OBJECT ORIENTED PARADIGM

Lesson # 01

Dr. Lasanthi De Silva

PROGRAMMING INNOVATIONS

Improved Software Development Processes

The Unified Modeling Language (UML)

Object-Oriented Programming (OOP)

PROCEDURAL LANGUAGES

- The problem is viewed as a sequence of things.
- A list of instructions.
- Example: C, Pascal, FORTRAN
- Each statement in the language tells the computer to do something:
  - Get some input,
  - add these numbers,
  - divide by six, and
  - display that output.

What happens when the Program becomes Larger?

ACTIVITY

Write a simple C program that takes two integers as input, adds them, and then prints the result.

```c
#include <stdio.h>

int main() {
    // Declare variables
    int num1, num2, sum;

    // Input
    printf("Enter first number: ");
    scanf("%d", &num1);

    printf("Enter second number: ");
    scanf("%d", &num2);

    // Add the numbers
    sum = num1 + num2;

    // Output
    printf("Sum: %d\n", sum);

    return 0;
}
```

PROCEDURAL LANGUAGES

- A program in a procedural language is a list of instructions.
- Example: C, Pascal, FORTRAN
- Each statement in the language tells the computer to do something:
  - Get some input,
  - add these numbers,
  - divide by six, and
  - display that output.
- A very small program
  - The programmer creates the list of instructions, and the computer carries them out.
  - No other organizing principle (a paradigm) is needed.

What happens when the Program becomes Larger?

DIVISION INTO FUNCTIONS

- When the program becomes Complex
  - The need arises to comprehend
- A grouping of components that execute lists of instructions
- Dividing a program into functions and modules (Subroutines, subprograms, procedures)
- Each function has a clearly defined purpose and a clearly defined interface to the other functions in the program
- Functions can be further grouped together to form Modules

ACTIVITY

Re-factor the C program including functions.

```c
#include <stdio.h>

// Function prototype
int addNumbers(int a, int b);

int main() {
    // Declare variables
    int num1, num2, sum;

    // Input
    printf("Enter first number: ");
    scanf("%d", &num1);

    printf("Enter second number: ");
    scanf("%d", &num2);

    // Function call to add numbers
    sum = addNumbers(num1, num2);

    // Output
    printf("Sum: %d\n", sum);

    return 0;
}

// Function definition to add two numbers
int addNumbers(int a, int b) {
    return a + b;
}
```

PROGRAMMING PARADIGMS

- The way you write the program
- Procedural vs Object Oriented Programming
- Procedural Programming
  - A paradigm that emphasizes on procedure
    - Divide the program into series of steps.
    - Called based on the concept of procedure calls.
    - Procedures - routines, subroutines or functions (consist of a series of computational steps to be carried out).

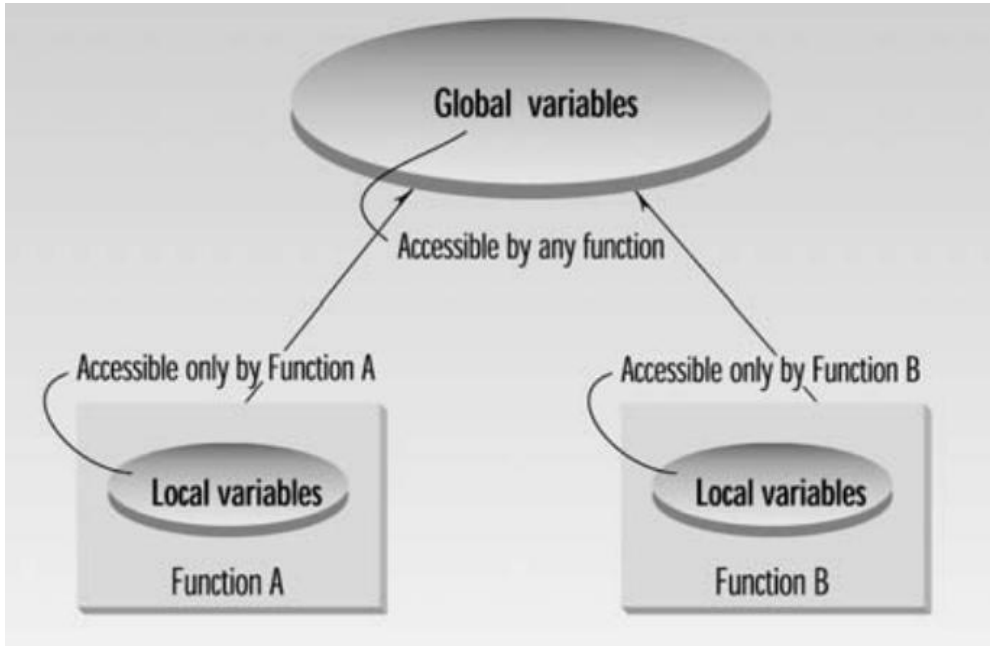
LANGUAGES: FORTRAN, ALGOL, COBOL, BASIC, Pascal and C.

