PARADIGMAS DE PROGRAMACIÓN
Licenciatura en Informática
Programador Universitario

UNIDAD IV
PARADIGMAS DE PROGRAMACIÓN
GENERALIZACIÓN

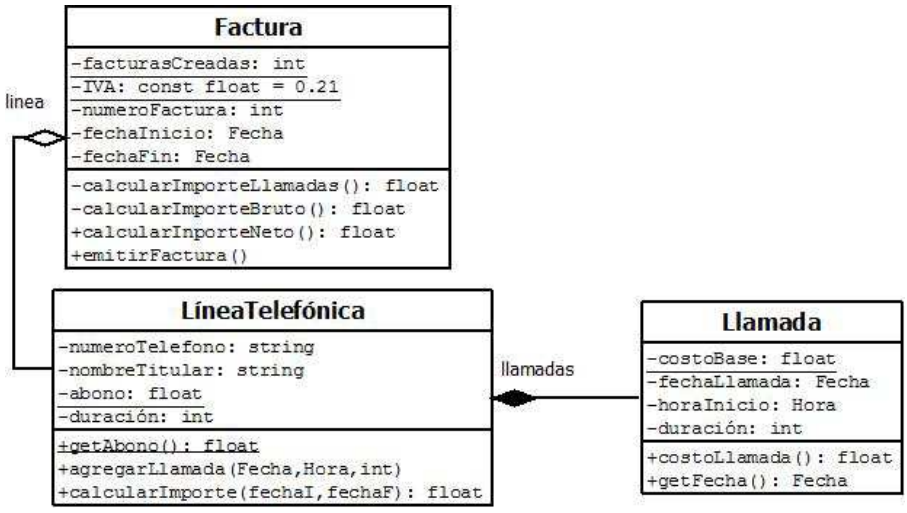
Factura Telefónica

Teniendo en cuenta el ejemplo FacturaTelefonicaSimple visto en clases diseñe e implemente las modificaciones necesarias para poder crear llamadas de distinto tipo (Celular, Urbana e Interurbana). El costo de la llamada dependerá del tipo de llamada:

- **Local**: el costo del minuto en una llamada local corresponde al costo base del minuto sin ningún incremento.
- **Interurbana**: el costo del minuto interurbano corresponde al costo base incrementado en un 50%. Pero si la llamada se inicia entre las 22 y las 6 horas, el costo de la llamada se calcula con tarifa reducida (mismo costo que llamada local)
- **Celular**: el costo del minuto de celular corresponde al costo base incrementado en 80%. Los primeros 5 minutos de llamada se calculan con este precio y los restantes, se reduce el costo en un 30%

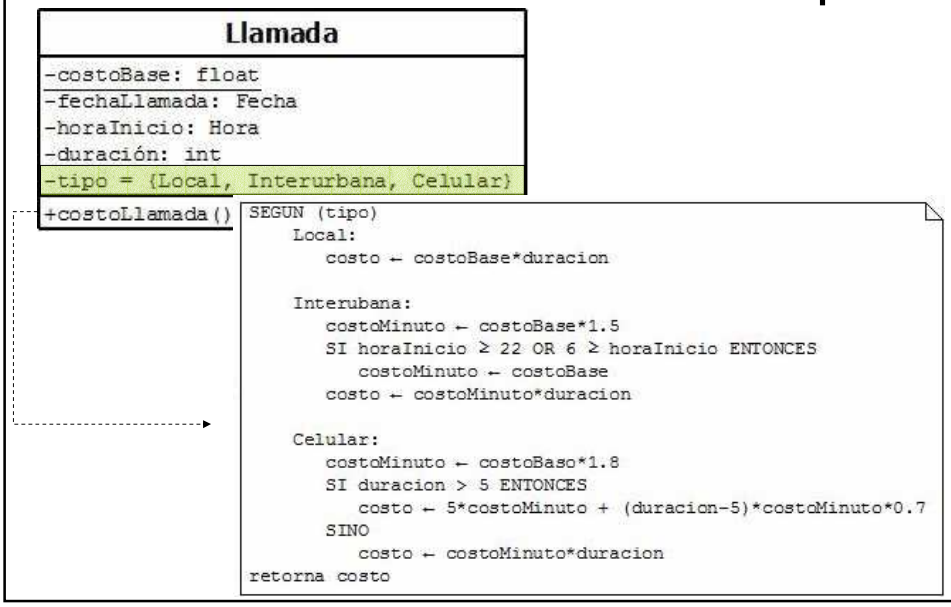
2

UML Factura Telefónica Simple



3

UML Factura Telefónica Simple



SWITCH

Enunciados switch → Forma alternativa de tratar objetos de muchos tipos diferentes

