



Object-Oriented Programming

An Overview



Contents

- OOP and software productivity
- Features and characteristics of OOP
- Major concepts of OOP
- OOP languages
- OOP and traditional design
- Advantages and disadvantages



What is Object-Oriented Programming

- A well-regarded and widely accepted programming technology
- Potential for much improved productivity
- Uses interacting program objects
- Objects are independent entities
- Objects respond to messages



Software Productivity Factors

- Modularity (separation of duties)
- Extensibility (responsive to future requirements)
- Modifiability (easy to make small changes)
- Flexibility (not cast in concrete)
- Maintainability (big savings)
- Reusability (don't reinvent the wheel)



Software Engineering Techniques

- Data abstraction (hidden data representation)
- Program encapsulation (operations married to data)
- Software libraries (fixed reusable)
- Reusable Objects (flexible, reusable)
- Polymorphism (type-related generic operations)
- Maintenance automation



OOP Central Concepts

- Data abstraction
- Encapsulation
- Classification
- Inheritance
- Polymorphism



OOP Characteristics

- Class definitions
- Inheritance and class hierarchy
- Operator and Function overloading
- Generic classes
- Class libraries



OOP Languages

- Simula
- Modula
- Smalltalk
- Ada
- Objective-C
- CLOS (Common Lisp Object Standard)
- Standard C++
- Java
- Scripting languages: Perl, Javascript, Python



Traditional vs. OOP

- Procedural Programming :

data structures + algorithms = Program

- OOP :

objects + messages = Program

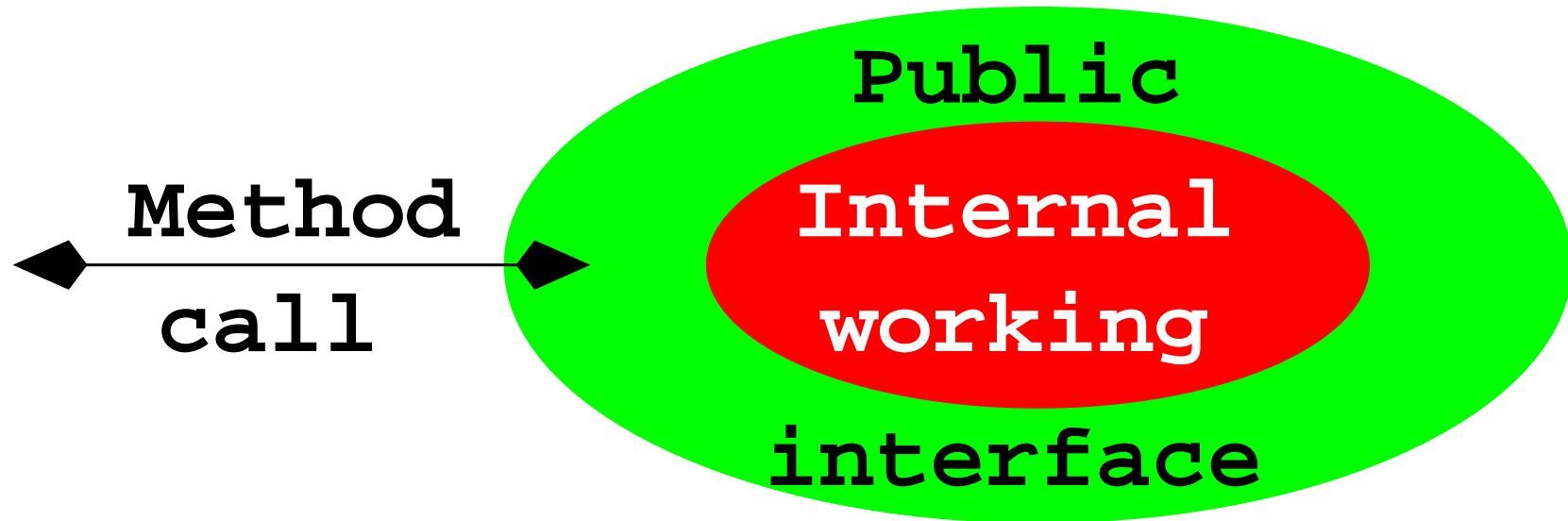


Class Definition

- Blueprint for building Objects
- Members: methods and fields
- private and public
- API—application programming interface
- relation with other classes