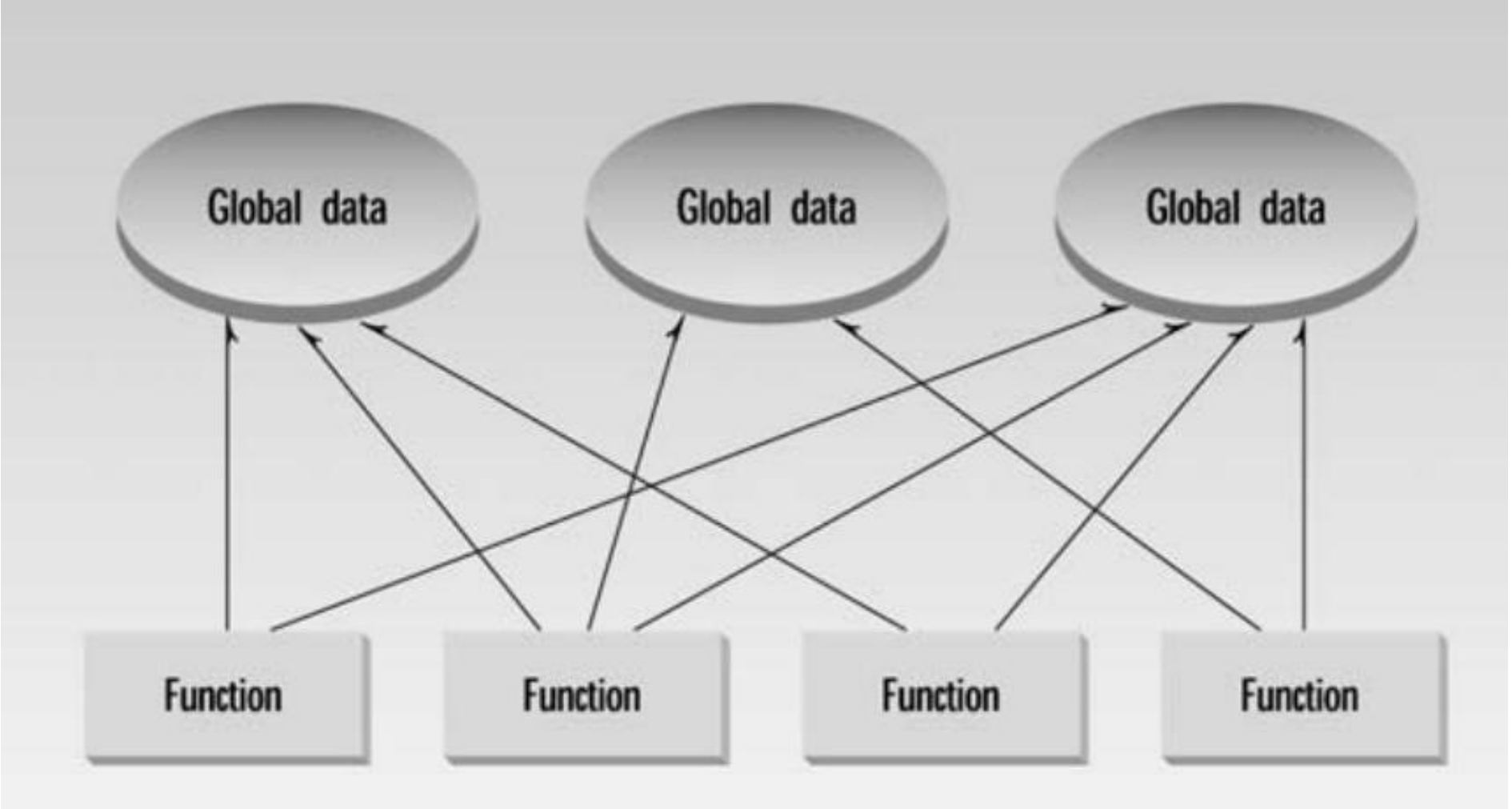
PROBLEMS IN PROCEDURAL PROGRAMMING

Project complexity, the schedule slips, more programmers are added, complexity increases, costs skyrocket …

WEAKNESSES: PROCEDURAL PARADIGM

1. Functions have unrestricted access to global data.
2. Unrelated functions and data, provide a poor model of the real world.




