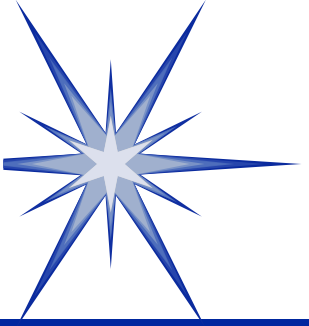


Object-Orientation

by Derek Peacock

Types of Objects

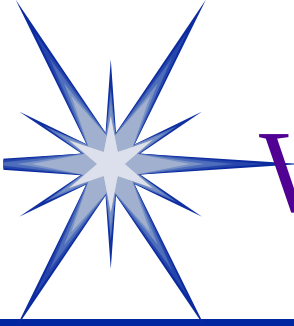
- Tangible things (Entities)
  - Cars, Cards, Books
- Role
  - Employer, Teacher, Student
- Incident
  - Flight, Purchase, Transaction

Types of Objects cont...

- Interactions
  - Electrical Circuit, a Contract
- Specifications
  - Architectural Design
  - Engineering Blueprint

Objects in Programming

- Direct representation of the application
  - Graphical(Fox, Rabbit)
  - Documents
  - Accounts
- Artefact of Implementation
  - List, Array
  - Stacks, Queue

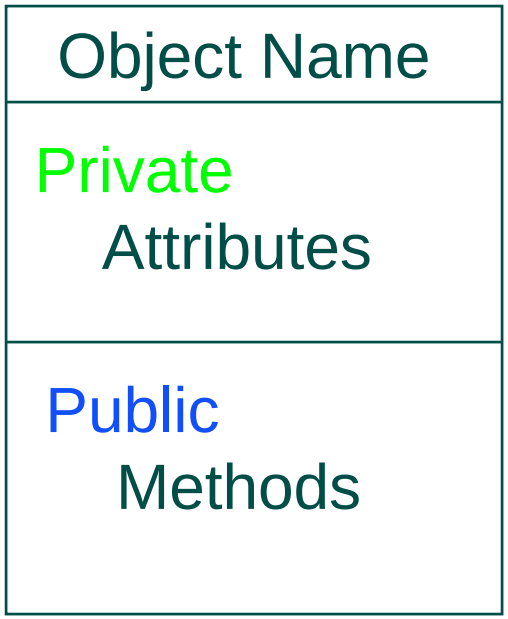


What are Objects?

- Data: Property, Attribute, State
- Function: Methods, Operations, Behaviour
- Composed of other Objects

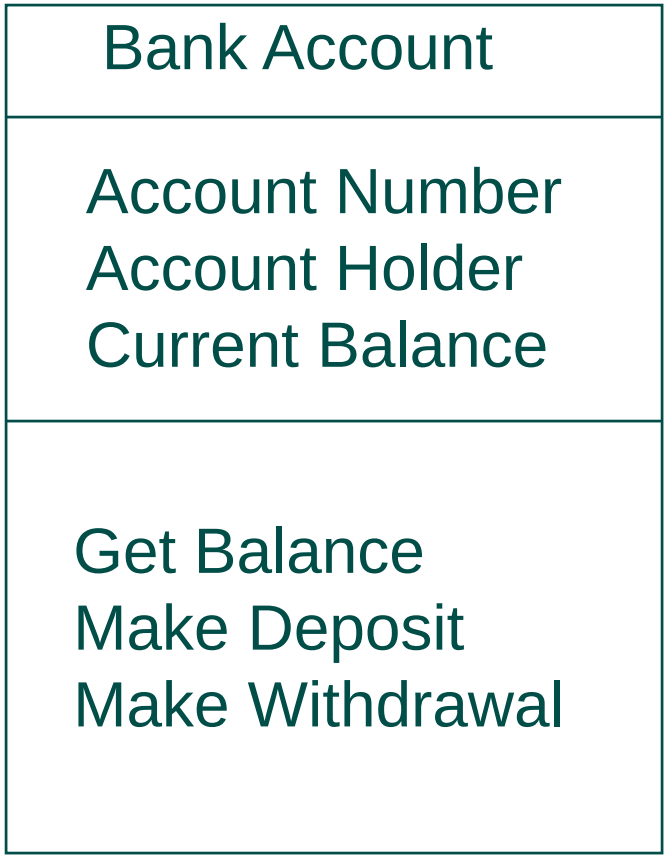
Program Objects

Object = (private) Data + (public) Operations



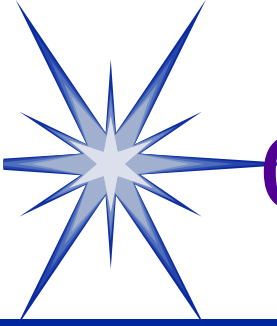
Encapsulation

- Data & Processes linked
- Public Interface
- Information is Hidden
- Private Data
- Private internal processes



