

CLASS -10 (2025-26)

INTRODUCTION TO

OBJECT ORIENTED

PROGRAMMING CONCEPT

CHAPTER 1

## Assignments:-

A. Tick (✓) the correct answer.

1. Which of the following is not a principle of OOP?

Answer: d. Class

2. Which of the following are the advantages of polymorphism?

Answer: c. Both a and b

(a. Codes can be reused, b. It makes the program run faster)

3. Which of the following principles of OOP allows the concept of reusability?

Answer: a. Polymorphism

4. Which of the following is the main element of object-oriented programming?

Answer: b. Objects

5. Procedural programming splits the programming code into small parts called

Answer: a. Procedures

B. Fill in the blanks.

A **paradigm** is a way of programming.

**Procedure-Oriented Programming** has global data sharing of functions.

A **low-level** language is a programming language that is machine-dependent.

The concept of **inheritance** is a good feature for avoiding data redundancy.

Java is an example of **object-oriented** programming language.

C. Short Answer Type Questions

What is the use of inheritance?

Answer: Inheritance allows a class to acquire the properties and behaviors (methods) of another class, promoting code reusability and reducing redundancy.

What does POP stand for?

Answer: POP stands for Procedure-Oriented Programming.

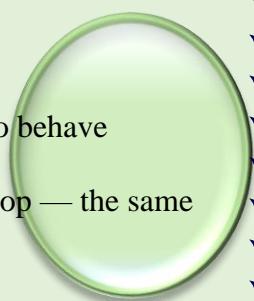
Define polymorphism with a real-life example.

Answer: Polymorphism means "many forms." It allows the same function or method to behave differently based on the object calling it.

Example: A person can be a teacher in school, a parent at home, and a customer in a shop — the same individual exhibiting different behaviors in different situations.

What are the disadvantages of Procedure-Oriented Programming?

Answer:





Difficult to manage large codebases.  
Poor data security due to global data access.  
Limited code reusability.  
Lack of scalability and maintainability.

### What are the differences between POP and OOP?

**Answer:**

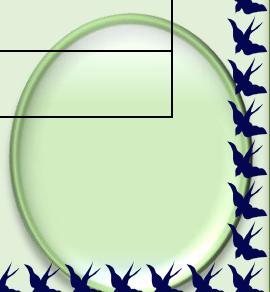
POP (Procedure-Oriented Programming)	OOP (Object-Oriented Programming)
Follows a <b>top-down</b> approach	Follows a <b>bottom-up</b> approach
Focuses on <b>procedures or functions</b>	Focuses on <b>objects and classes</b>
<b>Data is global</b> and shared among all functions	<b>Data is encapsulated</b> inside objects
Does <b>not support inheritance or polymorphism</b>	<b>Supports inheritance</b> and polymorphism
<b>Low data security</b> due to global access	<b>High data security</b> through encapsulation and access control
<b>Limited code reusability</b>	<b>High code reusability</b> through inheritance and modular design
<b>Harder to maintain and modify</b> as the program grows	<b>Easier to maintain and extend</b> due to modular structure
Examples: C, Pascal	Examples: Java, C++, Python (OOP features)

- **POP** follows a top-down approach; **OOP** follows a bottom-up approach.
- In **POP**, data is global; in **OOP**, data is encapsulated.
- **POP** focuses on procedures/functions; **OOP** focuses on objects.
- **OOP** supports inheritance and polymorphism; **POP** does not.
- **OOP** provides better data security and reusability than **POP**.

\*\*\*\*\*

## 1. Difference between Abstraction and Encapsulation

Abstraction	Encapsulation
Hides <b>implementation details</b> and shows only the <b>essential features</b> to the user.	Binds <b>data</b> and <b>functions</b> into a <b>single unit</b> (class) and hides internal data.
Focuses on <b>what</b> an object does.	Focuses on <b>how</b> data is protected and maintained.
Achieved using <b>abstract classes or interfaces</b> .	Achieved using <b>classes</b> and <b>access specifiers</b> (private, public, protected).
Example: Driving a car without knowing how the engine works.	Example: Data members are private, and access is provided via public methods.
Promotes <b>simplicity</b> .	Promotes <b>security</b> .



## 2. Difference between Encapsulation and Inheritance

Encapsulation	Inheritance
Encapsulation is the process of <b>binding data and methods</b> that operate on the data into a single unit.	Inheritance is the mechanism by which one class <b>acquires properties and behaviors</b> of another class.
It helps in <b>protecting data</b> from unauthorized access.	It helps in <b>code reusability</b> and creating a hierarchical relationship.
Achieved using <b>access modifiers</b> and <b>classes</b> .	Achieved using <b>extends</b> keyword in Java.
Example: Private data members with public getter and setter methods.	Example: A <code>Car</code> class inherits from a <code>Vehicle</code> class.
Promotes <b>data hiding</b> .	Promotes <b>reusability and extensibility</b> .