Global data
Global data
Global data
Function
Function
Function
Function

WEAKNESSES: PROCEDURAL PARADIGM

- A large number of connections makes the program more complex
- A program's structure is difficult to conceptualize
- It makes the program difficult to modify
- A change made in a global data item may necessitate rewriting all the functions that access that item
- Difficulty in modeling Real-World Problems

Complex real-world objects have both attributes and behavior.

PROGRAMMING PARADIGMS

Object Oriented Programming

- A **programming paradigm** that organizes software design around **objects** rather than **functions and logic**.
- An **object** represents a **real-world entity** (a *student, car, account, or sensor)* that has:
  - **Attributes (data)**
  - **Methods (behaviors)**

LANGUAGES: Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Objective-C, Dart, Swift, Scala.

ACTIVITY

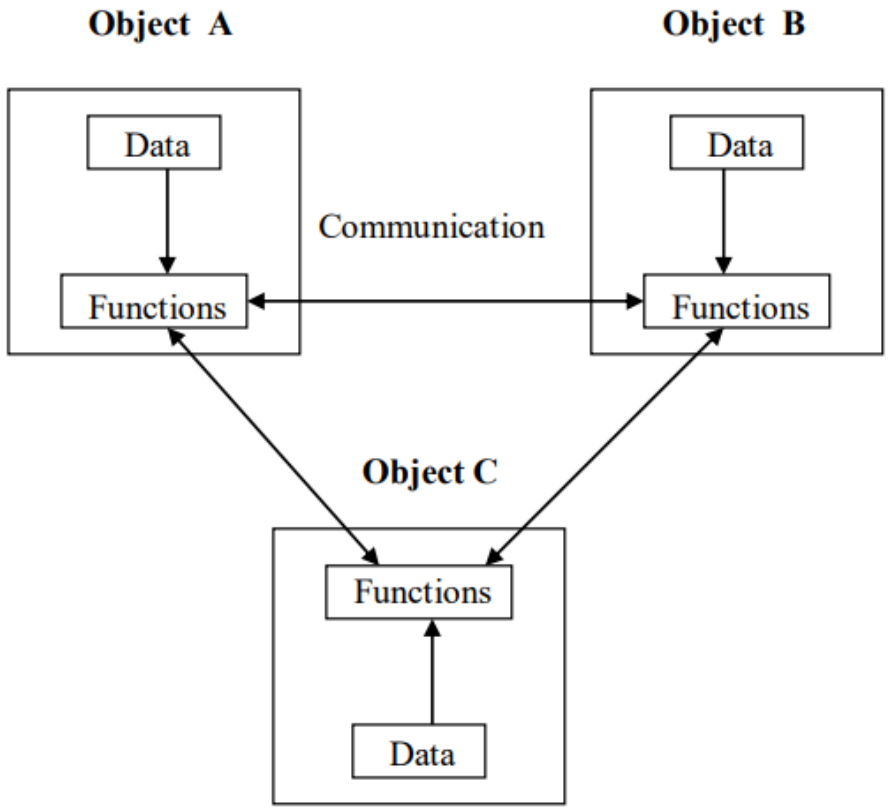
Procedural Paradigm

- **Suppose you were asked to design a simple Student Management System** that can:
  - Add student details (name, ID, marks).
  - Calculate the average mark.
  - Display a student's report.

OOP Paradigm

INTRODUCTION TO OOP

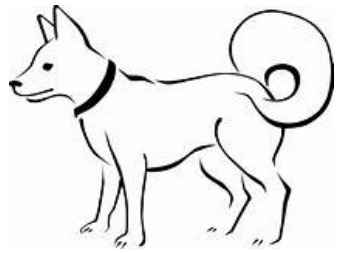
Using a modular, object-oriented design-and implementation approach can make software development groups much more productive than was possible with earlier techniques.