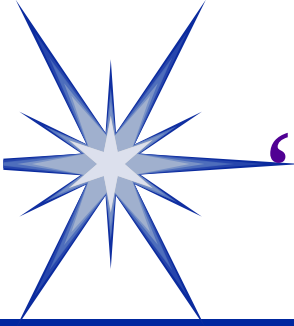
Traditional Modularity

Process Driven	Data Driven
Functional Decomposition	Data Decomposition
Data structured to suite the process	Process constrained by the structure of data
code re-use difficult	code re-use difficult



Object Oriented Modularity

- Data & Process tightly bound together (**encapsulated**)
- Internal data and processes hidden
- Public interface can remain fixed while internal workings are changed
- Can be extended without altering original code (**Inheritance**)



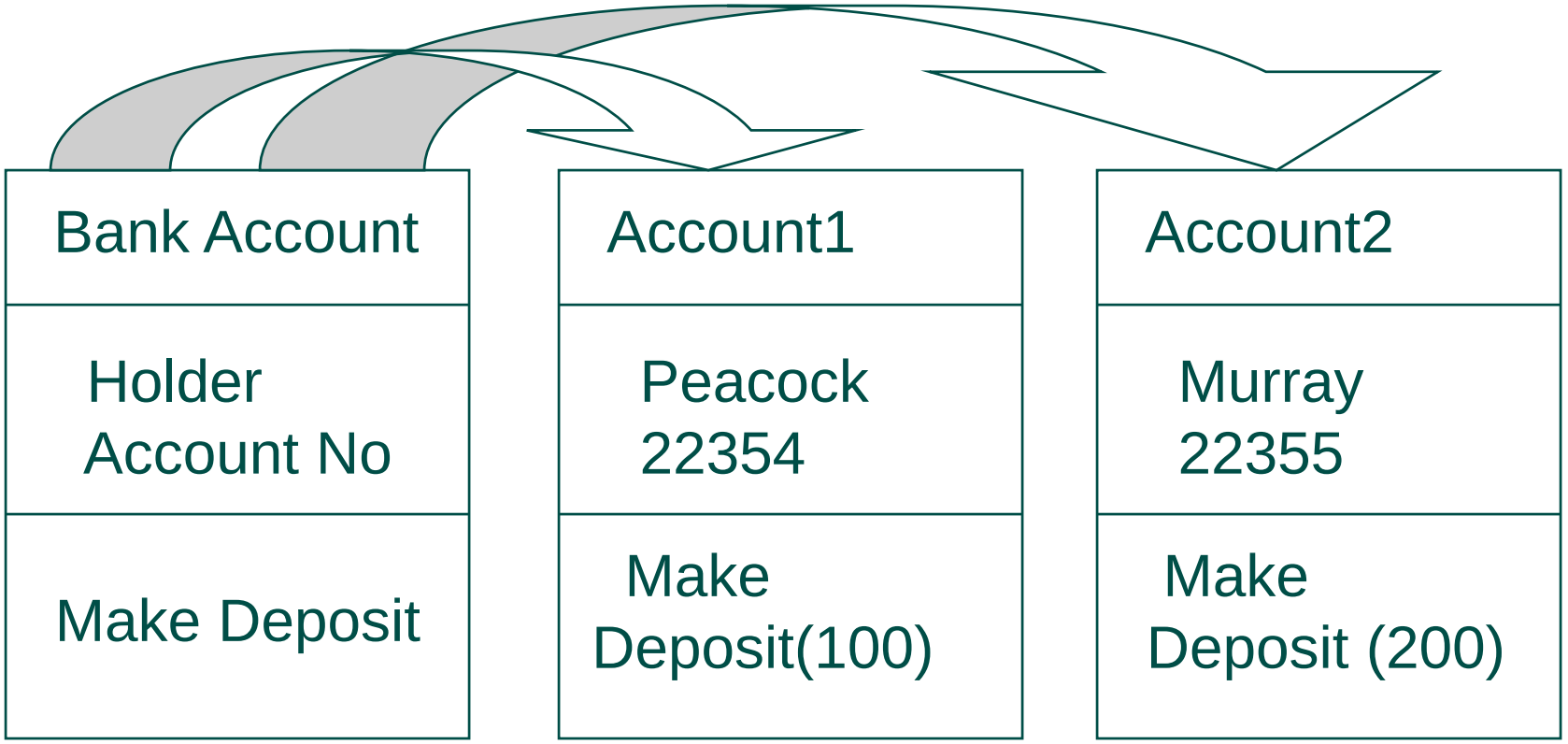
‘Good’ Modules (Classes)