## 3. Difference between Inheritance and Polymorphism

Inheritance	Polymorphism
Enables a new class to <b>inherit</b> properties and behaviors from an existing class.	Allows methods to <b>perform differently</b> based on the object calling them.
Promotes <b>code reusability</b> .	Promotes <b>flexibility and dynamic behavior</b> in code.
Achieved using the <b>extends</b> keyword in Java.	Achieved using <b>method overloading</b> or <b>overriding</b> .
Example: <code>Dog</code> class inherits from <code>Animal</code> class.	Example: <code>draw()</code> method behaves differently for <code>Circle</code> and <code>Rectangle</code> .
Relationship is “ <b>is-a</b> ” (e.g., Dog is an Animal).	Relationship is “ <b>behaves differently</b> ” for same interface.

## 4. Difference between Abstraction and Inheritance

Abstraction	Inheritance
Hides <b>implementation details</b> and shows only essential features.	Allows one class to <b>reuse</b> code from another class.
Focuses on <b>what</b> to do, not <b>how</b> to do it.	Focuses on building a <b>hierarchical relationship</b> .
Achieved using <b>abstract classes</b> and <b>interfaces</b> .	Achieved using the <b>extends</b> keyword in Java.
Example: Interface <code>Shape</code> has <code>draw()</code> method with no body.	<code>Circle</code> class inherits <code>draw()</code> from <code>Shape</code> .
Promotes <b>simplicity and clarity</b> .	Promotes <b>code reusability and organization</b> .

## 5. Difference between Encapsulation and Polymorphism

Encapsulation	Polymorphism
Binds <b>data and methods</b> into a single unit and restricts access.	Allows <b>one interface</b> to be used for <b>different implementations</b> .
Achieved using <b>classes</b> and <b>access specifiers</b> .	Achieved through <b>method overloading</b> and <b>overriding</b> .
Focuses on <b>data hiding and security</b> .	Focuses on <b>dynamic behavior and flexibility</b> .
Example: Private variables with getter/setter methods.	Example: <code>print()</code> method works for integers, strings, etc.
Promotes <b>security and control</b> .	Promotes <b>extensibility and readability</b> .

## Multiple Choice Questions (MCQs) with Answers

1. Which feature of OOP binds data and functions that operate on the data into a single unit?

- a) Inheritance
- b) Polymorphism
- c) Abstraction
- d) Encapsulation

✓ **Answer:** d) Encapsulation

2. Which OOP principle allows a function or method to behave differently based on the object?

- a) Inheritance
- b) Polymorphism
- c) Encapsulation
- d) Abstraction

✓ **Answer:** b) Polymorphism

3 Which of the following languages is primarily based on OOP?

- a) C
- b) Assembly
- c) Java
- d) Pascal

✓ **Answer:** c) Java

4 In Procedure-Oriented Programming, data is mainly:

- a) Hidden inside classes
- b) Shared globally among functions
- c) Accessed only by objects
- d) Managed by constructors

✓ **Answer:** b) Shared globally among functions

5 Which of the following is not a benefit of Object-Oriented Programming?

- a) Code Reusability
- b) Better Data Security
- c) Procedural Flow Control
- d) Easier Maintenance

✓ **Answer:** c) Procedural Flow Control



6. Which of the following is an example of a low-level language?

- a) Java
- b) C++
- c) Assembly
- d) Python

✓ **Answer:** c) Assembly

7. The feature of OOP that hides unnecessary details from the user is called:

- a) Polymorphism
- b) Encapsulation
- c) Inheritance
- d) Abstraction

✓ **Answer:** d) Abstraction

8. Which of the following is not a valid concept in OOP?

- a) Modularity
- b) Global Variables
- c) Polymorphism
- d) Inheritance

✓ **Answer:** b) Global Variables

9. The class in OOP serves as a:

- a) Blueprint for objects
- b) Function library
- c) Database
- d) Data entry form

✓ **Answer:** a) Blueprint for objects

10. Which programming approach is best suited for large and complex applications?

- a) Procedural Programming
- b) Structured Programming
- c) Object-Oriented Programming
- d) Linear Programming

✓ **Answer:** c) Object-Oriented Programming

## Assertion and Reason Questions with Options

1.