BASIC BUILDING BLOCKS

Object Oriented Programming

Concept	Description	Example
Class	A **blueprint** or **template** for creating objects. It defines attributes and methods.	class Student { … }
Object	An **instance** of a class — represents one specific entity.	val s1 = new Student()
Attributes (Fields)	Variables inside a class that hold data.	name, age, marks
Methods	Functions inside a class that define behaviors.	calculateAverage()



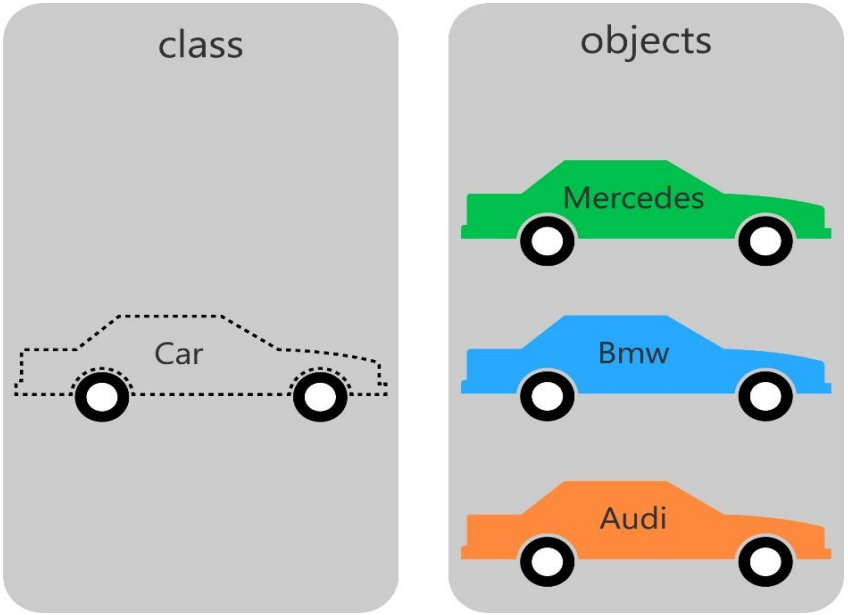
A CLASS

- A blueprint of an entity.
- A user-defined data type.
- Consists of data members and member functions.
- Accessed and used by creating an instance of the class.
- Represents the set of properties or methods that are common to all entities.

Example: Car

- All cars share some common properties (Brand, Color, Speed Limit, Mileage, etc)

