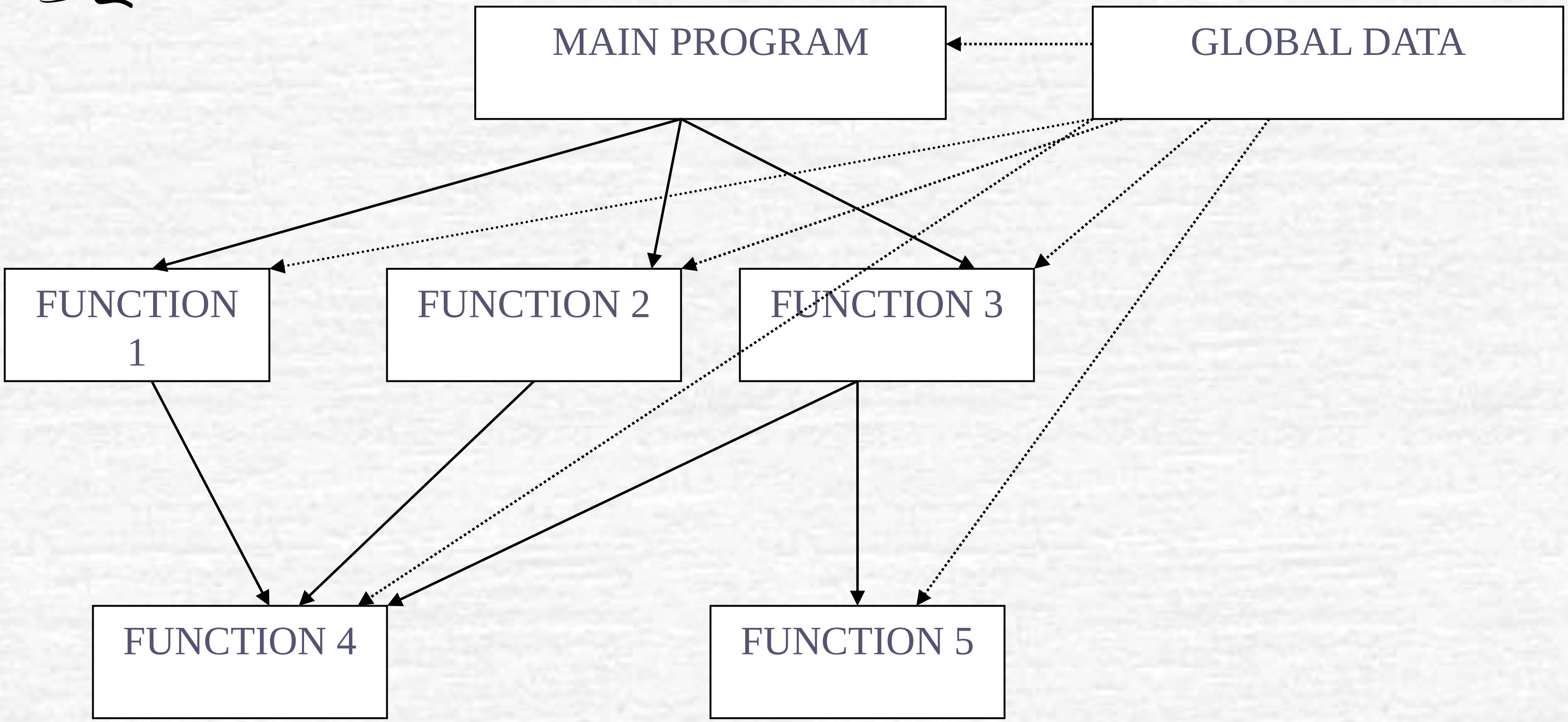
INTRODUCTION TO OOP

Objective:

- Know the difference between functional programming and OOP
- Know basic terminology in OOP
- Know the importance of OOP
- Know four design principles of OOP
- Know OOP programming languages

