1. Applying OOP
2. Summary

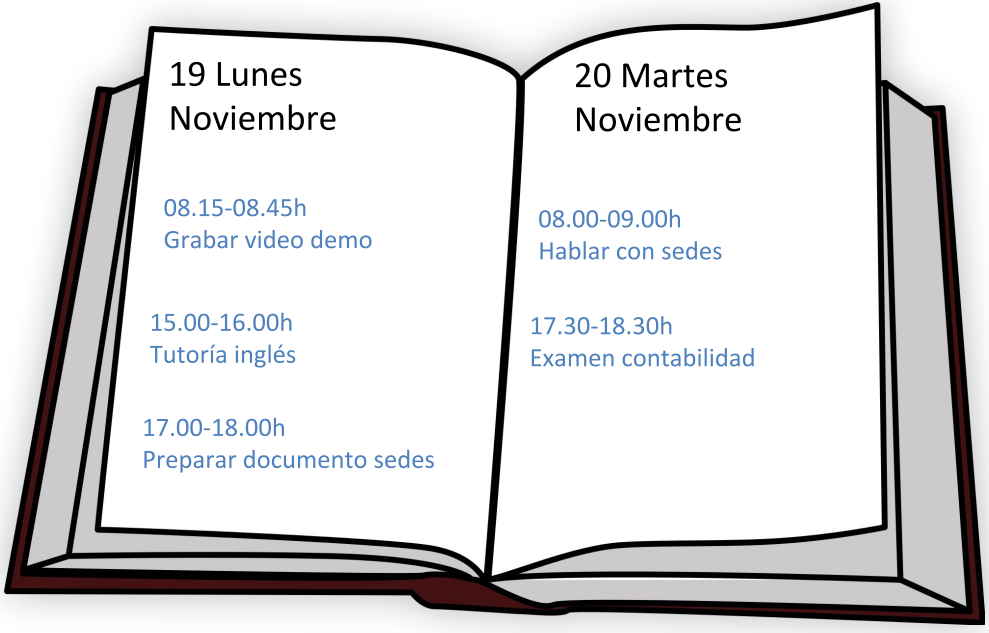
Applying OPP

Applying OPP

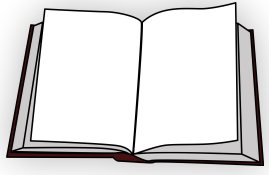
Una aplicación:

Una simulación de un escenario del mundo real, donde los objetos interactúan entre ellos para llevar a cabo una tarea

1. Applying OPP

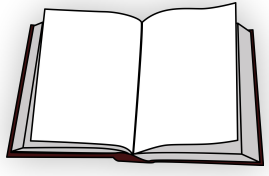


Agenda: mapa de objetos



Agenda: clases

class Agenda	class Pagina	class Cita

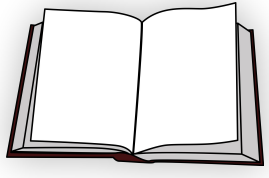


Agenda: clases (fields)

class Agenda
Year anyo Pagina[] paginas

class Pagina
LocalDate fecha boolean esFestivo ArrayList<Cita> citas

class Cita
LocalTime horaInicio LocalTime horaFin String descipcion



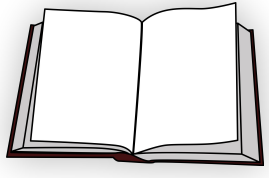
Agenda: clases (fields)

class Agenda
Year anyo Pagina[] paginas

class Pagina
LocalDate fecha boolean esFestivo ArrayList<Cita> citas

class Cita
LocalTime horaInicio LocalTime horaFin String descipcion

java.time

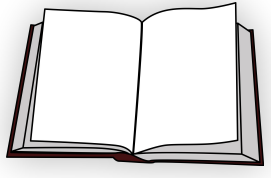


Agenda: clases (fields)

class Agenda
Year anyo Pagina[] paginas

class Pagina
LocalDate fecha boolean esFestivo ArrayList<Cita> citas

class Cita
LocalTime horaInicio LocalTime horaFin String descipcion

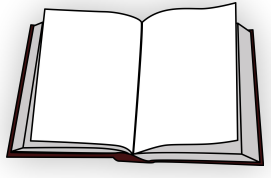


Agenda: clases (fields)

class Agenda
Year anyo Pagina[] paginas

class Pagina
LocalDate fecha boolean esFestivo ArrayList<Cita> citas

class Cita
LocalTime horaInicio LocalTime horaFin String descipcion



Agenda: clases (fields)