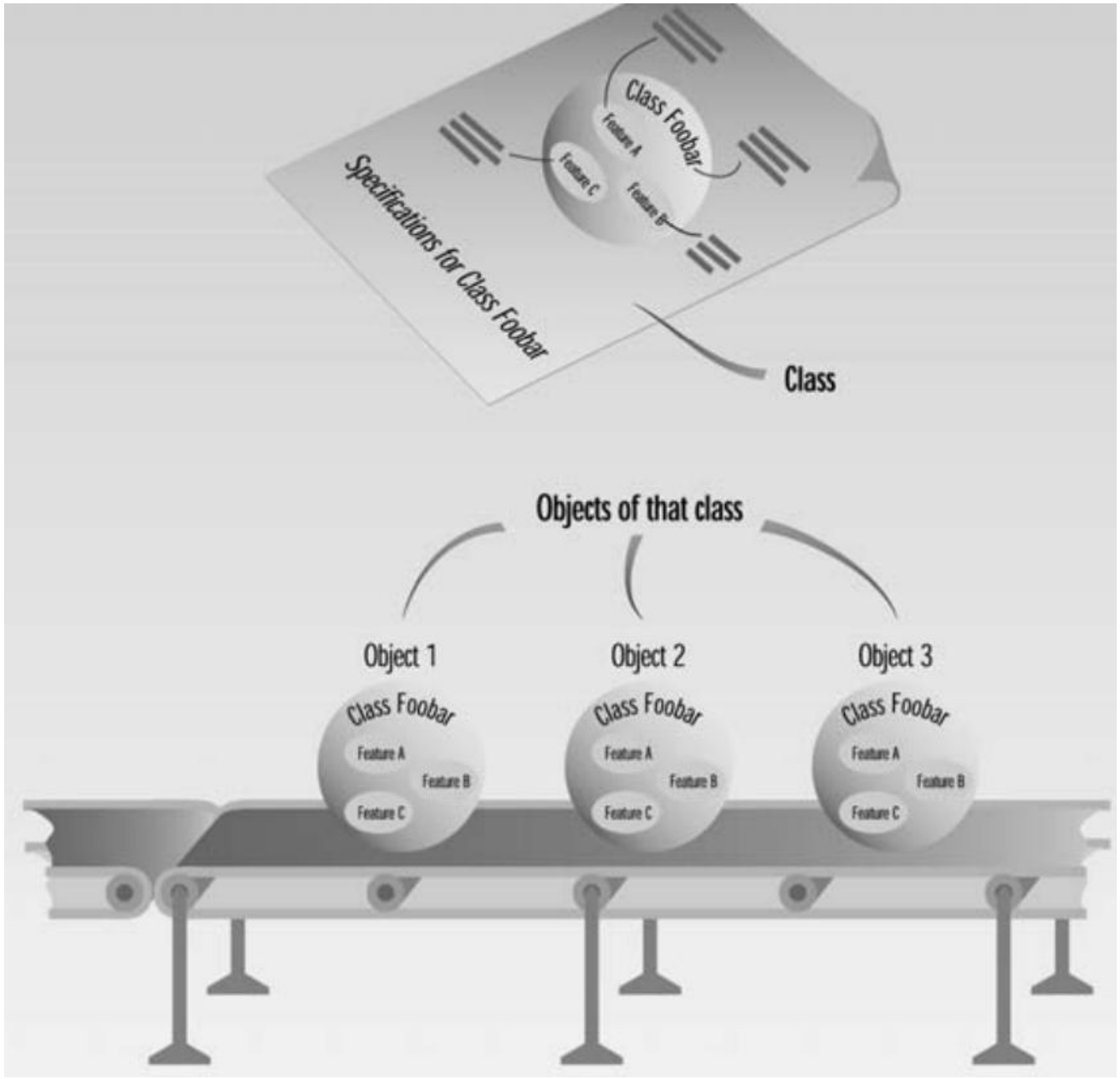
OBJECT



- Basic unit of Object-Oriented Programming
- Represents the real-life entities
- An instance of a Class.
- Has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Example: "My Car"

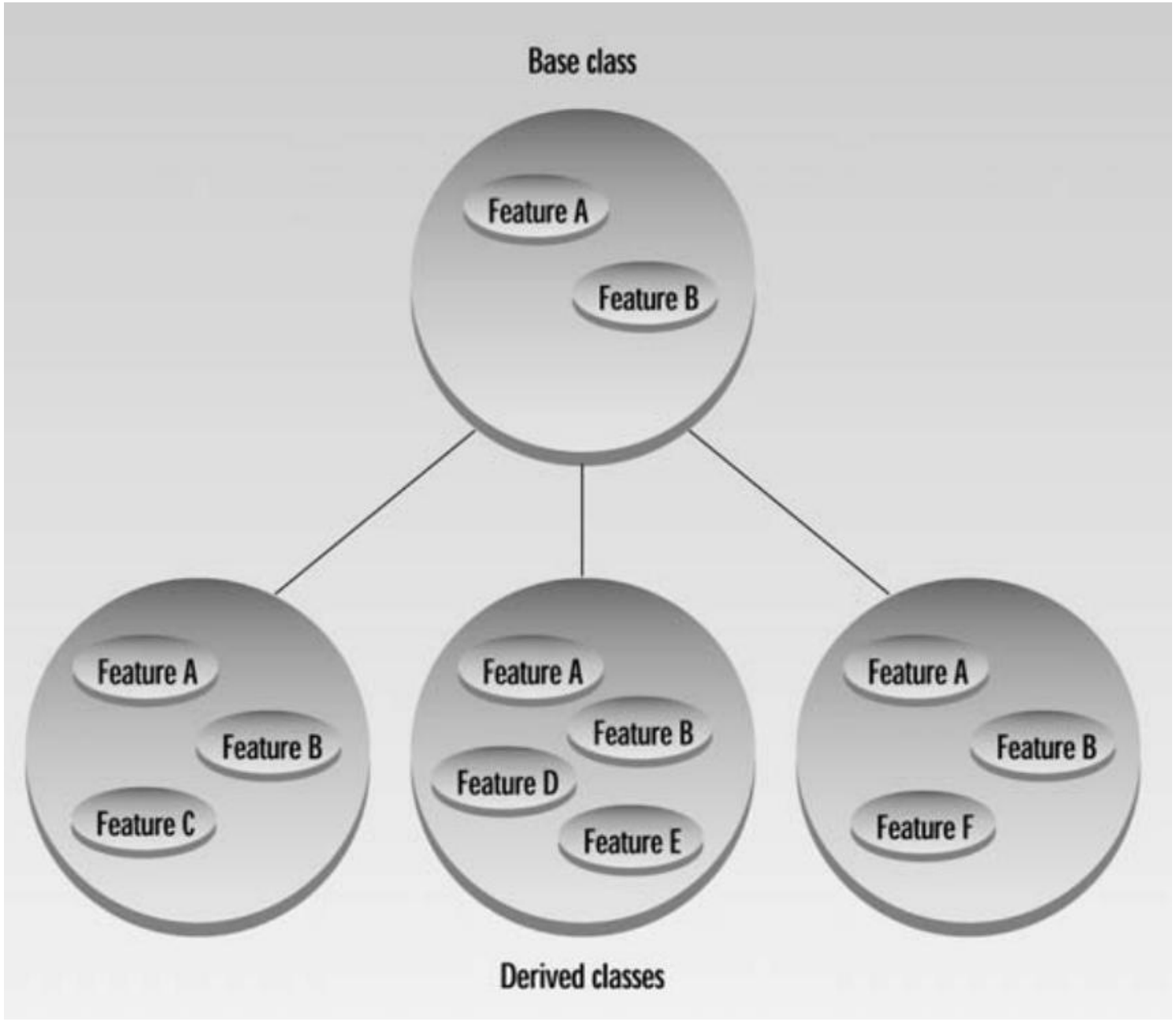
- Characteristics like color, Manufactured year, price, Drive, and Break.