Problemas

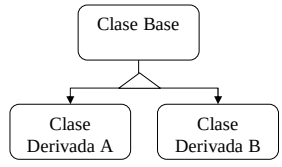
- El programador puede olvidarse de efectuar la prueba de tipo correspondiente cuando sea requerida
- El programador puede olvidarse alguno de los casos posibles
- La modificación de enunciados switch es lenta y está sujeta a errores

En la POO podemos eliminar la necesidad de la lógica switch utilizando **Herencia**

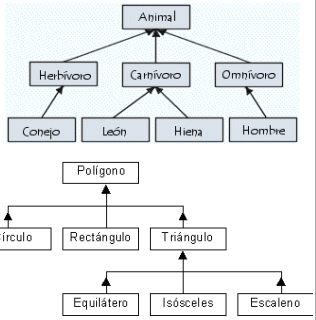
5

Herencia

HERENCIA → Proceso por el cual un O adquiere las propiedades de otro u otros O que constituyen sus predecesores jerárquicos



Relación *es-un* Un O de clase derivada también es un O de clase base (pero no al revés)

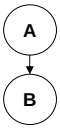


Relación de especialización ya que la clase derivada define un conjunto más reducido y especializado de O, que su clase base

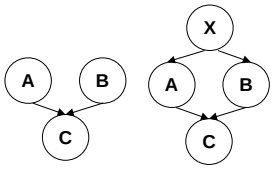
FORTALEZA → Capacidad de definir en una clase derivada, adiciones, reemplazos o refinamientos de las características heredadas de la clase base

Tipos

Simple

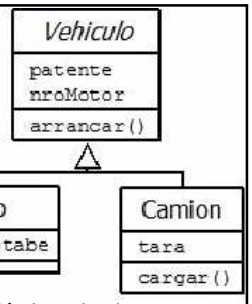


Múltiple



6

Herencia



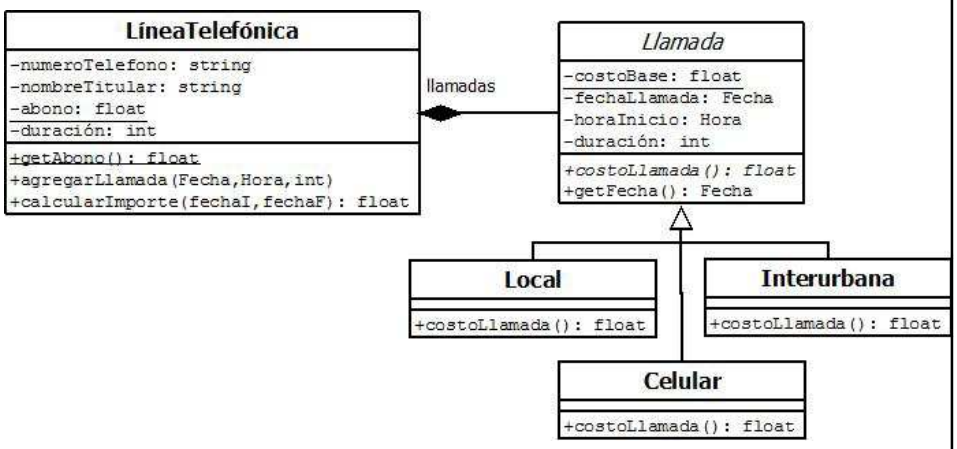
ASPECTOS FUNDAMENTALES

- **Carácter estructural**: Las subclases heredan las estructuras de sus clases base
- **Características de Comportamiento**: Las subclases también heredan los métodos de sus superclases o clases base
- **Especialización**: La herencia es contemplada como una especialización desde el punto de vista del tipo
- **Extensión**: La relación de herencia se puede contemplar como una extensión de la clase base.
- **Generalización**: La clase más general en una estructura de clases es la **clase base**. Ella representa la categoría de abstracción más generalizada dentro del dominio dado.