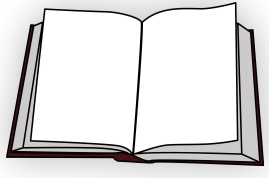
class Agenda
Year anyo Pagina[] paginas

class Pagina
LocalDate fecha boolean esFestivo ArrayList<Cita> citas

class Cita
LocalTime horaInicio LocalTime horaFin String descipcion

Colección estática:
nº de páginas es fijo = 365 o 366

Colección dinámica:
nº de citas es indeterminado

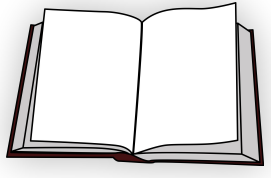


Agenda: clases (methods)

class Agenda
Year anyo Pagina[] paginas
crearCitaEnFecha borrarCitaEnFecha consultarCitasFecha cambiarDescripcionCita

class Pagina
LocalDate fecha boolean esFestivo ArrayList<Cita> citas
crearCita borrarCita listarCitas

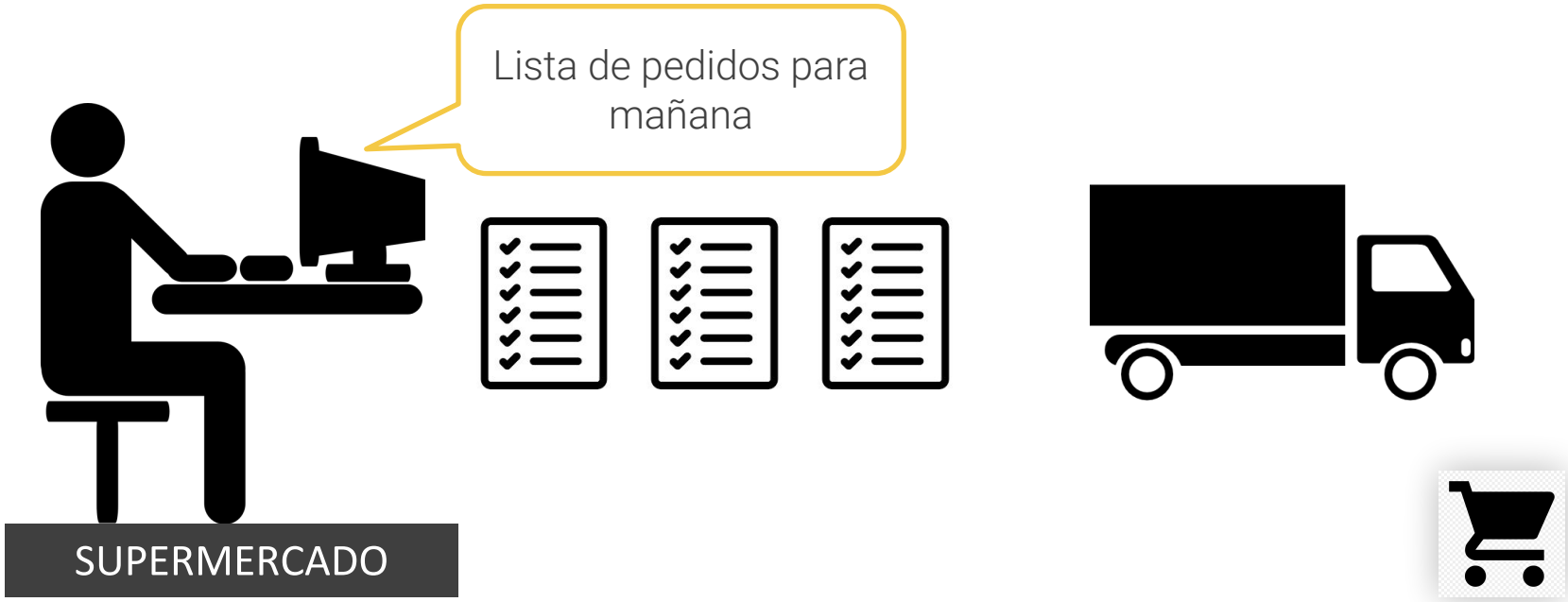
class Cita
LocalTime horaInicio LocalTime horaFin String descipcion
cambiarDescripcion



