplicación deberá permitir el registro de los datos de ambos empleados, así como el calculo del pago correspondiente tanto por quincena como por mes según lo indique el usuario.

Propuesta #1



Propuesta #2



Preguntas?