Reusability	Software components
Extensibility	New components from old
Decomposability	Breaking a system down
Composability	Building a system up
Understandability	Easy to understand the part
Continuity	Small changes lead to small effects
Protection	Errors confined to module

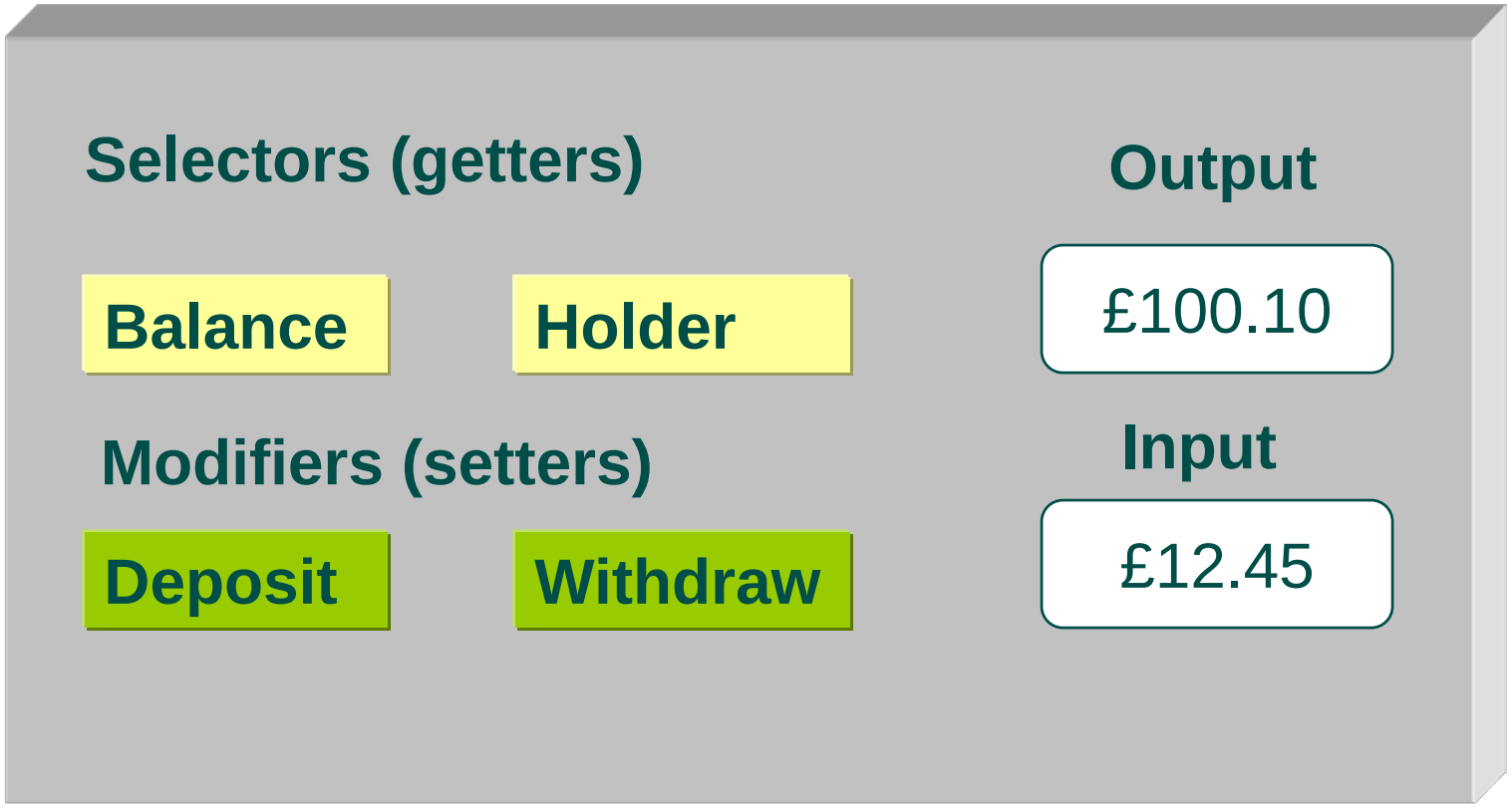
Features of OOP

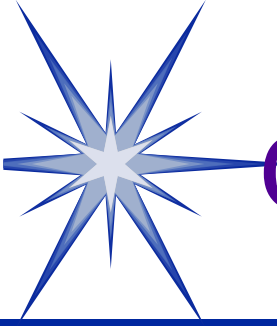
- Encapsulation/Abstraction
- Inheritance (Generalisation)
- Polymorphism (multiple forms)
- Aggregation/Composition

Objects are Instances of Classes



Finite State Machine





Construction & Destruction

- The Class always exists
- Objects are created when needed and destroyed when not needed
- A Class has a unique name, attributes and methods
- All objects of the same class have the same attributes and methods, but the attributes can have different values