Pedidos Supermercado

- Datos personales
- Condiciones de entrega: día, hora y dirección
- Lista de productos

Pedidos Supermercado



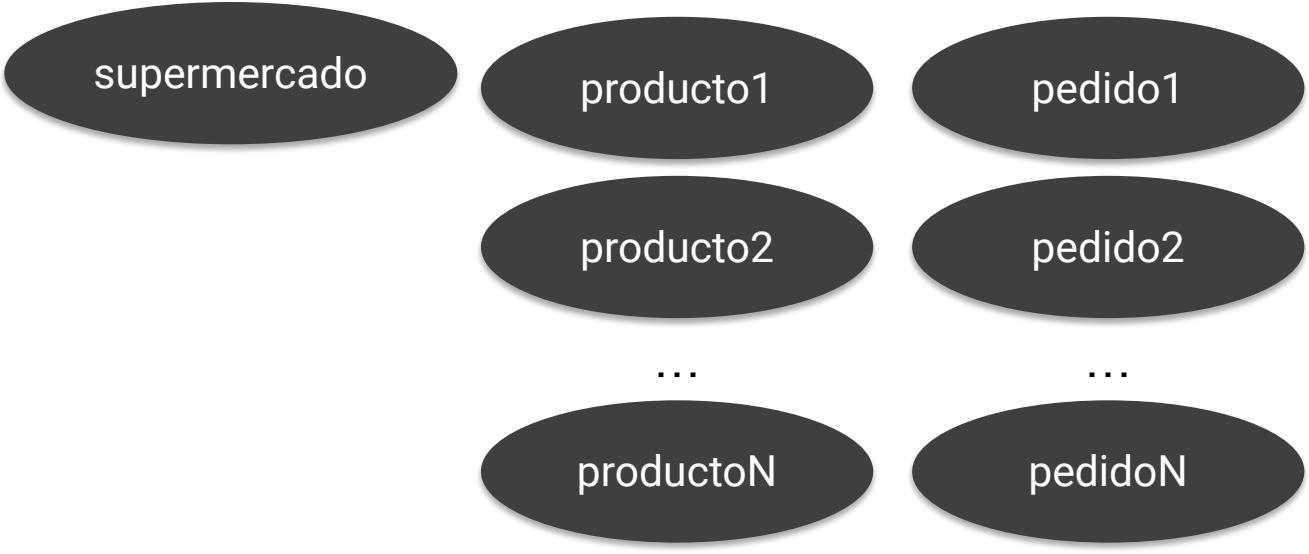
Pedidos Supermercado: descripcion del escenario

Un supermercado quiere crear una aplicación que gestione los pedidos de clientes vía web o teléfono.

Cuando un cliente hace un pedido, se registran sus datos personales y especifica las condiciones de entrega: día y hora, dirección. En el pedido hace constar la lista de productos que quiere que le sirvan.

Al final de la jornada, el supermercado consulta los pedidos para servir los productos a sus clientes y organizar la ruta de los transportistas del día siguiente.

Pedidos Supermercado: mapa de objetos



Pedidos Supermercado: descripcion del escenario

Un supermercado quiere crear una aplicación que gestione los pedidos de clientes via web o teléfono.

Cuando un cliente hace un pedido, se registran sus datos personales y especifica las condiciones de entrega: día y hora, dirección. En el pedido hace constar la lista de productos que quiere que le sirvan.

Al final de la jornada, el supermercado consulta los pedidos para servir los productos a sus clientes y organizar la ruta de los transportistas del día siguiente.

Pedidos Supermercado: clases

class Supermercado

class Producto

class Pedido

Pedidos Supermercado: clases (fields)

class Supermercado

class Producto
String nombre double precio int stock

class Pedido
String nombreCliente String telefonoCliente String emailCliente String direccionCliente LocalDateTime fechaAlta LocalDateTime fechaEntrega ¿? items

Pedidos Supermercado: clases (fields)

class Supermercado

class Producto
String nombre double precio int stock

class Pedido
String nombreCliente String telefonoCliente String emailCliente String direccionCliente LocalDateTime fechaAlta LocalDateTime fechaEntrega ¿? items

Pedidos Supermercado

Cantidad x Producto

SUPERMERCADO S.A.

