Systems programming

Object Oriented Programming

I. Object ***Based*** Programming

II. Object ***Oriented*** Programming

Telematics Engineering

M. Carmen Fernández Panadero

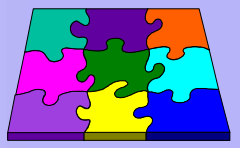
<mcfp@it.uc3m.es>

Scenario IV:
Declare and implement a class

- Now that you know how to read code and implement your own methods you will have to design a new class in order to create a new data type with its characteristics and behavior.

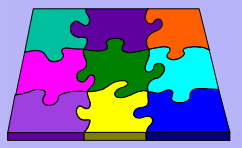
- Objective:
  - Be able to **declare a class** with a set of characteristics (**attributes**) and behaviour (**methods**)
  - Be able to **create objects** and modify or restrict access to their state and their behavior
- Workplan:
  - Memorize the basic **nomenclature** of the object-oriented programming
  - Practice **modeling objects** with simple examples to distinguish between a class, an object, its state and behavior
  - Review the **Java syntax** for declaring **class attributes, constructors** and **methods**
  - Review the mechanism and syntax for **message passing** between objects

Contents



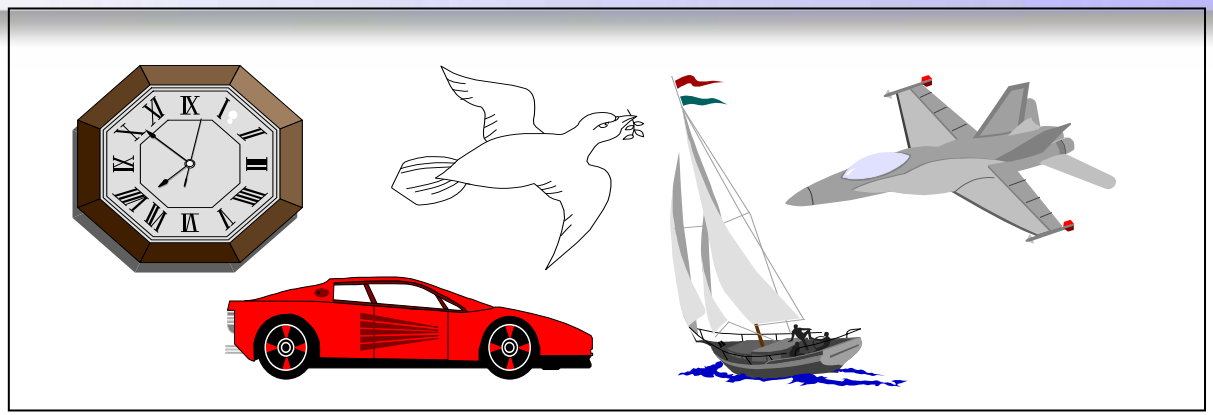
- Classes and Objects
- Object encapsulation
  - Functional abstraction
  - Data abstraction
- Class members (attributes and methods)
- Message passing
- Constructors
- Overloading

Objectives



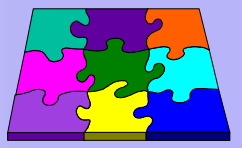
- Define ***basic concepts*** of object ***based*** programming
  - Classes, objects
  - Members (variables, methods)
  - Abstraction and shadowing of information
- Describe ***relationship*** between object and class
- ***Create*** a simple object and be able to ***model:***
  - its atributtes (with variables)
  - its behaviour (with methods)

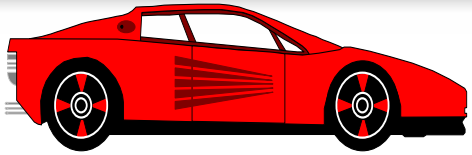
What is an object?



- **Objects** are (simple/complex) (real/imagined) representations of things: clock, airplane, employer, etc.
- Not everything can be considered as an object, some things are simply features or **attributes** of objects: color, speed, etc.

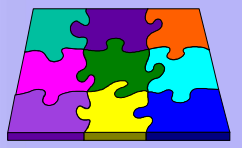
What is an object?





