

# Programming Logic & OOP's Concept



# Basic understanding of any computer program

- Any activity whether in the real world or software world, can be broken into separate phases:
  - The first phase is called the input phase
  - The second phase is called the process phase
  - The last phase is called the output phase
- The cycle of activities performed by a computer is referred to as the Input-Process-Output cycle or the I-P-O cycle
- A computer consists of several components
- Each component participates in either one of the input, process, or output phases




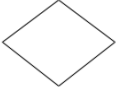

## Cont'd...

- A computer is designed to accept input, process it, and generate output
- A set of instructions to perform a task is called a program
- A number of programs together form an application




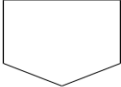

# Role of Flowchart

- A flowchart is a graphical representation of the steps to be followed for solving a problem.
- It consists of a set of symbols.
- Each symbol represents a specific activity.

# Flowchart Symbols

| Symbol  | Activity   |
|---|------------|
|  | Input      |
|  | Processing |
|  | Output     |
|  | Decision   |
|  | Subroutine |

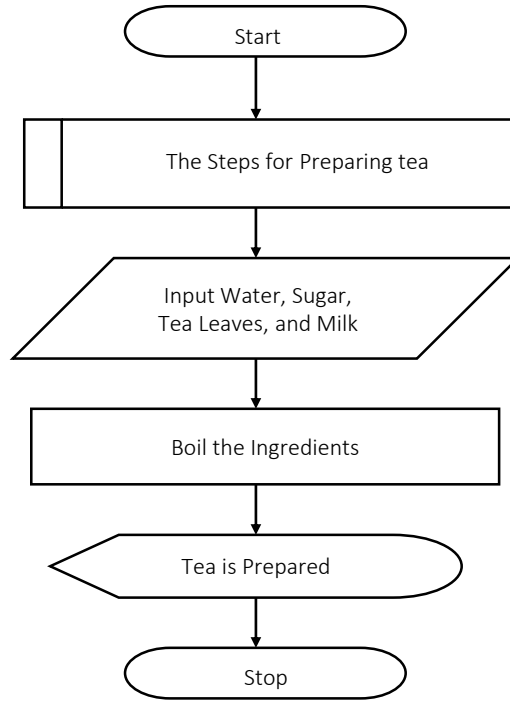
## Cont'd...

| Symbol  | Activity           |
|---|--------------------|
|  | Flow lines         |
|  | Terminator         |
|  | On page connector  |
|  | Off page connector |
|  | Annotation         |

# Example

◆ Example:

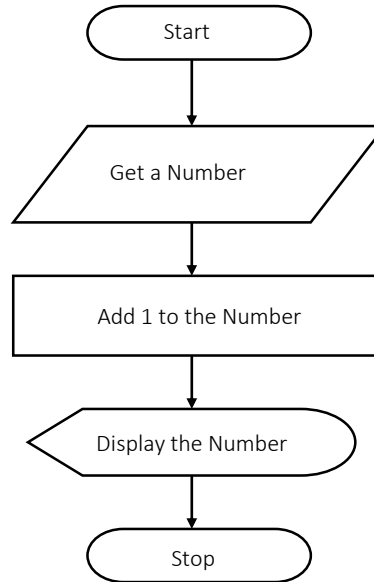
◆ Flowchart for preparing tea



## Cont'd...

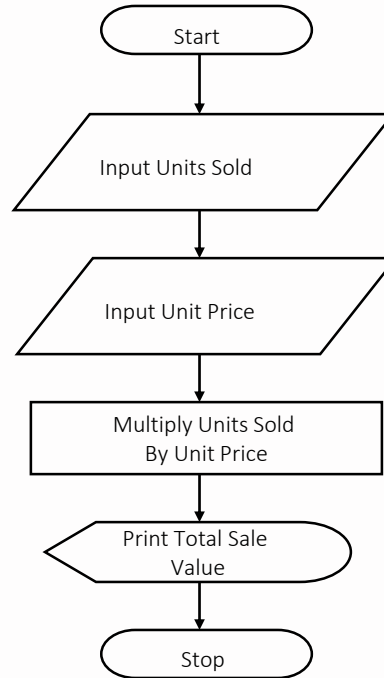
### ◆ Example:

- ◆ Flowchart for manipulating numbers





## Cont'd...



# Quiz Time

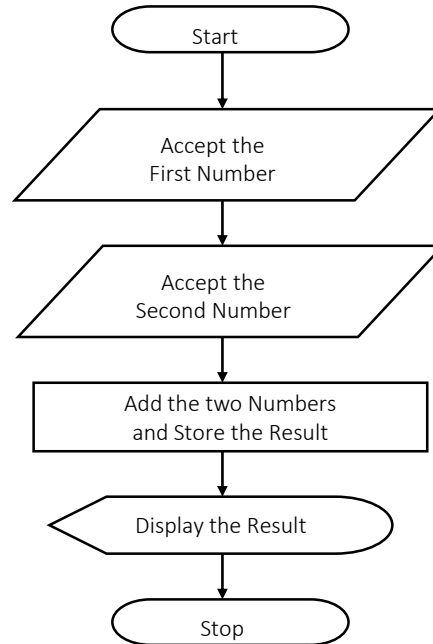
- Rearrange the following steps in the order of correct sequence to add two numbers and draw a flowchart for the same:
  - Get the First Number
  - Display the Result
  - Stop
  - Add the Two Numbers
  - Get the Second Number
  - Start

## Cont'd...

- Draw a flowchart to accept five numbers and display the sum of the numbers.
- Draw a flowchart to input any number, multiply it by 2, and display the result.

