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.

1. A **paradigm** is a way of programming.
2. **Procedure-Oriented Programming** has global data sharing of functions.
3. A **low-level** language is a programming language that is machine-dependent.
4. The concept of **inheritance** is a good feature for avoiding data redundancy.
5. Java is an example of **object-oriented** programming language.

C. Short Answer Type Questions

1. **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.
2. **What does POP stand for?**
Answer: POP stands for **Procedure-Oriented Programming**.
3. **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.
4. **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.

5. **What are the differences between POP and OOP?**

6. **Answer:**

POP (Procedure-Oriented Programming)	OOP (Object-Oriented Programming)
Follows a **top-down** approach	Follows a **bottom-up** approach
Focuses on **procedures** or **functions**	Focuses on **objects** and **classes**
Data is global and shared among all functions	**Data is encapsulated** inside objects
Does **not support inheritance** or polymorphism	**Supports inheritance** and polymorphism
Low data security due to global access	**High data security** through encapsulation and access control
Limited code reusability	**High code reusability** through inheritance and modular design
Harder to maintain and modify as the program grows	**Easier to maintain and extend** 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 **implementation details** and shows only the **essential features** to the user.	Binds **data** and **functions** into a **single unit** (class) and hides internal data.
Focuses on **what** an object does.	Focuses on **how** data is protected and maintained.
Achieved using **abstract classes** or **interfaces**.	Achieved using **classes** and **access specifiers** (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 **simplicity**.	Promotes **security**.

2. Difference between Encapsulation and Inheritance

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

3. Difference between Inheritance and Polymorphism

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

4. Difference between Abstraction and Inheritance

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

5. Difference between Encapsulation and Polymorphism

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

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) Encapsulatio

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

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

****