C/ DUERO Nº 13
TEL: 99 999 999
PEDIDO: dd/mm/yyyy
ENTREGA: dd/mm/yyyy

1	QUESO FRESCO	1,49	1,49
4	TOALLITAS	1,35	5,40
6	AGUA MINERAL	0,60	3,60
12	LECHE ENTERA	0,43	5,16

TOTAL.....EUR. 15,65

Pedidos Supermercado: clases (fields)

class Supermercado

class Producto

String nombre
double precio
int stock

class ItemPedido

int cantidad
Producto producto

class Pedido

String nombreCliente
String telefonoCliente
String emailCliente
String direccionCliente
LocalDateTime fechaAlta
LocalDateTime fechaEntrega
ArrayList<ItemPedido> items

Pedidos Supermercado: clases (fields)

class Supermercado

String nombre
String direccion
String email
String telefono
ArrayList<Producto> productos
ArrayList<Pedido> pedidos

class Producto

String nombre
double precio
int stock

class ItemPedido

int cantidad
Producto producto

class Pedido

String nombreCliente
String telefonoCliente
String emailCliente
String direccionCliente
LocalDateTime fechaAlta
LocalDateTime fechaEntrega
ArrayList<ItemPedido> items

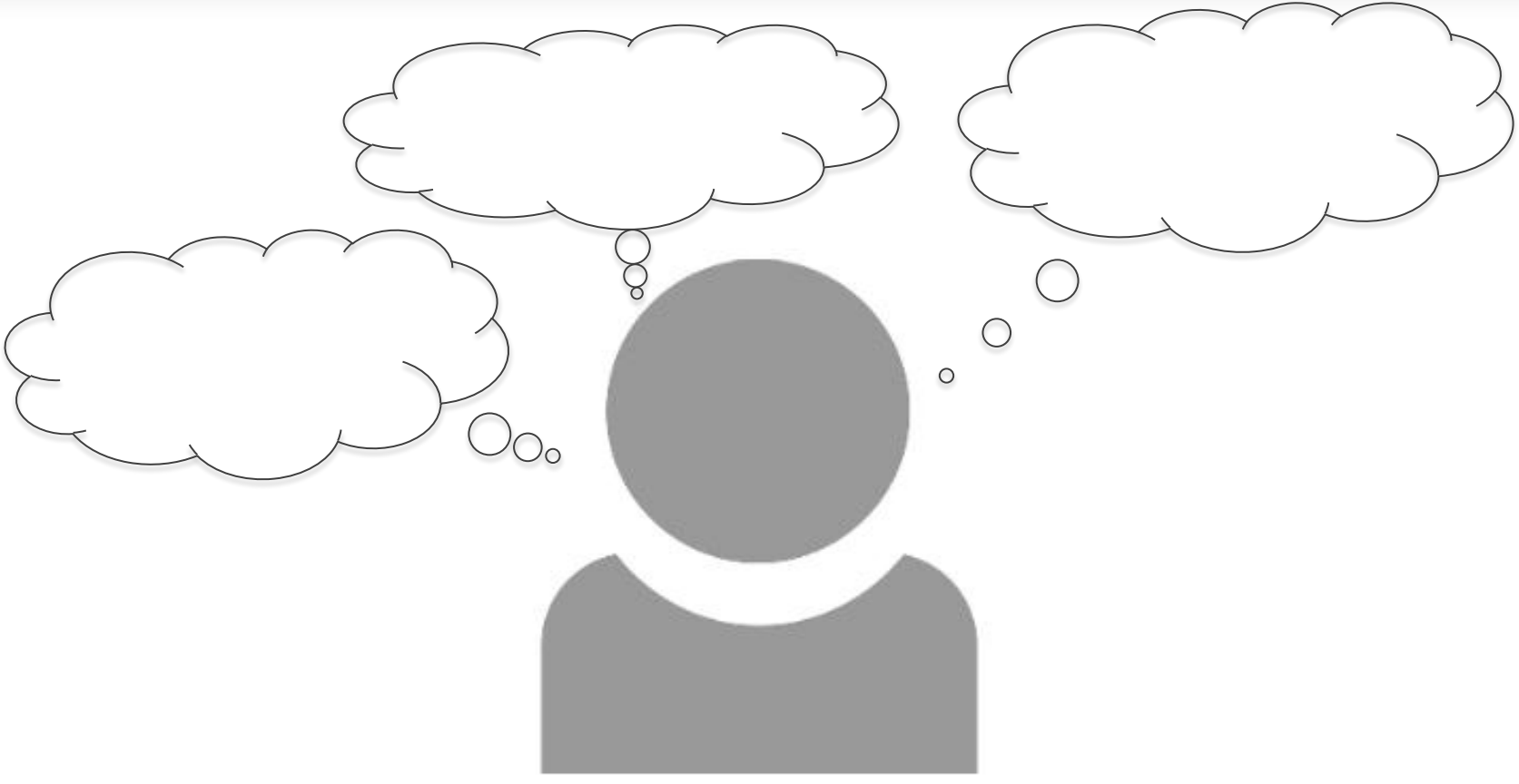
Pedidos Supermercado: clases (methods)