# Example

- Example:
  - Flowchart to display the sum of two numbers

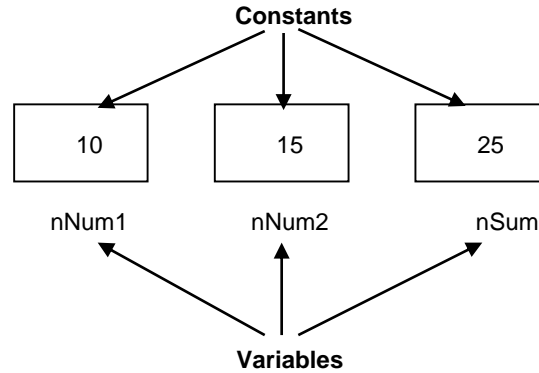


# Variables

- The internal memory consists of different locations in which data is stored.
- A computer needs to identify the memory locations to be able to retrieve values from or store values in them.
- The value of a variable changes each time the set of instructions is executed.

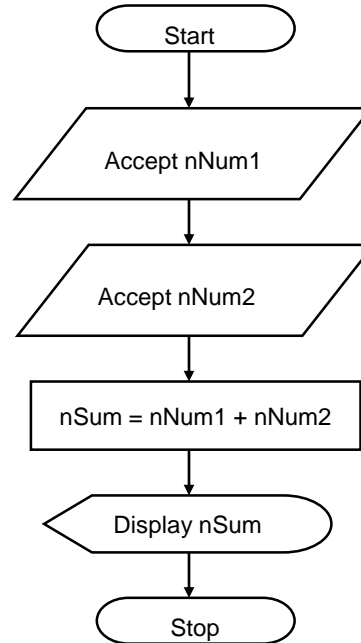
# Constants

- The values stored in the variables are known as constants.



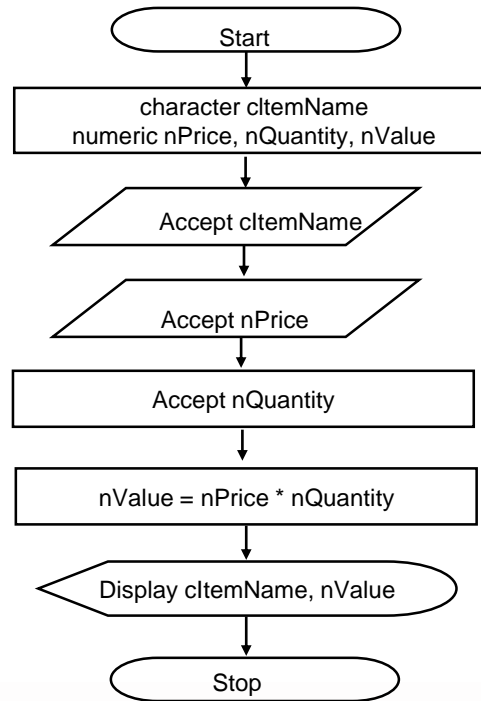
# Example

- Example:
  - Flowchart to display the sum of two numbers using variables



# Example

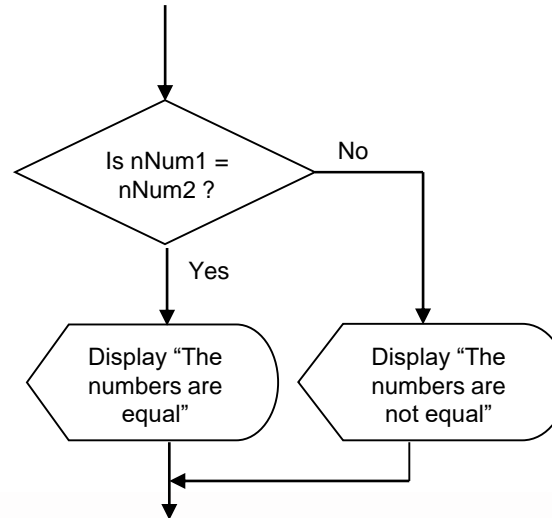
- Draw a flowchart to accept item name, price, and quantity. You need to calculate the value as the product of price and quantity, and display the calculated value and the item name using variables.





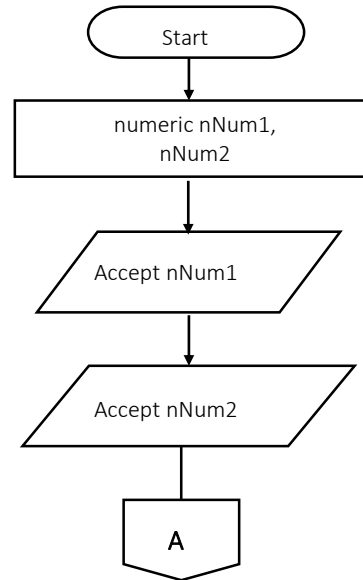
# Decision making

- Many problems require decisions to be made.
- All decisions may or may not state an action to be taken if the condition is false.
- Following is a flowchart segment to compare two numbers and check for equality.