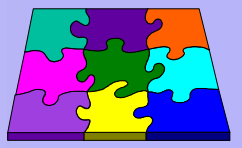
- ***Functional Abstraction***
  - Things that we know that cars do (but we do not know how:
    - advance
    - stop
    - turn right
    - turn left
- ***Data abstraction***
  - A car has certain attributes too:
    - color
    - speed
    - size
    - etc..
  - The way in which attributes are defined is not important for the design

What is an object?



- It is a way to group a set of data (**state**) and functionality (**behavior**) in the same block of code that can then be referenced from other parts of a program
- The **class** which the object belongs to can be considered as a new **data type**

Object encapsulation



- ***Encapsulation:*** explains the links between **behavior** and **state** to a particular object
- ***Information hiding***: Define which parts of the object are visible (the public interface) which parts are hidden (private)

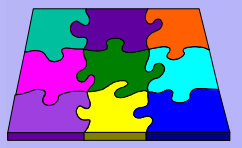
• *The ignition is a **public interface** mechanism to start a vehicle*

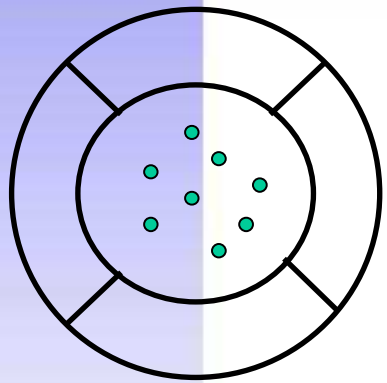
• *The implementation of how to really start a car is **private.** We only can access this information introducing the key into the ignition*

Adventages

The object may change but its public interface remains compatible with the original.This fact facilitates reuse of code

Object encapsulation





CLASS MEMBERS

Objects encapsulate attributes, allowing access to them only through methods

- **Attributes (Variables)**: Containers of values
- **Methods**: Containers of functions

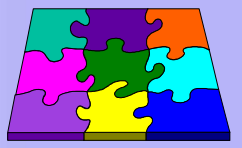
An object has

- **State**: represented by the values of its attributes
- **Behaviour**: defined by its methods

Usually:

- Methods are public
- Attributes are private
- There can be private methods
- It is dangerous to have public attributes

Object Definition



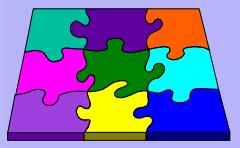
Public Members

- Public members (describe **what** an object can do)
  - What the object can do (methods)
  - What the object is (its abstraction)

Private Members

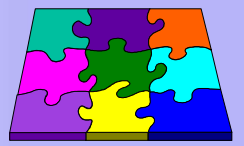
- **How** the object do its work (how it is implemented).
  - For example. the ignition key interacts with the electric circuit of the vehicle, the engine, etc.
  - ***In pure object-oriented systems, state is completely private and can only be modified through the public interface***.
  - Eg: public method stop can change the value of the private attribute speed.

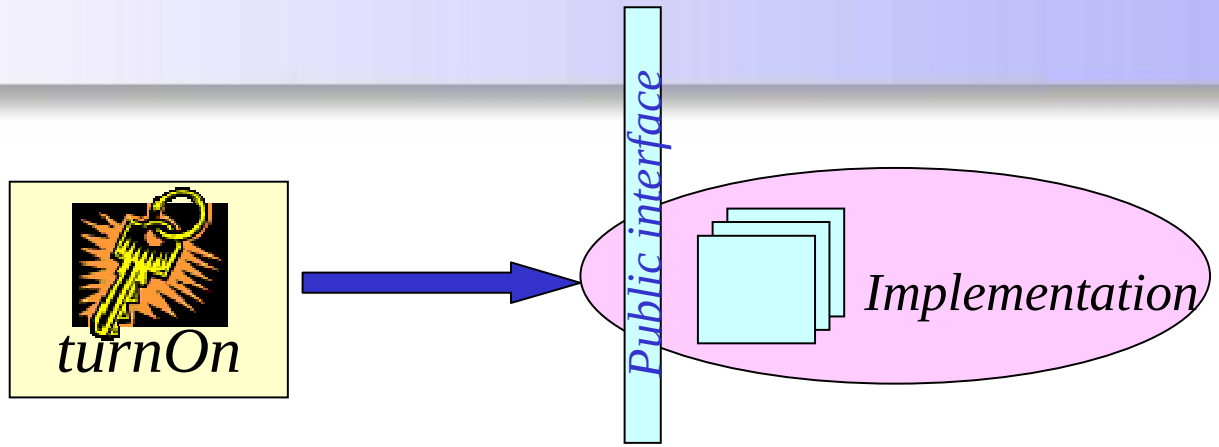
Interactions between objects



- **Object modelling** describes:
  - Objects and
  - Their interrelations
- To do a task, an object can **delegate** some work to another object, that can be part of itself, or can be any other object in the system
- Objects interact with each other sending **messages**

