# PBCST304:
# OBJECT ORIENTED PROGRAMMING

# Course Outcomes

| Course Outcome | | Bloom's Knowledge Level (KL) |
|---|---|---|
| **CO1** | Explain the process of writing, compiling, and executing basic Java programs, including their structure and components, to demonstrate proficiency. | **K2** |
| **CO2** | Utilize object-oriented programming principles in the design and implementation of Java applications. | **K3** |
| **CO3** | Develop and manage Java packages and interfaces, enhancing code modularity and reusability. | **K3** |
| **CO4** | Implement error handling using Java's exception mechanisms and leverage interfaces for modular applications. | **K3** |
| **CO5** | Develop event-driven Java GUI applications with database connectivity using Swing and JDBC. | **K3** |

Note: *K1- Remember, K2- Understand, K3- Apply, K4- Analyse, K5- Evaluate, K6- Create*

# Text Books

| Text Books | | | | |
|---|---|---|---|---|
| **Sl. No** | **Title of the Book** | **Name of the Author/s** | **Name of the Publisher** | **Edition and Year** |
| 1 | Java: The Complete Reference | Herbert Schildt | Tata McGraw Hill | 13/e, 2024 |
| 2 | Introduction to Java Programming, Comprehensive Version | Y Daniel Liang | Pearson | 10/e, 2014 |
| 3 | Head First Design Patterns | Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra | O'Reilly Media | 1/e, 2004 |

# Learning Outcomes

- Understand the concepts of Function-Oriented Design
- Understand the concepts of Object-Oriented Design
- Differentiate the differences of FOD and OOD

# Approaches To Software Design

- Function-Oriented Design
- Object-Oriented Design

# Function-Oriented Design

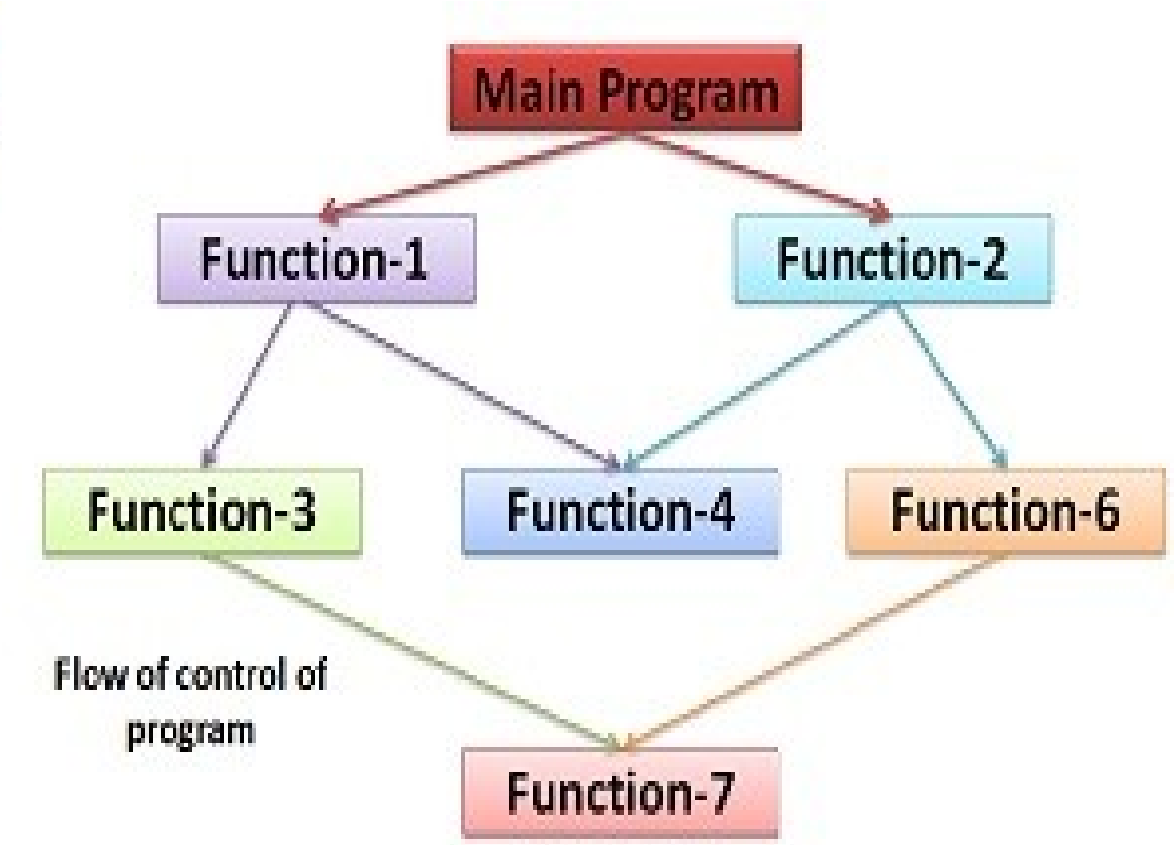
- In the function/procedure-oriented approach, the problem is viewed as **a sequence of jobs**, to be done such as **reading**, **calculating** and **printing**.
- A set of functions can be written to accomplish these tasks.
- The primary focus is on **functions**.
- Examples of function-oriented programming languages are C, FORTRAN, Pascal.
- Sometimes known as **Structured** or **Modular** programming