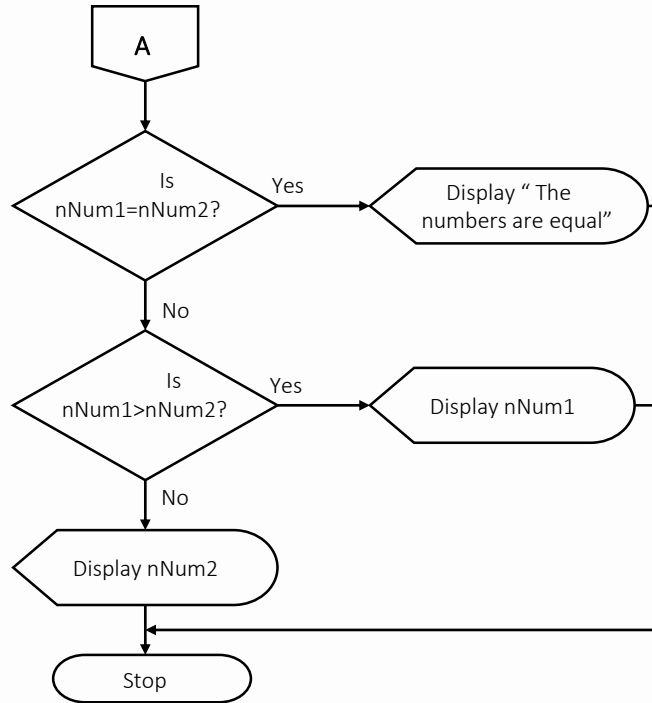


# Example

- Example:
  - Accept two numbers and print the larger of the two numbers

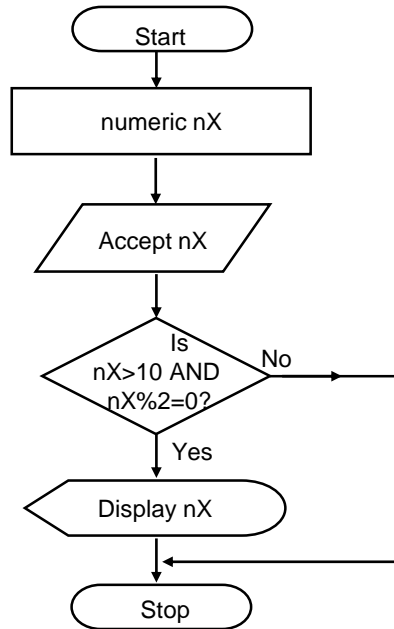


## Cont'd...



# Example

- Example:
  - Print the value of nX only if the value of nX is greater than 10 and nX is an even number

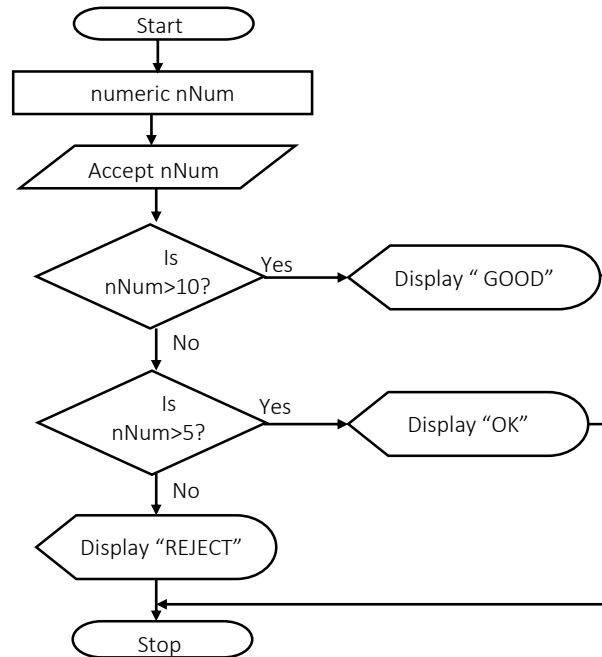


# Quiz Time

- Study the given flowchart and answer the following questions.

What will be the output when:

- nNum=7
- nNum=3
- nNum=11



## Cont'd...

- Draw a flowchart to accept a number and then find out whether or not the number is divisible by 5.

## Cont'd...

- Draw a flowchart to accept three numbers and display the largest number.