Class Foobar
Feature A
Feature C
Feature B
Specifications for Class Foobar
Class
Objects of that class
Object 1
Class Foobar
Feature A
Feature B
Feature C
Object 2
Class Foobar
Feature A
Feature B
Feature C
Object 3
Class Foobar
Feature A
Feature B
Feature C

BASIC OBJECT ORIENTED CONCEPTS

INHERITANCE

- Classes divided into subclasses.
  - The Animals can be divided into mammals, amphibians, insects, birds, etc.
  - The vehicles can be divided into cars, trucks, buses, motorcycles, etc.
- A new class of object derives properties and characteristics from another class.
- It inherits properties from other classes.
- No need to write all the properties and functions again and again, as these can be inherited from another class that possesses it.
- Allows the user to reuse the code whenever possible and reduce its redundancy.


Base class
Feature A
Feature B
Feature A
Feature B
Feature C
Feature A
Feature B
Feature D
Feature E
Feature A
Feature B
Feature F
Derived classes

DATA ABSTRACTION

- Provide only essential information about the data to the outside world.
- Hide the background details or implementation.

Example: A man driving the car

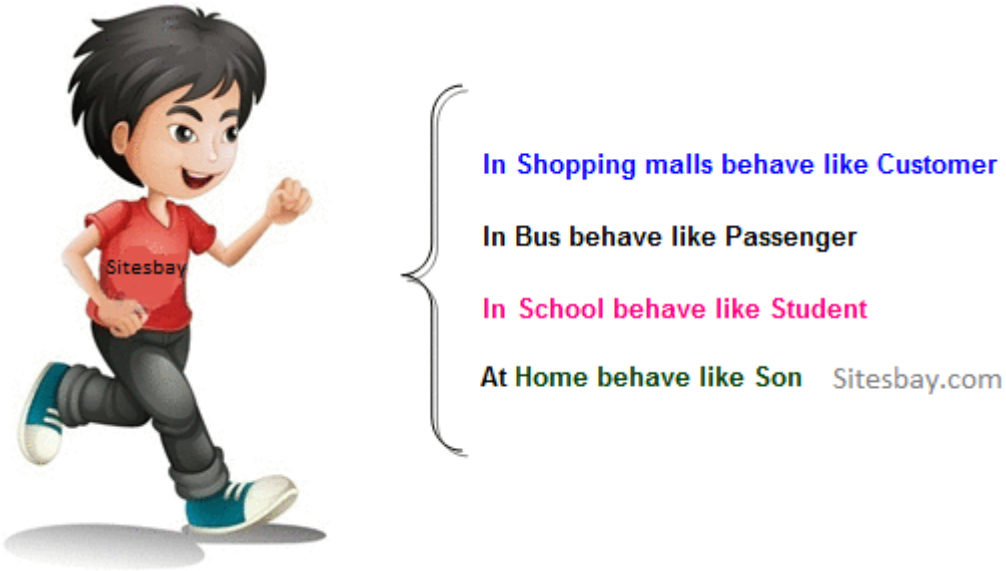
- Knows by pressing the accelerators will increase the speed/applying brakes will stop the car BUT NOT how.