# Function-Oriented Design

- Characteristics of Function-Oriented Design:
  - It focuses on **functions** rather than data.
  - A program is **divided** into a **number of functions** and each function has clearly defined purpose.
  - Most of the functions share **global** data.
  - **Data** can **move** through the program **freely** from function to function.

# Function-Oriented Design

**Top-down Decomposition:**




Structure of procedure oriented program

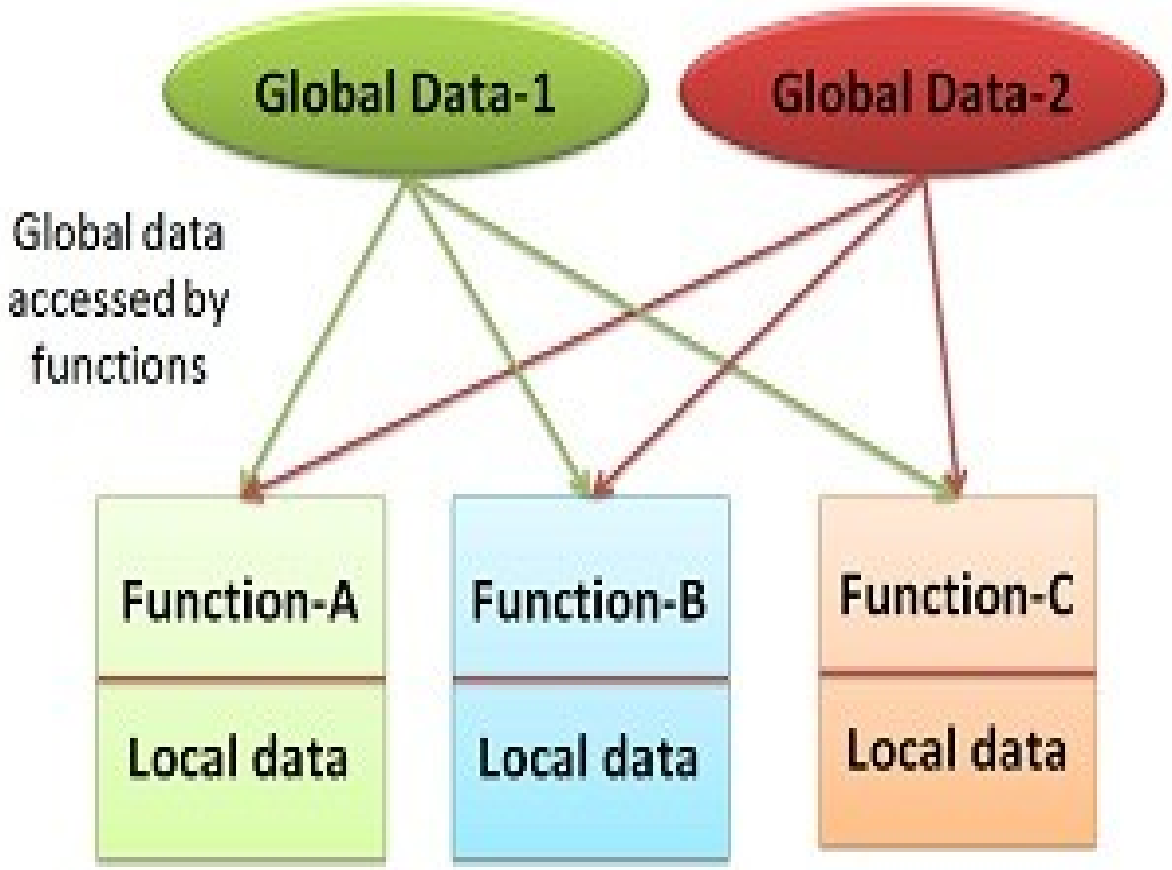
# Function-Oriented Design

**Centralised system state:**

- System state - the values of certain data items that determines the response of the system to a user action/external event.
- Such ***data usually have global scope and are shared by many functions.***
- The system state is centralised and shared among different functions.