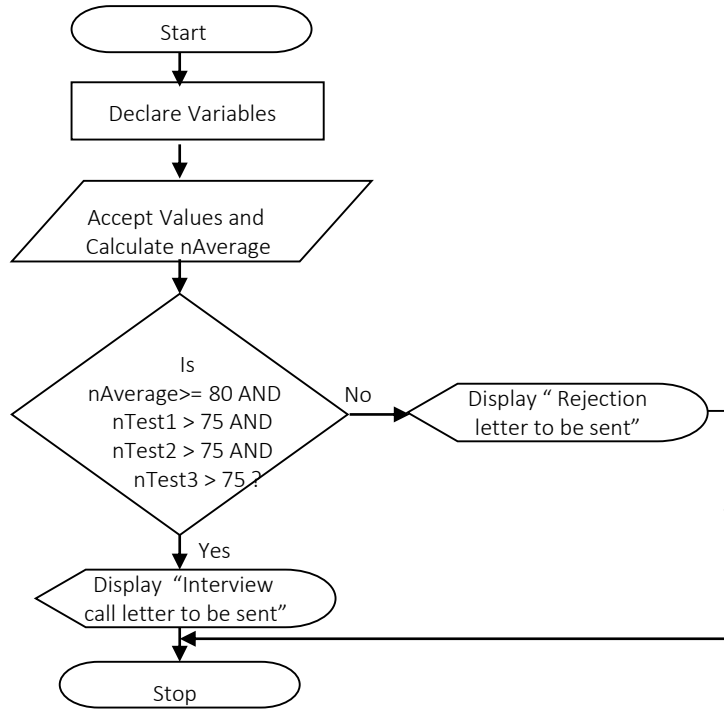
# Importance of Dry Run

- Helps you do a logic check
- Understand the flow of control in a flowchart
- Evaluate the output of the program with a set of sample values
- Provides a step by step evaluation of values in the variables of the program



# Example

- Flowchart to select a candidate



## Cont'd...

- Dry Run Table:

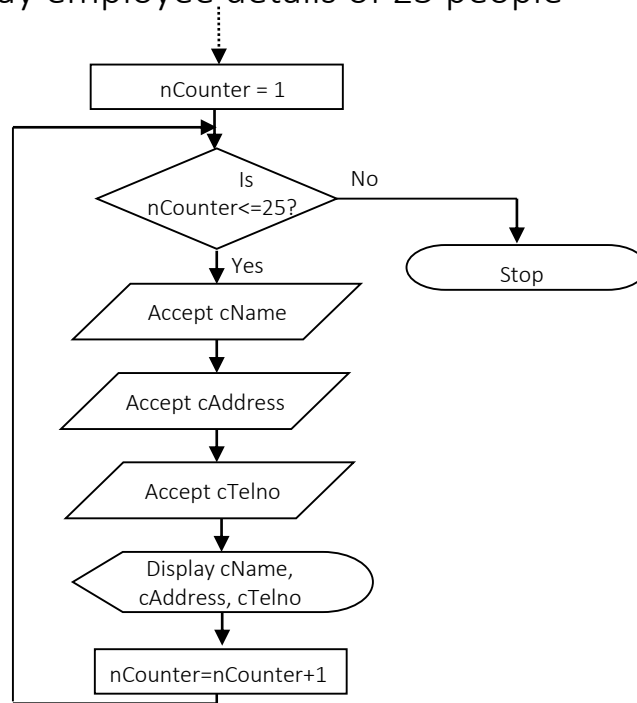
| S.No. | nTest1 | nTest2 | nTest 3 | nAverage | Output                           |
|-------|--------|--------|---------|----------|----------------------------------|
| 1.    | 95     | 90     | 88      | 91       | Interview call letter to be sent |
| 2.    | 80     | 77     | 83      | 80       | Interview call letter to be sent |
| 3.    | 90     | 92     | 74      | 85.33    | Rejection letter to be sent      |
| 4.    | 76     | 76     | 76      | 76       | Rejection letter to be sent      |

# Loops and Iterations

- An important characteristic of a computer is its ability to execute a series of instructions repeatedly.
- A loop is a sequence of instructions that will be repeated more than once.
- A loop performs steps in a specified sequence.
- There are two types of loops:
  - fixed loops where the number of repetitions is known
  - variable loops where the number of repetitions is not known