ENCAPSULATION

- Wrapping up of data under a single unit.
- An object's attributes and member functions are intimately related.
- Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented.
- Implementation details are *hidden* within the objects themselves.
- This is also known as **information hiding.**

POLYMORPHISM

- Polymorphism means having many forms.
- Ability of a message to be displayed in more than one form.



ACTIVITY: COMPARE & CONTRAST

Procedural vs Object-Oriented Programming

PROCEDURAL ORIENTED PROGRAMMING	OBJECT ORIENTED PROGRAMMING
In procedural programming, program is divided into small parts called ***functions***.	In object-oriented programming, program is divided into small parts called ***objects***.
Procedural programming follows ***top down approach***.	Object oriented programming follows ***bottom up approach***.
There is no access specifier in procedural programming.	Object oriented programming have access specifiers like private, public, protected etc.
Adding new data and function is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way for hiding data so it is ***less secure***.	Object oriented programming provides data hiding so it is ***more secure***.
In procedural programming, overloading is not possible.	Overloading is possible in object-oriented programming.
In procedural programming, function is more important than data.	In object-oriented programming, data is more important than function.
Procedural programming is based on ***unreal world***.	Object oriented programming is based on ***real world***.
Example: C, FORTRAN, Pascal, Basic etc.	Example: C++, Java, Python, C# etc.

Ref: https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/

ADVANTAGES OF OOP

Modularity	Code is organized into independent classes.
Reusability	Classes and methods can be reused in other programs.
Maintainability	Easier to fix and update.
Security	Encapsulation hides data from outside interference.
Extensibility	Inheritance and polymorphism allow easy expansion.

OOP USING C++

General-purpose programming language developed as an enhancement of the C language to include object-oriented paradigm.

FEATURES OF C++

Object-Oriented Programming

- C++ is an Object-Oriented Programming Language.
- C is a procedural programming language.
- It can create/destroy objects while programming.

Machine Independent/Platform Dependent

- C++ executable is not platform-independent
  - Compiled programs on Linux will not run on Windows
- However they are machine independent.
  - Source code can be ported from one machine to another.

FEATURES OF C++

SIMPLE Language

- Programs can be broken down into logical units and parts.
- Has rich library support, and a variety of data-types.

Compiler Based

- Compiled and runs the executable file.

