Universidad Autónoma de Madrid

Escuela Politécnica Superior

Análisis y Diseño de Software (ADSOF)

Curso 2017-2018

Práctica 3

Introducción a la Programación Orientada a Objetos (POO) en Java

Inicio: semana del 26 de febrero

Duración: 3 semanas

Entrega: semana del 19 de marzo

Peso en la calificación de prácticas: 20%

Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - **Tipos enumerados (*enum*)**
  - Herencia: subclases y superclases
  - Herencia: polimorfismo
  - Herencia: sobreescritura de métodos / Sobrecarga de métodos
  - Clases abstractas
  - Atributos de instancia vs. Atributos de clase
  - Métodos de instancia vs. Métodos de clase

Tipos enumerados

```
public class EnumAsig {
  enum  Asignatura {              // valores de la enumeración
      ALGEBRA, CALCULO, FISICA, PROGRAMACION, TALLER;
  }
  public static void main(String[] args) {
    Asignatura a = Asignatura.FISICA;
    System.out.println("vale: " + a); //imprime
    Asignatura[] valores = Asignatura.values();
    for (Asignatura x : valores) System.out.println(x);
  }
}
```

```
vale: FISICA
```

```
ALGEBRA
CALCULO
FISICA
PROGRAMACION
TALLER
```

Tipos enumerados

```
public class EnumDia {
  enum Dia {
    LUNES(1), MARTES(2), MIERCOLES(3), JUEVES(4),
    VIERNES(5), SABADO(6), DOMINGO(0);

    private Dia(int d) { valor = d; } // constructor privado
    private final int valor;      // valor interno controlado

    public int valor() { return valor; }
  }

  public static void main(String[] args) {
    Dia dia = Dia.VIERNES;
    …
    Dia[] semana = new Dia[Dia.values().length];
    for (Dia d : Dia.values()) { semana[d.valor()] = d; }
    for (Dia d : Dia.values())
      System.out.println("semana["+d.valor()+"] es " + d);
  }
}
```

```
semana[1] es LUNES
semana[2] es MARTES
semana[3] es MIERCOLES
semana[4] es JUEVES
semana[5] es VIERNES
semana[6] es SABADO
semana[0] es DOMINGO
```

Tipos enumerados

```
public enum Planeta {                    // en archivo Planeta.java
  MERCURIO(3.303e+23, 2.4397e6), VENUS(4.869e+24, 6.0518e6),
  TIERRA(5.976e+24, 6.37814e6), MARTE(6.421e+23, 3.3972e6),
  JUPITER(1.9e+27, 7.1492e7), SATURNO(5.688e+26, 6.0268e7),
  URANO(8.686e+25, 2.5559e7), NEPTUNO(1.024e+26, 2.4746e7);

  Planeta(double m, double r) { masa = m; radio = r; }

  private final double masa;   // kg
  private final double radio;  // metros

  private double masa() { return masa; }
  private double radio() { return radio; }

  // cte. gravitation universal
  public static final double G = 6.673E-11;
  // metodos adicionales
  double gravedadSup() { return G * masa / (radio*radio); }
  double pesoSup(double masa { return masa*gravedadSup(); }
}
```

Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - Tipos enumerados (enum)
  - **Herencia: subclases y superclases**
  - Herencia: polimorfismo
  - Herencia: sobreescritura de métodos / Sobrecarga de métodos
  - Clases abstractas
  - Atributos de instancia vs. Atributos de clase
  - Métodos de instancia vs. Métodos de clase

Herencia: subclases y superclases

```
public class Persona {
    public String nombre;
    public int edad;
    public String toString() {
        return "Nombre: " + nombre + "\nEdad: " + edad;
    }
}
```

```
public class Empleado extends Persona {
    public long sueldoBruto;
    public Directivo jefe;
    ...
}
```

*Empleado es una **subclase** de Persona;*

*Persona es la **superclase** de Empleado*

```
public class Directivo extends Empleado {
    public long incentivo;
    public ArrayList<Empleado> equipo;
    public void fijarIncentivo(long c) { incentivo = c; }
}
```

Herencia: subclases y superclases

```
public class Persona {
    public String nombre;
    public int edad;
    public String toString() {
        return "Nombre: " + nombre + "\nEdad: " + edad;
    }
}
```

Usamos atributos **`public`** *solamente por facilitar los ejemplos que vienen a continuación.*

```
public class Empleado extends Persona {
    public long sueldoBruto;
    public Directivo jefe;
    ...
}
```