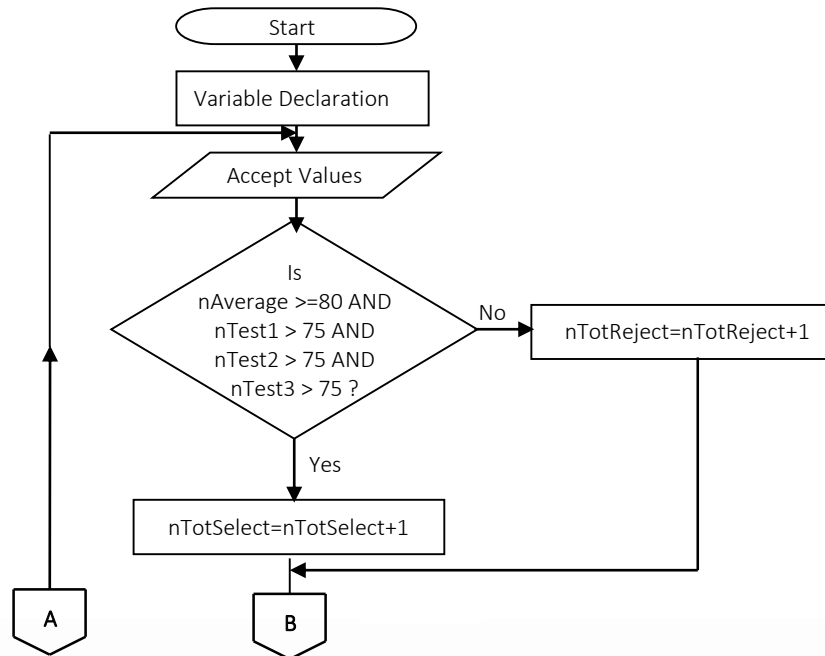
# Example

- Flowchart segment to display employee details of 25 people

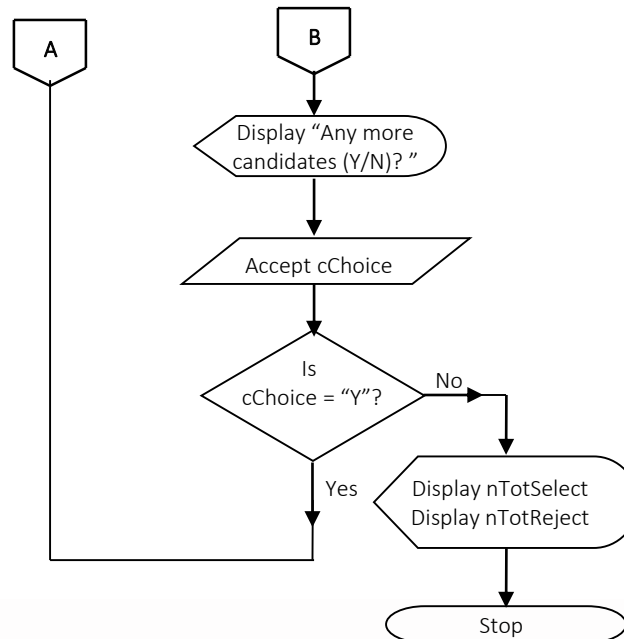


## Cont'd...

- Flowchart to calculate the total number of call letters and rejection letters sent

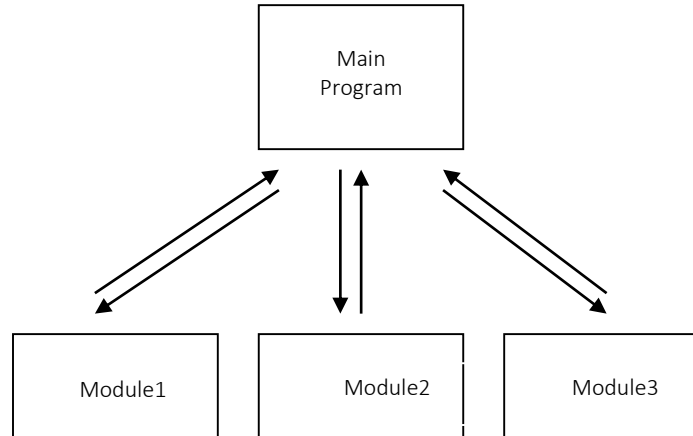


- Flowchart to calculate the total number of call letters and rejection letters sent (Contd.)



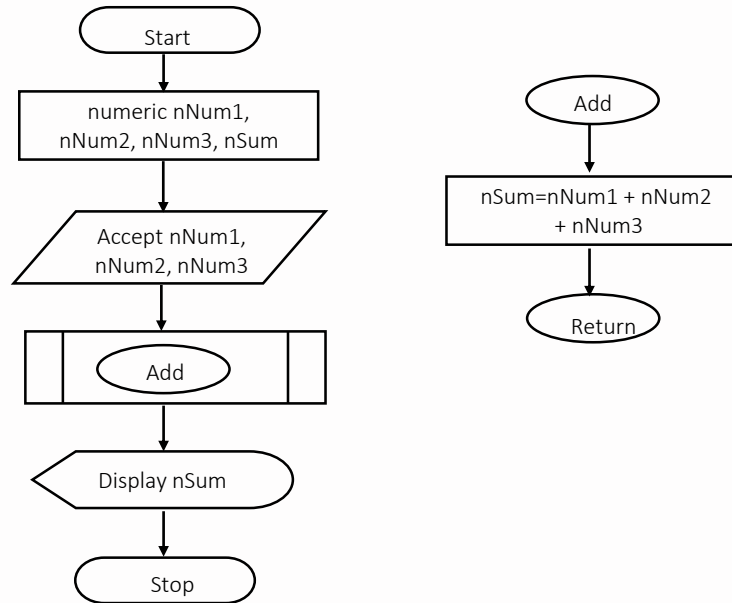
# Modular Programming

- Long, continuous programs can be broken up into a series of individual modules that are related to each other in a specified manner.