Case-sensitive

- **cin/Cin**

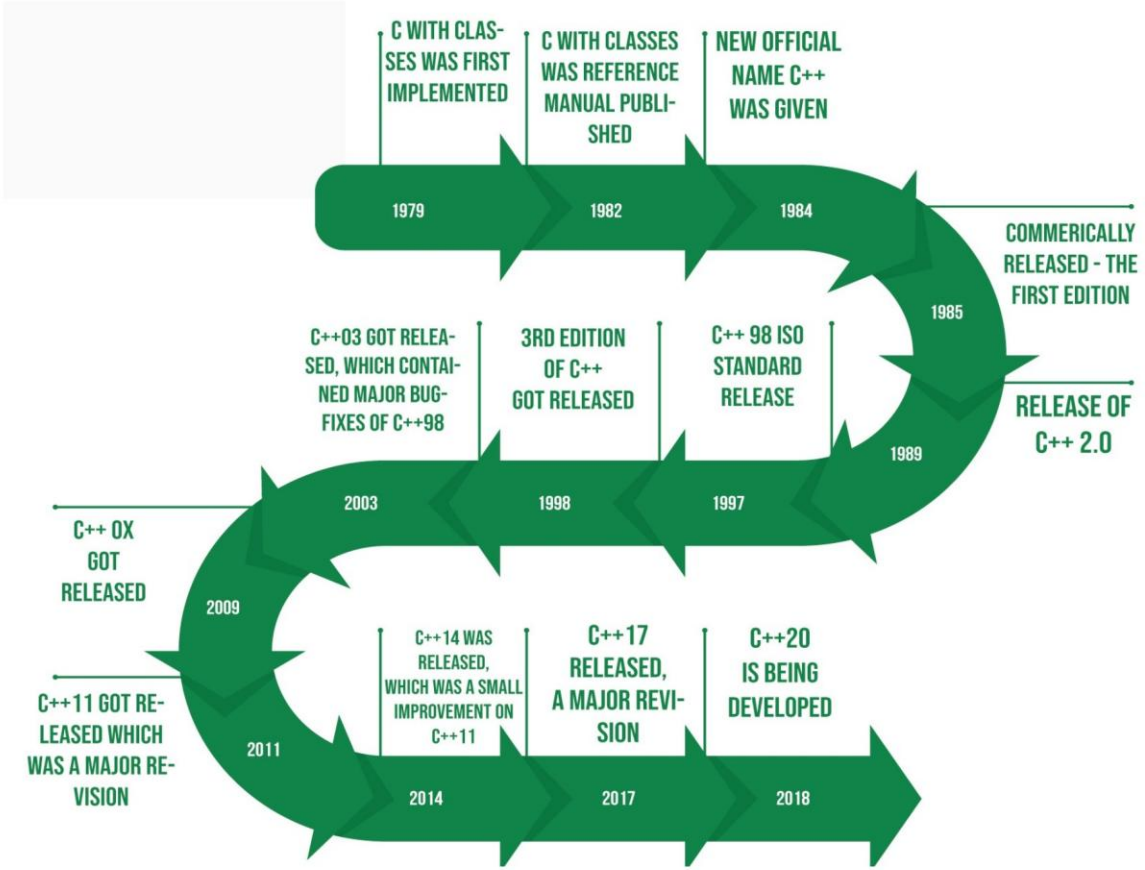
Dynamic Memory Allocation

- Allows to allocate the memory of a variable or an array in run time.

A BRIEF HISTORY

C++

History of C++



Ref: https://www.geeksforgeeks.org/history-of-c/

A BRIEF HISTORY TO C++

- Developed by Bjarne Stroustrup, as an extension to the C language (**1979).**
- Danish Computer Scientist at Bell Telephone Laboratories (Nokia Bell Labs) in New Jersey
- Simula (Used for simulations) : primary language to support the object-oriented programming- too slow for practice & practical use.
- Worked on "**C with Classes" to get** advanced object-oriented programming, into the C language
- In 1983, the name Change from C with categories to C++.
  - The ++ operator for incrementing a variable
- The language was updated 3 major times in 2011, 2014, and 2017 to C++11, C++14, and C++17.

C++ BASICS

Getting Familiar

GETTING STARTED: C++

- An editor to write C++ code
- A compiler to translate the C++ code into a language the computer understands

IDEs (Integrated Development Environment) to edit AND compile

- Eclipse
- Visual Studio
- **Dev C++**

EXERCISE 1

Print "Hello World"

```
#include <iostream>
int main()
{
        std::cout<<"Hello World";
        return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
        cout<<"Hello World";
        return 0;
}
```

EXERCISE 2

Print in multiple lines

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World\n";
    cout<<"This is my first C++ code\n";
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World"<<endl;
    cout<<"This is my first C++code"<<endl;
    return 0;
}
```

EXERCISE 3

```
#include <iostream>
using namespace std;
int main(){
int varNum = 456;
double varFloatNum = 65.459;
 char varChara = 'X';
 string varText = "Hello world";
 bool varBoolean = true;
   cout<<varNum<<endl;
   cout<<varFloatNum<<endl;
   cout<<varChara<<endl;
   cout<<varText<<endl;
   cout<<varBoolean<<endl;
 }
```

Declaring Variables

Syntax:

type variable = value;

EXERCISE 4

User Inputs

```
#include <iostream>
using namespace std;
int main()
{
    int x;
    int y;
       cout<<"Enter X = ";
       cin>>x;
       cout<<"Enter Y = ";
       cin>>y;
             cout<<"SUM of X and Y is "<<x+y;
}
```