class Supermercado
String nombre String direccion String email String telefono ArrayList<Producto> productos ArrayList<Pedido> pedidos
altaPedido anularPedido listarPedidosFechaEntrega

class Producto
String nombre double precio int stock
actualizarStock

class ItemPedido
int cantidad Producto producto

class Pedido
String nombreCliente String telefonoCliente String emailCliente String direccionCliente LocalDateTime fechaAlta LocalDateTime fechaEntrega ArrayList<ItemPedido> items
precioTotal



Pedidos Supermercado: descripcion del escenario

Un supermercado quiere crear una aplicación que gestione los pedidos de clientes via web o teléfono.

Cuando un cliente hace un pedido, se registran sus datos personales y especifica las condiciones de entrega: día y hora, dirección. En el pedido hace constar la lista de productos que quiere que le sirvan.

Al final de la jornada, el supermercado consulta los pedidos para servir los productos a sus clientes y organizar la ruta de los transportistas del día siguiente.

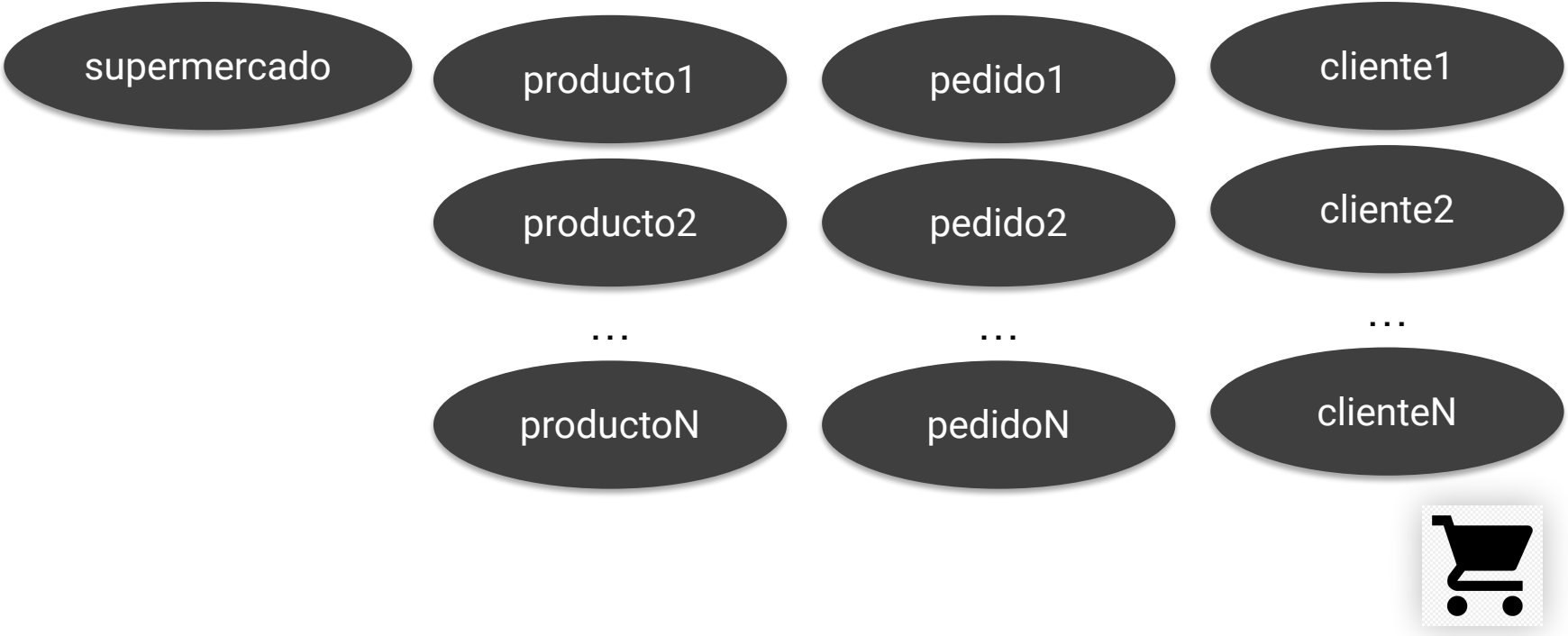
Pedidos Supermercado: descripcion del escenario