# Example

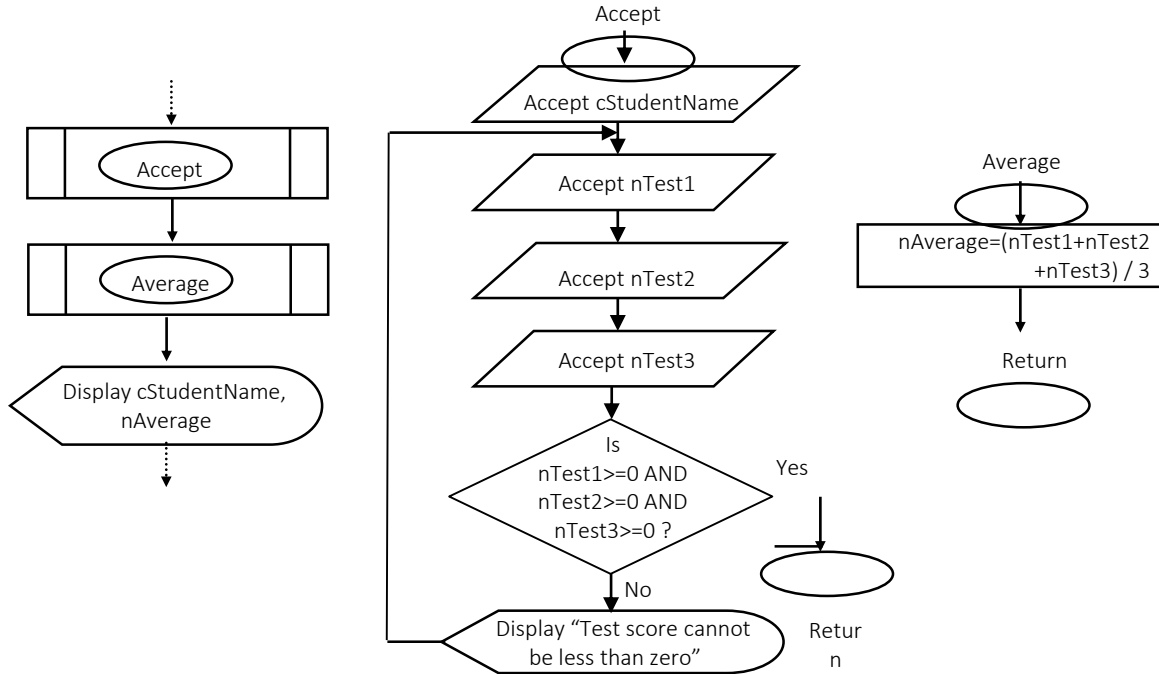
- Example:
  - Flowchart to show modular programming





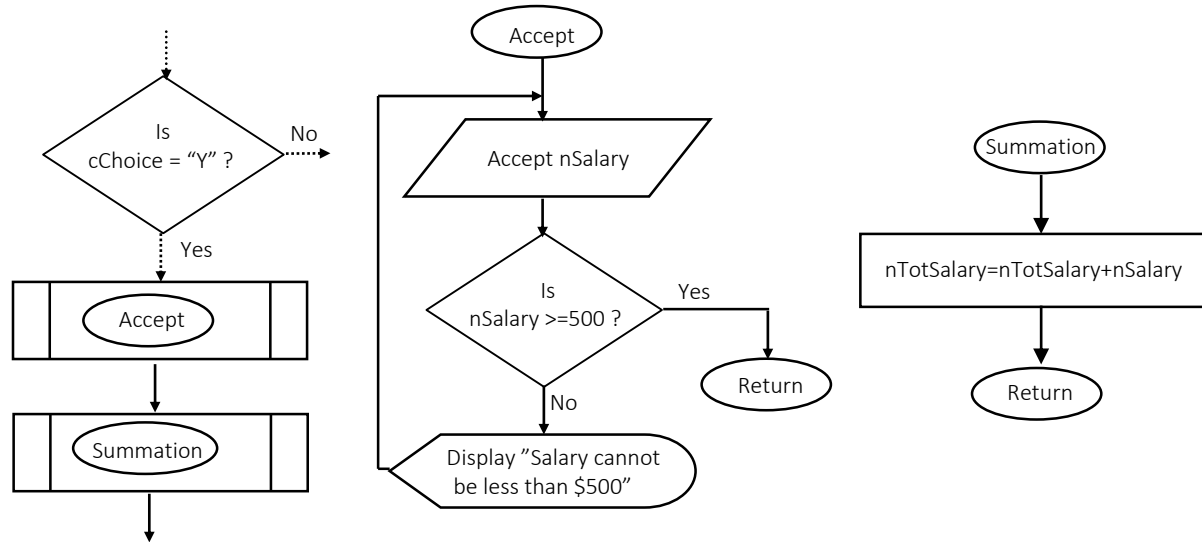
## Cont'd...

- Flowchart to calculate average marks of 10 students



# Example

- Flowchart to calculate total monthly expenditure on salaries



# Dry Run table

| S. No. | nSalary | nTotSalary | Output                           |
|--------|---------|------------|----------------------------------|
| 1.     | -       | 0          |                                  |
| 2.     | 4500    | 4500       |                                  |
| 3.     | 5500    | 10000      |                                  |
| 4.     | 3400    | 13400      |                                  |
| 5.     | 5600    | 19000      |                                  |
| 6.     | 3000    | 22000      |                                  |
| 7.     | 5000    | 27000      |                                  |
| 8.     | 450     | 27000      | Salary cannot be less than \$500 |
| 9.     | 9000    | 36000      |                                  |
| 10.    | 8900    | 44900      |                                  |
| 11.    | 4500    | 49400      | 49400                            |

# Quiz Time

- Draw a flowchart to print the product of the first 10 even numbers.
- Draw a flowchart to accept 50 numbers and also display the total number of odd and even numbers.