STRUCTURED vs. OO PROGRAMMING

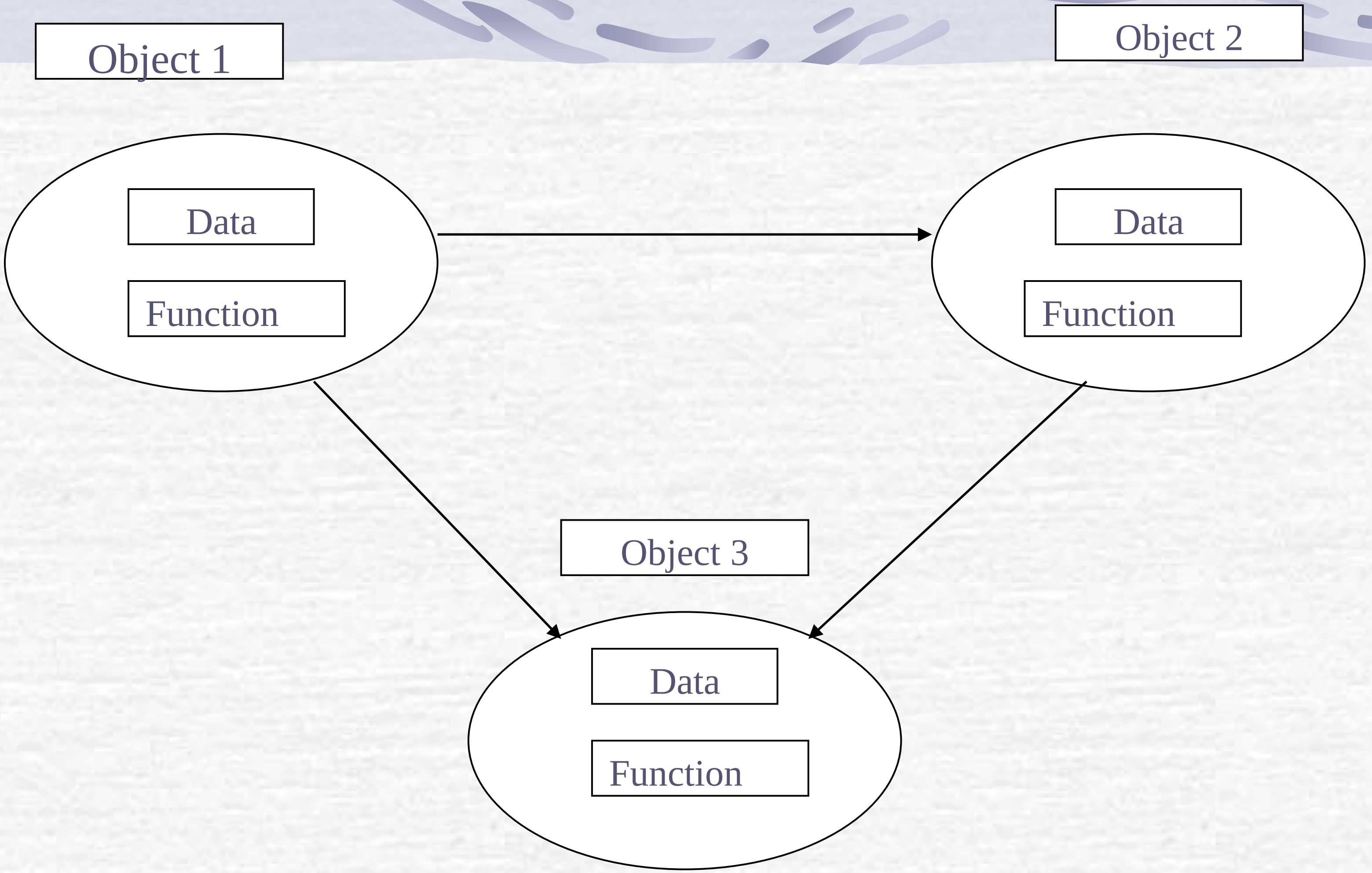
STRUCTURED PROGRAMMING:



Structured Programming

- Using function
- Function & program is divided into modules
- Every module has its own data and function which can be called by other modules.