Empleado es una ***subclase*** *de Persona;*

Persona es la ***superclase*** *de Empleado*

```
public class Directivo extends Empleado {
    public long incentivo;
    public ArrayList<Empleado> equipo;
    public void fijarIncentivo(long c) { incentivo = c; }
}
```

Herencia: subclases y superclases

```
Empleado emp = new Empleado();
Directivo dir = new Directivo();

emp.nombre = "Pedro";      // campo de emp como Persona
emp.edad = 28;
emp.sueldoBruto = 2000;    // campo de emp como Empleado
emp.jefe = dir;
System.out.println(emp);   // llamada implícita a emp.toString()
                           // método de Persona
dir.nombre = "Maria";
dir.edad = 45;
dir.sueldoBruto = 5000;
dir.jefe = null;
dir.incentivo = 1500;
System.out.println(dir.toString());
```

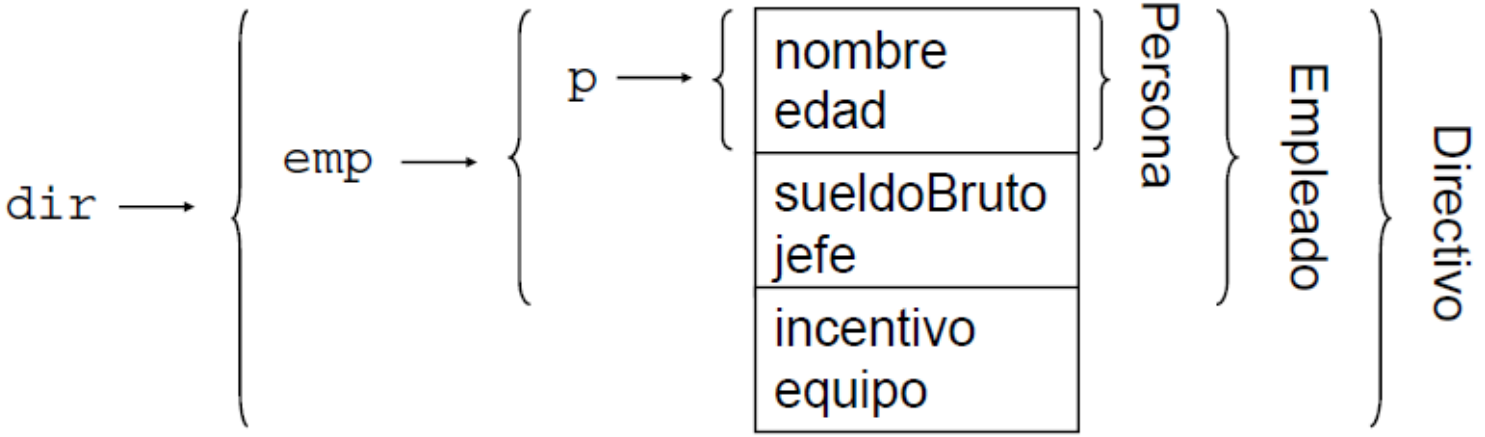
Aquí aprovechamos que antes declaramos atributos **`public`**.

Herencia: subclases y superclases

```
Directivo dir = new Directivo();
Empleado emp = dir;
Persona p = dir;
```

Tipo estático vs. tipo dinámico

```
p.sueldoBruto = 1000; // Error: sueldoBruto no
                      // definido para Persona
emp.fijarIncentivo(0); // Error: fijarIncentivo no definido
                       // para Empleado
```



Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - Tipos enumerados (enum)
  - Herencia: subclases y superclases
  - Herencia: polimorfismo
  - **Herencia: sobreescritura de métodos / Sobrecarga de métodos**
  - Clases abstractas
  - Atributos de instancia vs. Atributos de clase
  - Métodos de instancia vs. Métodos de clase

Sobreescritura y sobrecarga de métodos

```
public class Point {
  private int x = 0, y = 0, color;

  void move(int mx, int my) { x += mx; y += my; }


}

class RealPoint extends Point {
  double dx = 0.0, dy = 0.0;
  void move(int mx, int my) {
     move((double)mx, (double)my);
  }
  void move(double mx, double my) { dx += mx; dy += my; }


}
```

El método move(int,int) se sobrescribe y se añade el método move(double,double) con sobrecarga en RealPoint

Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - Tipos enumerados (enum)
  - Herencia: subclases y superclases
  - **Herencia: polimorfismo**
  - Herencia: sobreescritura de métodos / Sobrecarga de métodos
  - Clases abstractas
  - Atributos de instancia vs. Atributos de clase
  - Métodos de instancia vs. Métodos de clase

Herencia: polimorfismo – ligadura dinámica

```
public class Persona {
    String nombre;
    int edad;
    public String toString() {
        return "Nombre: " + nombre + "\nEdad: " + edad;
    }
}

public class Empleado extends Persona {
    long sueldoBruto;
    Directivo jefe;
    public String toString() {
        return super.toString() +
               "\nSueldo: " + sueldoBruto + "\nJefe: " +
               ((jefe == null)? nombre : jefe.nombre);
    }
}
public class Directivo extends Empleado {
    long incentivo;
    ArrayList<Empleado> equipo = new ArrayList<Empleado>();
    public String toString() {
        return super.toString()+ // toString() de Empleado
               "\nIncentivo: " + incentivo;
    }
    public void fijarIncentivo(long c) { incentivo = c; }
}
```

Herencia: polimorfismo – ligadura dinámica

```
Directivo dir = new Directivo();
Empleado emp = new Empleado();
Empleado e = dir;
Persona p = new Persona();
Persona x = emp;
Persona y = e;
String s;
s = p.toString();     // toString de Persona
s = emp.toString();   // toString de Empleado
s = dir.toString();   // toString de Directivo
s = x.toString();     // toString de Empleado
s = y.toString();     // toString de Directivo
s = e.toString();     // toString de Directivo
y.fijarIncentivo(1500); // ERROR
```

Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - Tipos enumerados (enum)
  - Herencia: subclases y superclases
  - Herencia: polimorfismo
  - Herencia: sobreescritura de métodos / Sobrecarga de métodos
  - **Clases abstractas**
  - Atributos de instancia vs. Atributos de clase
  - Métodos de instancia vs. Métodos de clase

Clases abstractas

```
public abstract class Figura {
  public abstract double perimetro();

}

class Circulo extends Figura {
  Punto2D centro;
  double radio;
  public double perimetro() { return 2 * Math.PI * radio; }
}

class Triangulo extends Figura {
  Punto2D a, b, c;
  public double perimetro() {
    return a.distancia(b) + b.distancia(c) + c.distancia(a);
  }
}
```

Aquí `perimetro()` *es un método abstracto, sin* `{}` *sin implementación, solo* `;`

Estas subclases dan su propia implementación de `perimetro()` *heredado y ya no es método abstracto*

Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - Tipos enumerados (enum)
  - Herencia: subclases y superclases
  - Herencia: polimorfismo
  - Herencia: sobreescritura de métodos / Sobrecarga de métodos
  - Clases abstractas
  - **Atributos de instancia vs. Atributos de clase**
  - Métodos de instancia vs. Métodos de clase

Atributos de instancia vs. Atributos de clase

```
class CuentaBancaria {
    long numero;
    String titular;
    long saldo = 0;

    void ingresar(long cantidad) {
        saldo += cantidad;
    }
    void retirar(long cantidad) {
        if (cantidad > saldo)
            System.out.println("Saldo insuficiente");
        else saldo = saldo - cantidad;
    }
}  // fin declaración de clase CuentaBancaria
```

Variables de instancia

Métodos

Atributos de instancia vs. Atributos de clase

```
public class Punto {
    private long x, y;    // abcisa y ordenada de cada punto
    private static long nmrPuntos = 0; // variable de clase

    public Punto(long x, long y) {
        this.x = x;
        this.y = y;
        nmrPuntos++;  // contar cada punto
    }
    …
}
```

```
Punto.nmrPuntos
p.nmrPuntos   // suponiendo Punto p;
```

Práctica 3: Introducción a la POO en Java

- **Objetivos**
  - Tipos enumerados (enum)
  - Herencia: subclases y superclases
  - Herencia: polimorfismo
  - Herencia: sobreescritura de métodos / Sobrecarga de métodos
  - Clases abstractas
  - Atributos de instancia vs. Atributos de clase
  - **Métodos de instancia vs. Métodos de clase**

Métodos de instancia vs. Métodos de clase

```
public class Math {    // clase predefinida en java.lang

   // variable de clase de valor final, o constante
   public static final double PI = 3.141592653589793;

   static long round(double a) {    // método de clase
      …
   }
   static double sin(double a) { … }
   …
}
```