# Object Oriented Principles



# Overview of Object-Oriented Programming

- Object-Oriented Programming is a programming pattern that makes use of objects and their interactions to design and implement applications
- Objects are entities that serve as the basic building blocks of an object-oriented application
- An object is a self-contained entity with attributes and behaviors
- In some way everything can be an object.
- In general, an object is a person, place, thing, event, or concept.
- Because different people have different perceptions of the same object, what an object is depends upon the point of view of the observer.
  - That is, we describe an object on the basis of the features and behaviors that are important or relevant to us.

# Types of Objects

- Objects are usually classified as:
  - Objects representing physical things
    - e.g. students, furniture, buildings, classrooms
- Objects representing concepts
  - e.g., courses, departments, loan

# What Are Classes and Objects?

- A class:
  - Models an abstraction of objects
  - Defines the attributes and behaviors of objects
  - Is the blueprint that defines an object
- An object:
  - Is stamped out of the class mold
  - Is a single instance of a class
  - Retains the structure and behavior of a class





# An Object's Attributes Maintain Its State

- Objects have knowledge about their current state.
- Each piece of knowledge is called an attribute.
  - The values of attributes dictate the objects' state.



Object: My blue pen



Attribute: Ink amount



Object: Acme Bank ATM



Attribute: Cash available

# Objects Have Behavior

- An object exists to provide behavior (functionality) to the system.
- Each distinct behavior is called an operation.



Object: My blue pen



Operation: Write



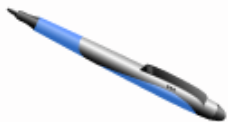
Object: Acme Bank ATM



Operation: Withdraw

# Objects Are Modeled as Abstractions

- A Java object is modeled as an abstract representation of a real-world object.
- Model only those attributes and operations that are relevant to the context of the problem.



- Context: Product catalog
- Real-world attributes/operations that you may want to model:
  - Attributes: Model, manufacturer, price
  - Operations: Change price
- Real-world attributes/operations that you may not want to model:
  - Attributes: Ink color
  - Operations: Refill, change color, point, write

# Defining Object Composition

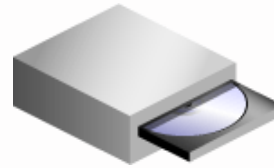
- Objects can be composed of other objects.
- Objects can be part of other objects.
- This relationship between objects is known as aggregation.



A PC may be an object.

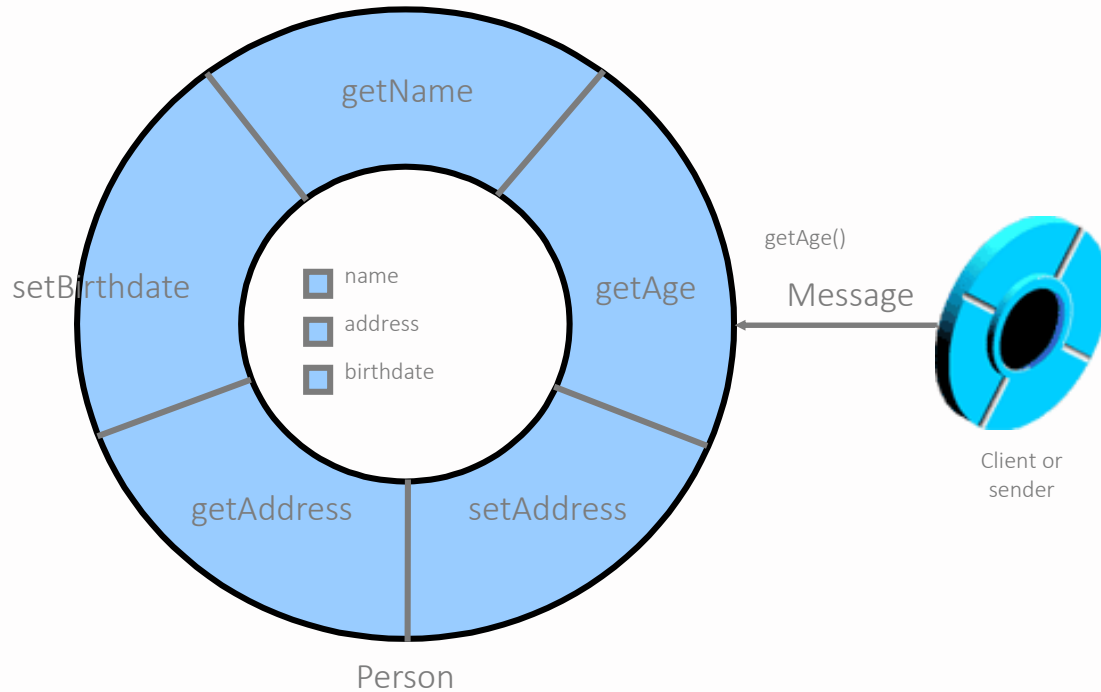


A PC may have a keyboard, mouse, and network card, all of which may be objects.



A PC may have a CD drive, which may be an object.

## Cont'd...



# Collaborating Objects

- Collaborating objects work together to complete a task and form the basis of an application system.
  - All methods are defined within a class and are not defined globally as in traditional languages.
  - All objects are created from classes and contain all the attributes and methods of that class.
  - Objects must associate with each other to collaborate on common tasks.
  - Associated objects communicate by sending messages.