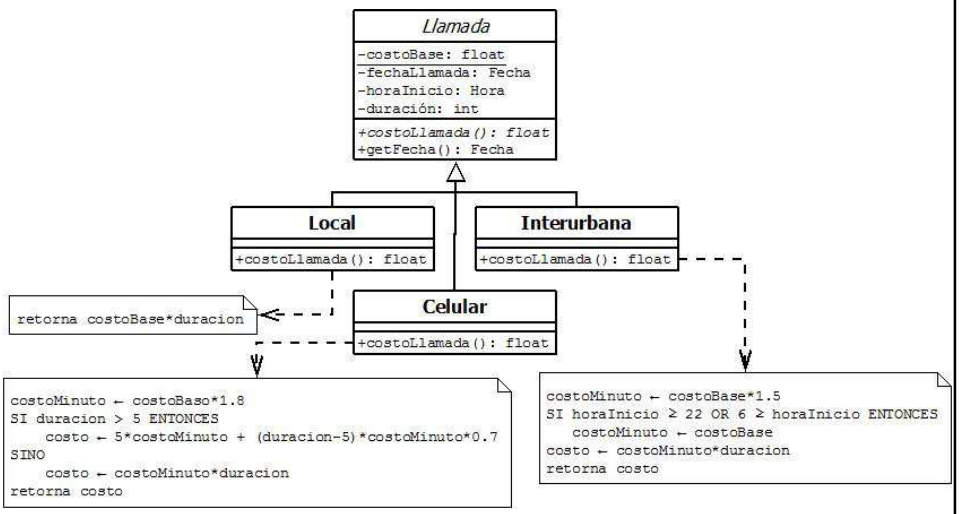
7

Factura Telefónica

Modificar el ejemplo Factura Telefónica para que cada tipo de llamada (local, interurbana y celular) represente una clase de objetos distinta



Factura Telefónica



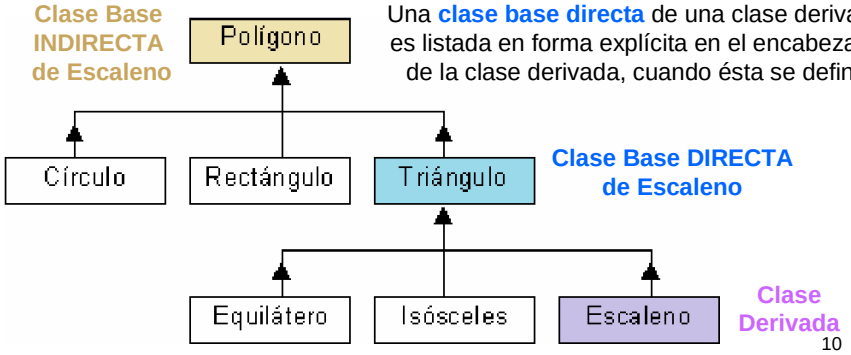
9

Clases Base Directas e Indirectas

Una **clase base indirecta** se hereda desde varios niveles arriba en la jerarquía de herencia de la clase. No se lista en forma explícita en el encabezado de la clase derivada

Clase Base INDIRECTA de Escaleno

Una **clase base directa** de una clase derivada es listada en forma explícita en el encabezado de la clase derivada, cuando ésta se define



Clase Base DIRECTA de Escaleno

Clase Derivada

10

Herencia

11

Herencia
Ocultamiento de información

```
class persona {
    private:
        string nombre;
        fecha fechaNacimiento;
        float peso, altura;

    protected:
        string direccion;

    public:
        persona (string,fecha,float,string);
        void cambioDireccion (string);
};
```

El diseñador de una clase puede definir (dependiendo del lenguaje) qué atributos y métodos son **visibles** en las subclases mediante los especificadores de acceso **public**, **protected** o **private** en la herencia

12

Herencia
Ocultamiento de información

```
class persona {
  private:
    string nombre;
    fecha fechaNacimiento;
    float peso, altura;

  protected:
    string direccion;

  public:
    persona (string,fecha,float,string);
    void cambioDireccion (string);
};
```

Miembros privados: son accesibles sólo a funciones miembros y amigos (C++) de la clase

Miembros protegidos: son accesibles sólo por funciones miembros y amigos (C++) de la clase base y por funciones miembros y amigos de las clases derivadas

Miembros públicos: son accesibles a todas las funciones del programa

13

Herencia Pública

```
class empleado: public persona {
    // variables privadas
          int departamento;
          string tipoTrabajo;
          int salario;
          string tipoFormación;
    //protected: string direccion;
    public:
          empleado (int, string, int, string, ...);
          cambioDepartamento(int);
          calculoSalario(int);
};        //void cambioDireccion (string);
```