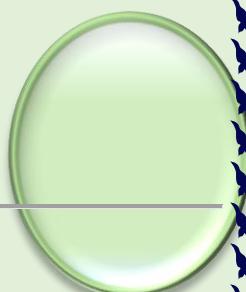
**Assertion (A):** Object-Oriented Programming provides better data security than Procedure-Oriented Programming.

**Reason (R):** OOP uses encapsulation to restrict direct access to data.

**Options:**

- a) Both A and R are true, and R is the correct explanation of A
- b) Both A and R are true, but R is not the correct explanation of A
- c) A is true, but R is false
- d) A is false, but R is true

✓ **Answer:** a) Both A and R are true, and R is the correct explanation of A





2.

**Assertion (A):** In POP, code reusability is high due to the use of global variables.

**Reason (R):** Global variables can be accessed by any function in POP.

**Options:**

- a) Both A and R are true, and R is the correct explanation of A
- b) Both A and R are true, but R is not the correct explanation of A
- c) A is false, but R is true
- d) A is true, but R is false

**✓ Answer:** c) A is false, but R is true

---

3.

**Assertion (A):** Inheritance helps reduce code redundancy.

**Reason (R):** Inheritance allows a class to reuse the properties of another class.

**Options:**

- a) Both A and R are true, and R is the correct explanation of A
- b) Both A and R are true, but R is not the correct explanation of A
- c) A is true, but R is false
- d) A is false, but R is true

**✓ Answer:** a) Both A and R are true, and R is the correct explanation of A

---

4.

**Assertion (A):** Polymorphism allows the same method to behave differently based on context.

**Reason (R):** It helps in defining multiple methods with the same name but different parameters or behavior.

**Options:**

- a) Both A and R are true, and R is the correct explanation of A
- b) Both A and R are true, but R is not the correct explanation of A
- c) A is true, but R is false
- d) A is false, but R is true

**✓ Answer:** a) Both A and R are true, and R is the correct explanation of A

---

5.

**Assertion (A):** Java is a procedural programming language.

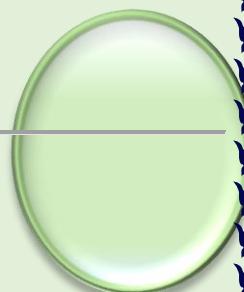
**Reason (R):** Java does not support classes and objects.

**Options:**

- a) Both A and R are true, and R is the correct explanation of A
- b) Both A and R are true, but R is not the correct explanation of A
- c) A is true, but R is false
- d) Both A and R are false

**✓ Answer:** d) Both A and R are false

---



6.

**Assertion (A):** POP provides more modular code than OOP.

**Reason (R):** In POP, the entire program is divided into procedures or functions.

**Options:**

- a) Both A and R are true, and R is the correct explanation of A



7. b) Both A and R are true, but R is not the correct explanation of A  
c) A is false, but R is true  
d) A is true, but R is false  
**✓ Answer:** c) A is false, but R is true

---

7. **Assertion (A):** Encapsulation is the process of hiding implementation details.  
**Reason (R):** Encapsulation restricts direct access to class members using access modifiers.  
**Options:**  
a) Both A and R are true, and R is the correct explanation of A  
b) Both A and R are true, but R is not the correct explanation of A  
c) A is true, but R is false  
d) A is false, but R is true  
**✓ Answer:** a) Both A and R are true, and R is the correct explanation of A

---

8. **Assertion (A):** OOP makes large software systems more manageable.  
**Reason (R):** OOP supports abstraction, encapsulation, and modularity.  
**Options:**  
a) Both A and R are true, and R is the correct explanation of A  
b) Both A and R are true, but R is not the correct explanation of A  
c) A is true, but R is false  
d) A is false, but R is true  
**✓ Answer:** a) Both A and R are true, and R is the correct explanation of A

---

9. **Assertion (A):** Low-level languages are machine-independent and portable.  
**Reason (R):** They use natural language for programming.  
**Options:**  
a) Both A and R are true, and R is the correct explanation of A  
b) Both A and R are true, but R is not the correct explanation of A  
c) A is true, but R is false  
d) Both A and R are false  
**✓ Answer:** d) Both A and R are false

---

10. **Assertion (A):** POP supports inheritance, which promotes reusability.  
**Reason (R):** Functions in POP can be called from anywhere in the program.  
**Options:**  
a) Both A and R are true, and R is the correct explanation of A  
b) Both A and R are true, but R is not the correct explanation of A  
c) A is false, but R is true  
d) A is true, but R is false  
**✓ Answer:** c) A is false, but R is true

---