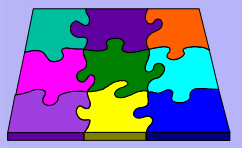
Message passing

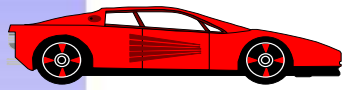




- An object sends a **message** to another object
  - By **calling** a method (method call / method invocation)
- Messages are handled by the **public interface** of the receiving object
  - We can only call methods from another object that are **public** or **accessible** from the calling object
- The receiving (called) object will react:
  - **Changing its state** (i.e. modifing its attributes) and/or
  - **Sending other messages** (i.e. calling other (public or private) methods from the same object (from himself) or calling other methods from other objects (public or accessible from that object)

Classification of objects



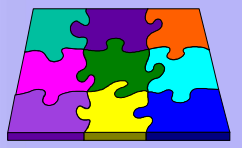



- **Class**: Set of objects with similar states and behavior
  - We can refer to the class “Car” (any instance in the classification of cars) r1
- “My car” is an **object**, i.e. a particular **instance** of the class Car
- How to classify depends on the problem to be solved

Diapositiva 13

r1 Esto es confuso: las instancias son objetos y aquí se define la clase como instancia.
rcrespo; 31/01/2011

Objects vs. Classes



A **class** is an abstract entity

- It is a kind of data classification
- Defines the behaviour and attibutes of a group of objects with similar structure and similar behaviour

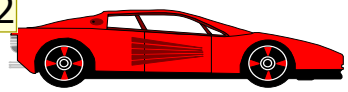
Class car
Methods: turn on, advance, stop, ...
Atributtes: color , speed, etc.

→ Class name
→ Methods (functions)
→ Attributes (data)

An **object** is an instance of a class

- An object can be distinguished from other members of the class by the value of its attributes

Object MyFerrari r2
Belongs to the class Car



Name: MyFerrar r3
Methods: turn on, advance, stop, ...
Attributes : color = "red";
speed = 300Km/h

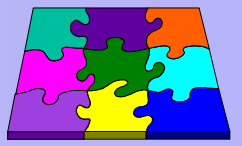
- A **class** is declared, an **object** is also created

Diapositiva 14

r2 mejor un Ferrari concreto y no "cualquier Ferrari" (que podría confundirse con la clase Ferrari que hereda de Car)
rcrespo; 31/01/2011

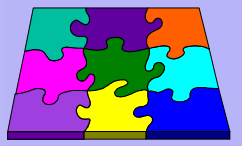
r3 ver comentario anterior
rcrespo; 31/01/2011

Constructors
Ideas to recall



- When an object is created, its members are ***initialized*** using a constructor method
- Constructors:
  - Have the ***same name*** as the class
  - They have ***no return type*** (not even void)
- At least 1 constructor is recommended to exist
- Several constructors can exist that are distinguished by their parameters (***overloading***)
- A ***default constructor*** without parameters is created if no explicit constructors are defined, that initializes the attributes to their default values.
- If there is a constructor in the class, the default constructor no longer exits. In that case, if a constructor without parameters is desired, it needs to be explicitly declared.

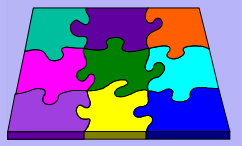
Overloading
What is it?



- Two methods with the ***same name*** can be defined in a class if they have ***different parameters.***
- It is widely used for constructors.
- The method actually executed depends on the parameters passed when it is called..
- In this case, ***no*** information ***hiding*** exists, both methods can be accessed.

Overloading

What is it used for?



Classroom
- name
- description
- location
- printName()
- printDescription()
- printDescription(String furniture)
- printLocation()

*Although they have **equal names**, they are two different methods, because they have **different parameters***

*They have **different functionality** as shown in the example*

describe the classroom in general

describe the furniture inside the classroom that is passed as a parameter