Todos los miembros de la superclase (excepto los privados) son visibles en la subclase con la visibilidad que tenían en la superclase

C++ no admite la herencia de constructores, destructores ni el operador sobrecargado =

Herencia Protegida

```
class empleado: protected persona {
    // variables privadas
          int departamento;
          string tipoTrabajo;
          int salario;
          string tipoFormación;
   //protected: string direccion;
   //           void cambioDireccion (string);

    public:
          empleado (int, string, int, string, ...);
          cambioDepartamento(int);
          calculoSalario(int);
};
```

Los miembros privados de la superclase no son visibles en la subclase, y los miembros public y protected de la superclase pasan como protected a la subclase

15

Herencia Privada

```
class empleado: private persona {
    // variables privadas
          int departamento;
          string tipoTrabajo;
          int salario;
          string tipoFormación;
          //string direccion;
          //void cambioDireccion (string);
    public:
          empleado (int, string, int, string, ...);
          cambioDepartamento(int);
          calculoSalario(int);
};
```

Ningún miembro privado de la superclase es visible en la subclase, y los miembros protected y public de la superclase pasan como privados a la subclase

16

Herencia
Constructores y Destructores

Ya que una clase derivada hereda los atributos de su clase base, cuando se crea un O de una clase derivada, el **constructor de clase derivada siempre llamará primero al constructor correspondiente de su clase base** para inicializar los atributos de la clase base del O de la clase derivada que se desea crear

Forma Implícita

El constructor de la clase derivada llama **automáticamente** al constructor por defecto de la clase base

Forma Explícita

Se provee al constructor de la clase derivada de un **inicializador de clase base**

Los **destructores**, en general, **son llamados en orden inverso a las llamadas a constructor**, por lo que el destructor de la clase derivada será llamado antes que el destructor de la clase base

Ver ejemplo HerenciaConstDest en Eclipse

Herencia
Cómo usar funciones miembros

```
class ClaseBase{
 …
 public:
    void metodo1();
};
```

Ver ejemplo HERENCIA3 en Eclipse

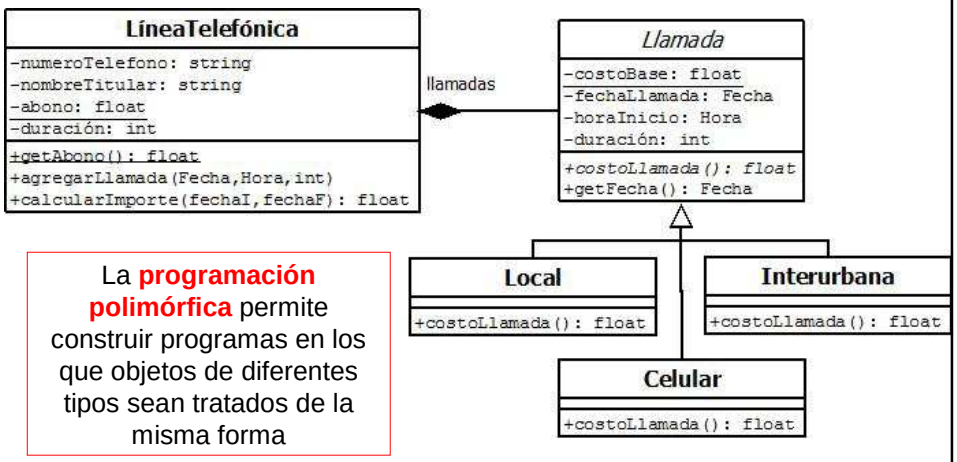
```
class ClaseDerivada:ClaseBase{
 …
 public:
    void metodo1();

    void metodo2(){
      this->metodo1();
       .
       .
       .
      this->ClaseBase::metodo1();
    }
};
```

Una clase derivada puede **redefinir** una función miembro de su clase base con una implementación adecuada

Cuando la función es mencionada por su nombre por un O de la clase derivada, de manera automática, se selecciona la versión de la clase a la que pertenece el O

Se puede acceder a la versión de la clase base usando el operador de resolución de alcance **::**

Factura Telefónica



La **programación polimórfica** permite construir programas en los que objetos de diferentes tipos sean tratados de la misma forma

Ver **FacturaTelefónicaHerencia** en Eclipse

19