Un supermercado quiere crear una aplicación que gestione los pedidos de clientes via web o teléfono.

Cuando un cliente hace un pedido, se registran sus datos personales y especifica las condiciones de entrega: día y hora, dirección. En el pedido hace constar la lista de productos que quiere que le sirvan.

Al final de la jornada, el supermercado consulta los pedidos para servir los productos a sus clientes y organizar la ruta de los transportistas del día siguiente.

Pedidos Supermercado: mapa de objetos



Pedidos Supermercado: clases

class Supermercado

String nombre
String direccion
String email
String telefono
ArrayList<Producto> productos
ArrayList<Pedido> pedidos
ArrayList<Cliente> clientes

class Producto

String nombre
double precio
int stock

class Pedido

~~String nombreCliente~~
~~String telefonoCliente~~
~~String emailCliente~~
~~String direccionCliente~~
Cliente cliente
LocalDateTime fechaAlta
LocalDateTime fechaEntrega
ArrayList<ItemPedido> items

class ItemPedido

int cantidad
Producto producto

class Cliente

String nombre
String direccion
String email
String telefono

Summary

2. Summary

2. Summary

- Describir el escenario

2. Summary

- Describir el escenario
- Localizar los objetos: nombres

2. Summary

- Describir el escenario
- Localizar los objetos: nombres
- Definir clases: objetos del mismo tipo

2. Summary

- Describir el escenario
- Localizar los objetos: nombres
- Definir clases: objetos del mismo tipo
- Definir clases “auxiliares”. Ejemplo: ItemPedido

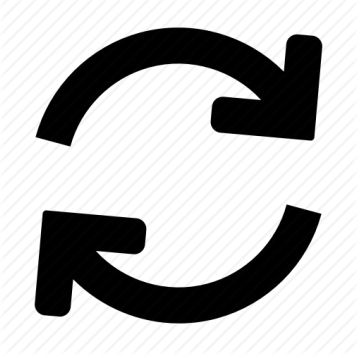
2. Summary

- Describir el escenario
- Localizar los objetos: nombres
- Definir clases: objetos del mismo tipo
- Definir clases “auxiliares”. Ejemplo: ItemPedido
- Para cada clase: identificar
    - *Fields*: propiedades o atributos de los objetos
    - *Methods*: operaciones o acciones a realizar con los objetos

2. Summary

- Describir el escenario
- Localizar los objetos: nombres
- Definir clases: objetos del mismo tipo
- Definir clases “auxiliares”. Ejemplo: ItemPedido
- Para cada clase: identificar
    - *Fields*: propiedades o atributos de los objetos
    - *Methods*: operaciones o acciones a realizar con los objetos
- Para cada clase especificar formalmente:
    - *Fields*: tipo e identificador (ejemplo: double speed)
    - *Methods*: nombre del método, parámetros de entrada, tipo que retorna (ejemplo void seepUp(double increment)

2. Summary



- Describir el escenario
- Localizar los objetos: nombres
- Definir clases: objetos del mismo tipo
- Definir clases “auxiliares”. Ejemplo: ItemPedido
- Para cada clase: identificar
    - *Fields*: propiedades o atributos de los objetos
    - *Methods*: operaciones o acciones a realizar con los objetos
- Para cada clase especificar formalmente:
    - *Fields*: tipo e identificador (ejemplo: double speed)
    - *Methods*: nombre del método, parámetros de entrada, tipo que retorna (ejemplo void seepUp(double increment)

“Da tu primer paso con fe, no es necesario que veas toda la escalera completa, sólo da tu primer paso.“

Martin Luther King, activista estadounidense conocido por luchar pacíficamente contra la segregación y discriminación racial.