# Function-Oriented Design

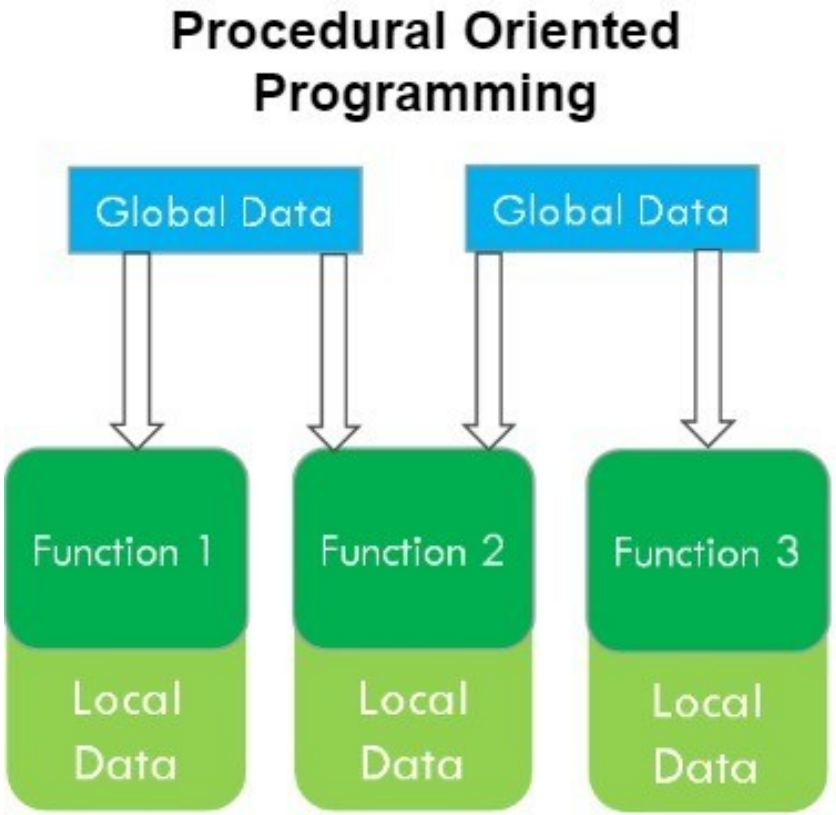
**Centralised system state:**

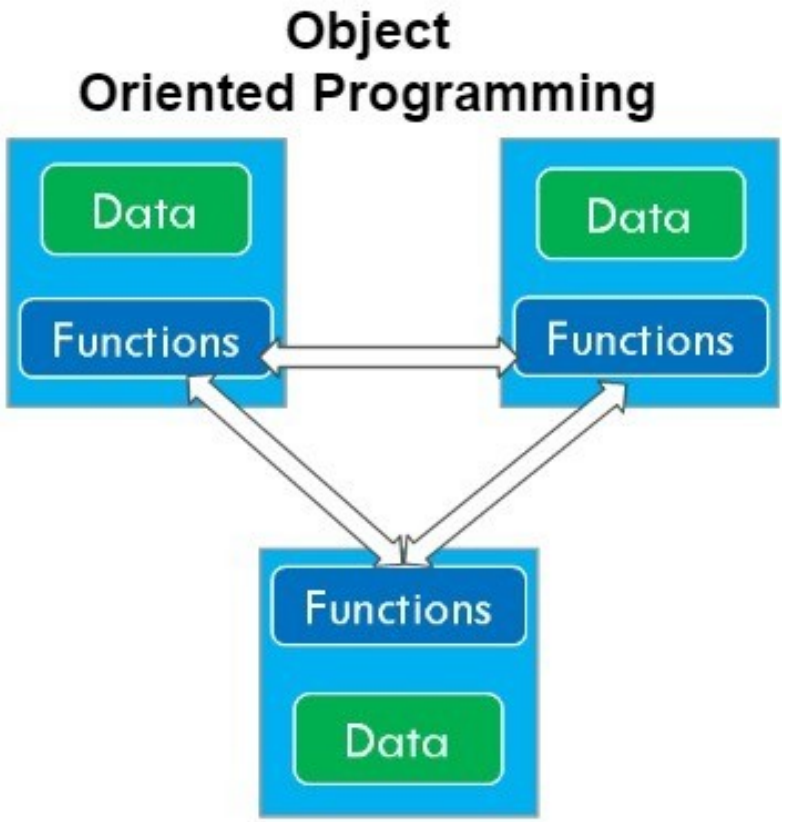


# Object-Oriented Design

- The core of the pure object oriented programming is to create an ***object***, that has certain properties and methods.
- The main idea behind the object oriented approach is to **combine *process*** (function) and ***data*** into a unit called an ***object***.
- Hence, it focuses on object rather than procedure/function.
- Examples: Java, C++, Smalltalk and Python.

# Object-Oriented Design



# Object-Oriented Design

- Characteristics of Object-oriented Design:
  - Emphasis on **data** rather than function.
  - Programs are divided into entities known as ***objects***.
  - Functions that operate on data of an object are **tied** together in data structure.
  - Objects may **communicate** with each other through functions.

# Object-Oriented Design

- A system is viewed as being made up of a **collection of objects** (i.e. entities).
- Each object is associated with a set of functions that are called its **methods**.
- Each object contains **its own data** and is responsible for managing it.
- The data internal to an object **cannot** be accessed directly by other objects and only through invocation of the methods of the object.

# Object-Oriented Design

- The system state is **decentralised** since there is **no globally shared data** in the system and data is stored in each object.
- The methods defined for one object cannot directly refer to or change the data of other objects.
- Objects can also be considered as instances of ***abstract data types*** (ADTs).
- Three important concepts associated with an ADT:
  - *Data Abstraction*, *Data Structure*, *Data Type*

# Object-Oriented Design

**Data Abstraction:**

- How data is exactly stored, is abstracted away.
- This means that *any entity external to the object would have no knowledge about how data is exactly stored, organised, and manipulated inside the object.*
- The entities external to the object can access the data internal to an object only by calling *certain well-defined methods supported by the object.*

# Object-Oriented Design

**Data Abstraction:**

- Example - Consider an ADT such as a stack.
- The data of a stack object may internally be stored in an array, a linearly linked list, or a bidirectional linked list.
- The external entities have no knowledge of this.
- They can access data of a stack object only through the supported operations such as push and pop.

# Object-Oriented Design

**Data Structure:**

- A data structure is constructed from ***a collection of primitive data items.***
- A civil engineer builds a large civil engineering structure using primitive building materials such as bricks, iron rods, and cement.
- A programmer can construct a data structure as an organised collection of primitive data items such as integer, floating point numbers, characters, etc.

