# Object

# Object

When developing software you need to structure your code. One way is to think of things that exist in your system, such as a customer or an audio file.

The descriptions of these are abstractions of (part of) the problem.

One class of things your system deals with could be audio files for an audio player. One specific audio file can be called an object, which is an instance of the class audio file.

# Object

Grouping of data and operations/behaviour.

# Object - data

Grouping of **data** and operations/behaviour.

Examples of data:

- Length
- birthYear

# Object - data

Grouping of **data** and operations/behaviour.

Data is often hidden from the user (programmer) of the object.

# Object - operations/behaviour

Grouping of data and **operations/behaviour**.

“What can the object do?”

Examples of operations:

- age (how old is the object? Calculated from birthYear)
- move (move a character in a video/computer game)

# Object - operations/behaviour

Grouping of data and **operations/behaviour**.

**Operations/behaviour** are often available to the user (programmer). Usually referred to as the publicly available interface of the object. With interface we mean the way we communicate with the object.

# Object

The point of using objects is that we can focus on how to communicate with the object and let the objects handle their own data (state) in a way we don’t have to know.

The communication with the objects is sometimes called “To send messages” to the objects.

# Example messages

Say that we have an object representing an audio file. We could ask the object to “play” or “stop”. Or “changeVolume”.

When asked to “changeVolume”, the data representing the volume changes.

Sometimes this is described as “the state of the object changes”.