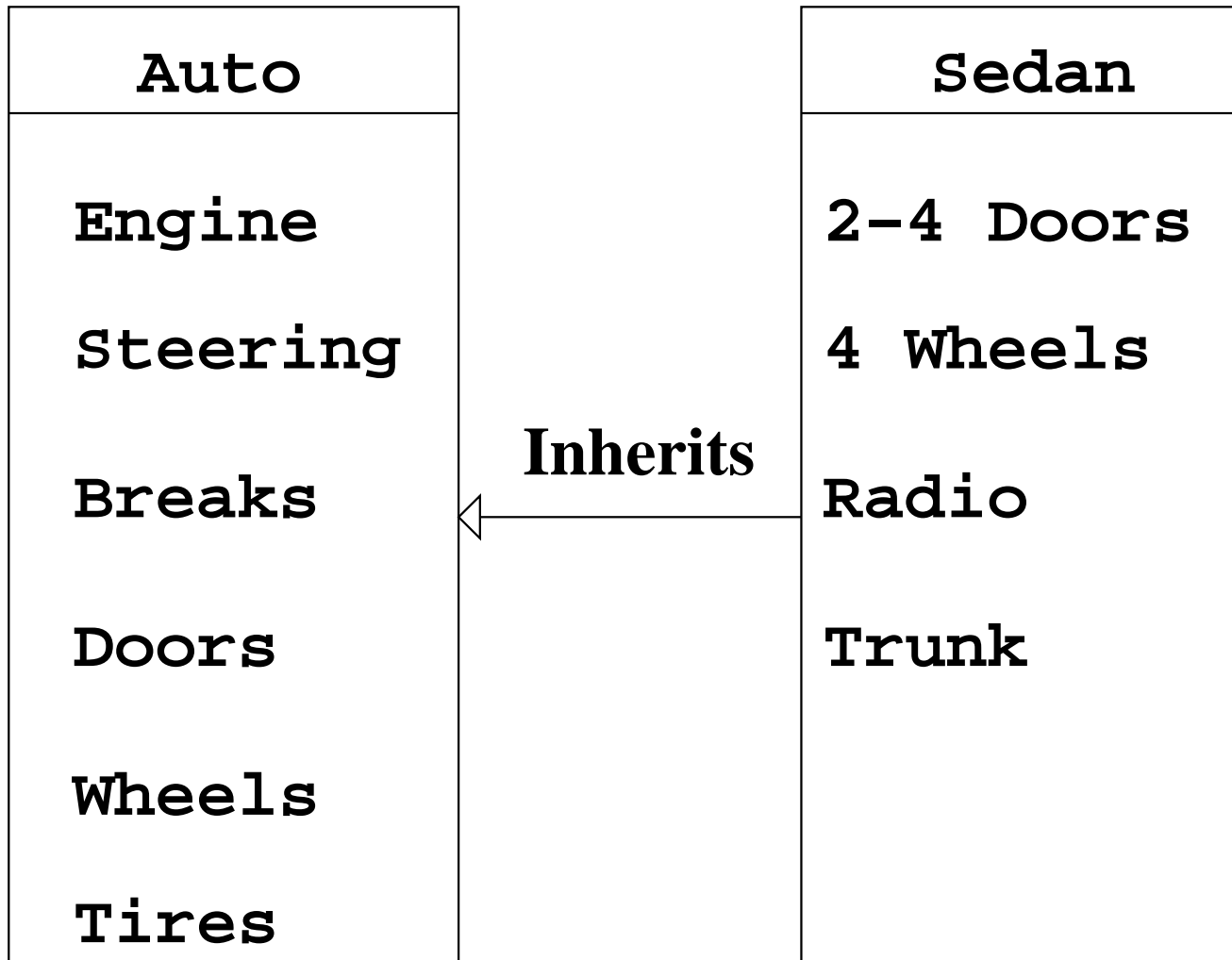


An Object



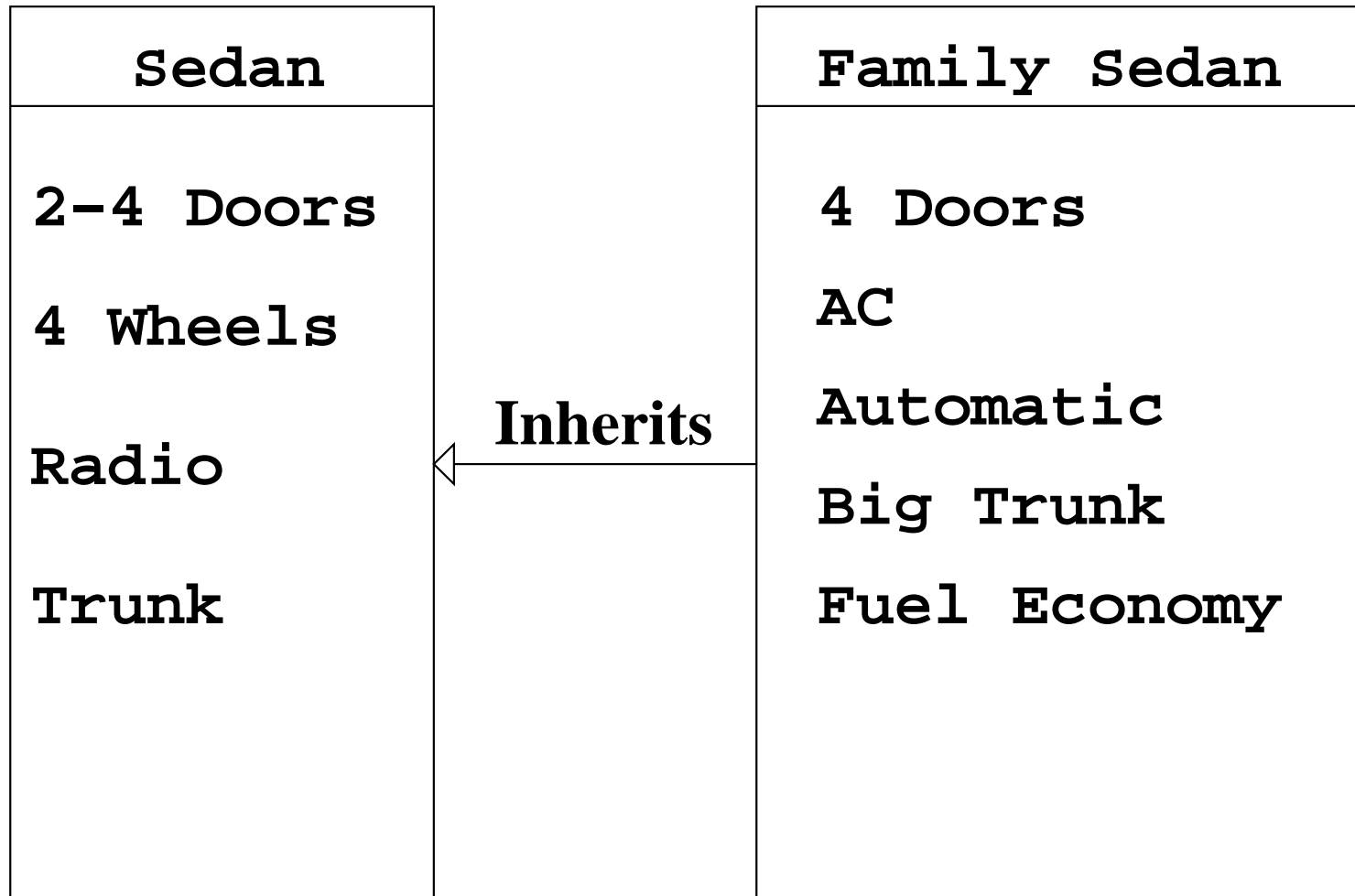


Inheritance Example



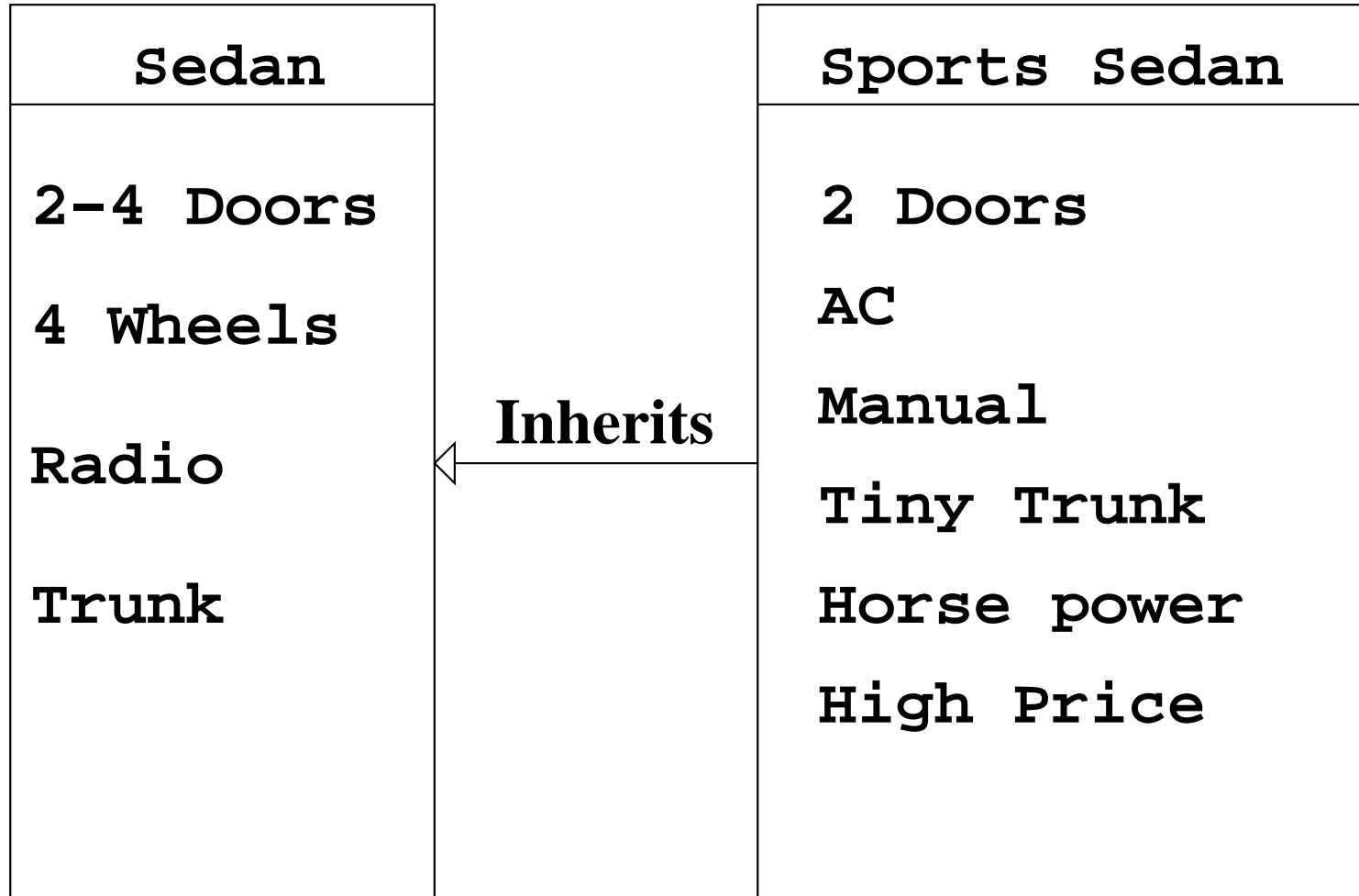


Inheritance Example





Inheritance Example





Inheritance Concepts

- Deriving new classes from old ones
- Single inheritance
- Partial inheritance
- Multiple inheritance
- Inheritance chain



Generic Classes

- Class definition for unspecified data
- Container class
- Flexible and reusable



Reusability Considerations

- Pipes / filter (ala UNIX): slow, restrictive
- Subroutine libraries: fast, inflexible
- Object libraries: flexible, and efficient



Design Approaches

- Ways to Break The Problem Down
- Procedural decomposition
- Data decomposition
- Object-oriented decomposition



Procedural Decomposition

- Also known as functional or traditional design
- Decompose the solution into major steps
- Decompose each major step further
- Decomposition procedural-oriented



Traditional Design Disadvantages

- Data and operations are separated
- No data abstraction or info hiding
- Not responsive to changes in problem space
- Inadequate for concurrent problems



Object-oriented Design Principle

- Identify interacting objects
- Characterize each object, establish attributes
- Identify the data and operations within each object
- Identify requests answered by each object
- Identity services required of other objects
- Establish relationships to other objects
- Group similar objects together
- Implement common super classes
- Implement different objects as classes



Advantages

- Responsive to changes
- Encapsulation
- Simplify Testing, debugging
- Easy to understand
- Avoid reinventing the wheel
- Easier to manage, to maintain
- Off-the-shelf software



Potential Disadvantages

- Over generalization
- Artificial class relations
- Unnecessary complications