# Object-Oriented Design

**Data Types:**

- A type is a programming language terminology that refers to anything that can be instantiated.
- Example - int, float, char etc., are the basic data types supported by C programming language.
- ADTs are user defined data types. Eg: classes in OOP

# Object-Oriented Vs. Fuction-Oriented

| Function-Oriented Design | Object-Oriented Design |
| --- | --- |
| 1. The program is divided into small modules called functions. | 1. The program is divided into number of parts called objects. |
| 2. Importance is given to functions rather than data. | 2. Importance is given to the data rather than functions. |
| 3. Does not have any access specifier. | 3. It has access specifiers named as public, private and protected. |
| 4. Most of the functions share global data. | 4. Data is hidden and cannot be accessed by external functions. |

# Object-Oriented Vs. Fuction-Oriented

| Function-Oriented Design | Object-Oriented Design |
|---|---|
| 5. It follows top-down approach. | 5. It follows bottom-up approach. |
| 6. No proper way for hiding data; hence it is less secure. | 6. Data hiding provides more security |
| 7. Overloading is not possible. | 7. Function and operator overloadings are possible |
| 8. State information is available in a centralised shared data store | 8. State information exists in the form of data distributed among several objects of the system |
| 9. Eg: FORTRAN, Pascal, C | 9. Eg:  C++, Java, Python |

# Case Study: Automated Fire Alarm System

- **Smoke detectors**, **sprinkler** and **fire alarms** would be placed in each room of the building.
- The fire alarm system would monitor the status of these smoke detectors.
- Whenever a fire condition is reported by any of the smoke detectors, the fire alarm system should determine the location at which the fire has been sensed and then sound the alarms only in the neighbouring locations.
- The fire alarm system should also flash an alarm message on the computer console.
- After a fire condition has been successfully handled, the fire alarm system should support resetting the alarms by the fire fighting personnel.

# Case Study: Automated Fire Alarm System

**Function-Oriented Approach:**

- ***Different high-level functions are first identified, and then the data structures are designed.***
- Functions:
  - *interrogate_detectors(), get_detector_location(), determine_neighbour_alarm(), determine_neighbour_sprinkler(), ring_alarm(), activate_sprinkler(), reset_alarm(), reset_sprinkler(), report_fire_location()*

# Case Study: Automated Fire Alarm System

## Function-Oriented Approach:

- Data Structures:

```
/* Global data (system state) accessible by various functions */
        BOOL  detector_status[MAX_ROOMS];
        int   detector_locs[MAX_ROOMS];
        BOOL  alarm-status[MAX_ROOMS]; /* alarm activated when status is set */
        int   alarm_locs[MAX_ROOMS]; /* room number where alarm is located */
        int   neighbour-alarms[MAX-ROOMS][10]; /* each detector has at most */
                                               /* 10 neighbouring alarm locations */
        int   sprinkler[MAX_ROOMS];
```

# Case Study: Automated Fire Alarm System

**Object-Oriented Approach:**

- ***Different classes of objects are identified.***
- Subsequently, ***the methods and data for each object are identified.***
- Finally, ***an appropriate number of instances of each class is created.***
- *class* ***detector***

  *attributes: status, location, neighbours*

  *operations: create, sense-status, get-location, find-neighbours*

# Case Study: Automated Fire Alarm System

**Object-Oriented Approach:**

- *class* ***alarm***

  *attributes: location, status*

  *operations: create, ring-alarm, get_location, reset-alarm*

- *class* ***sprinkler***

  *attributes: location, status*

  *operations: create, activate-sprinkler, get_location, reset-sprinkler*

# Case Study: Automated Fire Alarm System

**Differences b/w FOD and OOD:**

- In a function-oriented program,
  - the system state (data) is **centralised** and several functions access and modify this central data.
- In case of an object-oriented program,
  - the state information (data) is **distributed** among various objects.

# Case Study: Automated Fire Alarm System

**Differences b/w FOD and OOD:**

- In the object-oriented design,
  - data is **private** in different objects and these are **not available to the other objects** for direct access and modification.
- The basic unit of designing an object-oriented program is objects, whereas it is functions and modules in procedural designing.
- Objects appear as ***nouns*** in the problem description; whereas functions appear as ***verbs***.