OBJECT ORIENTED PROGRAMMING



OBJECT ORIENTED PROGRAMMING

- Objects have both data and methods
- Objects of the same class have the same data elements and methods
- Objects send and receive *messages* to invoke actions

Key idea in object-oriented:

The real world can be accurately described as a collection of objects that interact.

Basic terminology

- *object*

- usually a person, place or thing (**a noun**)

- *method*

- an action performed by an object (**a verb)**

- *attribute*

- description of objects in a class

- *class*

- a category of similar objects (such as *automobiles)*

- does not hold any values of the object’s attributes

Example for attributes and methods

Attributes:

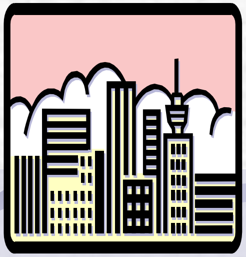
- manufacturer’s name
- model name
- year made
- color
- number of doors
- size of engine
- etc.

Methods:

- Define data items (specify manufacturer’s name, model, year, etc.)
- Change a data item (color, engine, etc.)
- Display data items
- Calculate cost
- etc.

Why OOP?

- Save development time (and cost) by reusing code
    - once an object class is created it can be used in other applications
- Easier debugging
    - classes can be tested independently
    - reused objects have already been tested



Design Principles of OOP

Four main design principles of Object-Oriented Programming(OOP):

- Encapsulation
- Abstraction
- Polymorphism
- Inheritance

Encapsulation

- Also known as *data hiding*
- Only object’s methods can modify information in the object.

<u>Analogy:</u>

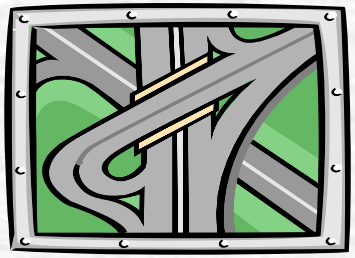
- ATM machine can only **<u>update accounts</u>** of one person or object only.

Abstraction

- Focus only on the important facts about the problem at hand
- to design, produce, and describe so that it can be easily used without knowing the details of how it works.

Analogy:

- When you drive a car, you don’t have to know how the gasoline and air are mixed and ignited.
- Instead you only have to know how to use the controls.
- Draw map

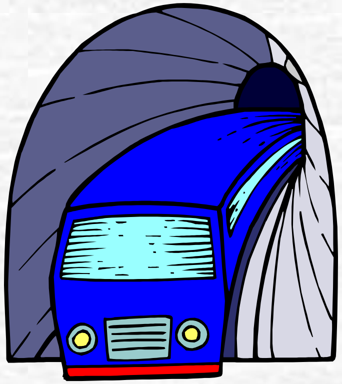


Polymorphism

- the same word or phrase can mean different things in different contexts

Analogy:

- In English, **bank** can mean side of a river or a place to put money
- move -



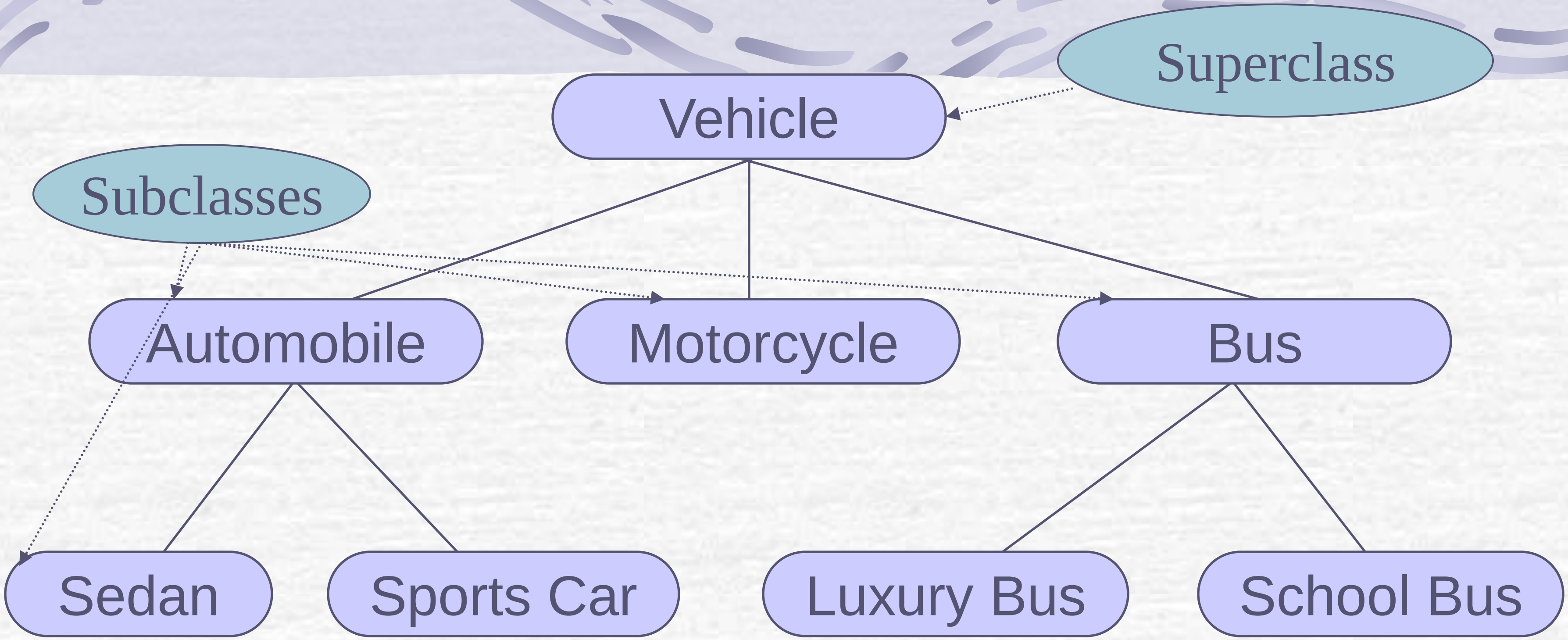
Function Overloading

- The operation of one function depends on the argument passed to it.
- Example: Fly(), Fly(low), Fly(150)

Inheritance

- *Inheritance*—a way of organizing classes
- Term comes from inheritance of traits like eye color, hair color, and so on.
- Classes with properties in common can be grouped so that their common properties are only defined once.
- *Superclass* – inherit its attributes & methods to the subclass(es).
- *Subclass* – can inherit all its superclass attributes & methods besides having its own unique attributes & methods.

An Inheritance Hierarchy



What properties does each vehicle inherit from the types of vehicles above it in the diagram?

Object-Oriented Programming Languages

- Pure OO Languages

  Smalltalk, Eiffel, Actor, Java

- Hybrid OO Languages

  C++, Objective-C, Object-Pascal

Review: Introduction to Object Orientation

- What are the four basic principles of object orientation? Provide a brief description of each.
- What is an Object and what is a Class? What is the difference between them?
- What is an Attribute?
- What is an Operation?
- What is inheritance?
- What is polymorphism?
- Describe the strengths of object orientation.

Review: Introduction to Object Orientation

- State 2 differences between functional programming and OOP.
- What are the four basic principles of object orientation? Provide a brief description of each.
- What is an Object and what is a Class? What is the difference between them?
- What is an Attribute?
- What is an Operation?
- Describe the strengths of object orientation.