# Objects Interact Through Messages

- Objects communicate by sending messages.
- A sending object must be associated with or linked to the receiving object.
- The message sender requests the receiver to perform the operation that is named in the message.
- This communication is similar to calling a procedure:
  - The sender calls a method of the receiver.
  - The receiver executes the called method.
- Calling a method is always in the context of a particular object:
  - `myPen.write( )`: Object-oriented programming
  - `write(myPen)`: Traditional structured programming

# What Is a Class?

- A class is a template for objects.
- A class definition specifies the operations and attributes for all instances of that class.
- A class is used to manage complexity.



When you create *my blue pen*, you do not have to specify its operations or attributes. You simply say what class it belongs to.



# How Do You Identify a Class?

- Identify the common behavior and structure for a group of objects.
- Recognize a single coherent concept.
- Caution: A common misconception is the use of the words classes and objects interchangeably. Classes define objects.



My blue pen

ops:

write, refill

attrs:

ink amount, color of ink



Your blue pen

ops:

write, refill

attrs:

ink amount

# Comparing Classes and Objects

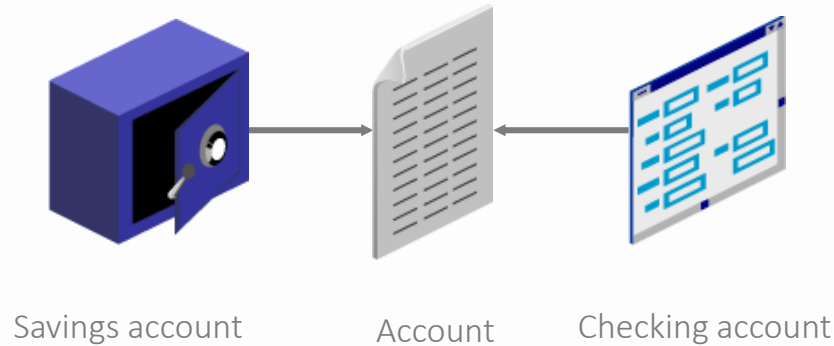
- Classes are static definitions that you can use to understand all the objects of that class.
- Objects are the dynamic entities that exist in the real world and your simulation of it.
- Caution: OO people almost always use the words *classes* and *objects* interchangeably; you must understand the context to differentiate between the two meanings.

# What Is Encapsulation?

- Encapsulation hides the internal structure and operations of an object behind an interface.
  - A bank ATM is an object that gives its users cash.
    - The ATM hides (encapsulates) the actual operation of withdrawal from the user.
    - The interface (way to operate the ATM) is provided by the keyboard functions, screen, cash dispenser, and so on.
    - Bypassing the encapsulation is bank robbery.
  - Bypassing encapsulation in object-oriented programming is impossible.

# What Is Inheritance?

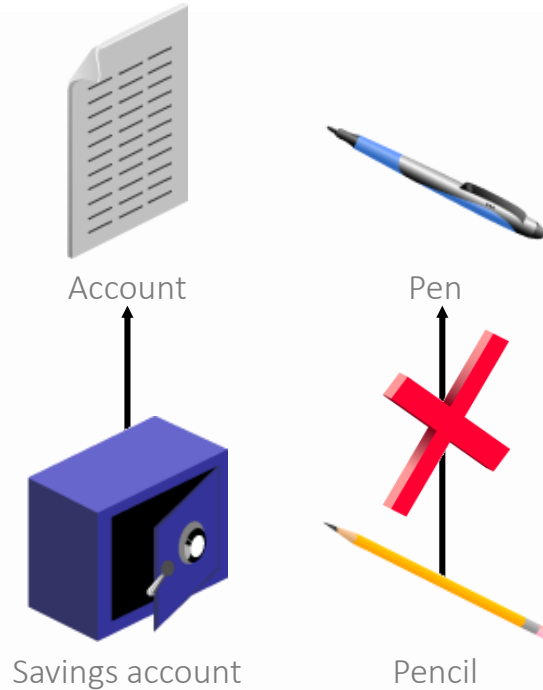
- There may be a commonality between different classes.
- Define the common properties in a superclass.



- The subclasses use inheritance to include those properties.

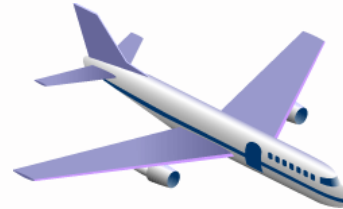
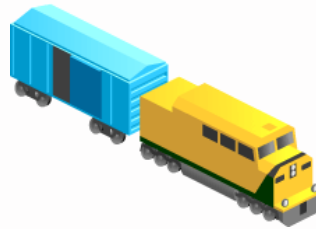
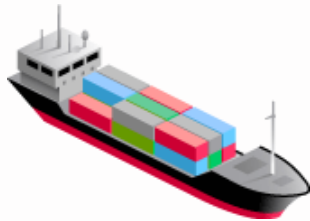
# Using the “Is-a-Kind-of” Relationship

- A subclass object “is-a-kind-of” superclass object.
- A subclass must have all the attributes and behaviors of the superclass.



# What Is Polymorphism?

- Polymorphism refers to:
  - Many forms of the same operation
  - The ability to request an operation with the same meaning to different objects. However, each object implements the operation in a unique way.
  - The principles of inheritance and object substitution.



Load passengers



Let's Solve