

# Object-Oriented Programming Concepts Test

---

## Section A: Multiple Choice Questions (10 marks)

---

*Choose the best answer for each question.*

1. **Which of these is NOT a core principle of OOP?**
  - a) Encapsulation
  - b) Compilation
  - c) Polymorphism
  - d) Inheritance
  
2. **The process of bundling data and methods that operate on that data is called:**
  - a) Inheritance
  - b) Polymorphism
  - c) Encapsulation
  - d) Abstraction
  
3. **In C++, what is the default access specifier for class members?**
  - a) public
  - b) private
  - c) protected
  - d) friend
  
4. **Which keyword is used to implement inheritance in C++?**
  - a) inherits
  - b) extends
  - c) implements
  - d) : (colon)
  
5. **A function that is called automatically when an object is created is called:**
  - a) Destructor
  - b) Constructor
  - c) Member function
  - d) Static function
  
6. **Which type of polymorphism is achieved through function overloading?**
  - a) Runtime polymorphism
  - b) Compile-time polymorphism
  - c) Virtual polymorphism
  - d) Dynamic polymorphism
  
7. **What does the 'static' keyword indicate when applied to a class member?**
  - a) The member cannot be changed

- b) The member belongs to the class, not individual objects
- c) The member is constant
- d) The member is private

8. Which operator is used to define a member function outside its class?

- a) ->
- b) .
- c) ::
- d) :

9. What is the main advantage of using inheritance?

- a) Data hiding
- b) Code reusability
- c) Faster execution
- d) Smaller code size

10. Which of these relationships represents an "is-a" relationship?

- a) Car has Engine
- b) Dog is Animal
- c) Book has Pages
- d) Student has ID

---

## Section B: Short Answer Questions (15 marks)

---

*Answer the following questions concisely.*

1. Explain the difference between a class and an object with an example. (2 marks)
2. What is the purpose of access specifiers (public, private, protected) in OOP? (2 marks)
3. Differentiate between function overloading and function overriding. (2 marks)
4. Why would you declare a member function or variable as 'static'? Provide an example. (2 marks)
5. Explain the concept of constructor overloading with an example. (2 marks)
6. What is a virtual function and why is it important in polymorphism? (2 marks)
7. How does encapsulation provide data security in OOP? (3 marks)

---

## Section C: Code Analysis (10 marks)

---

*Analyze the following code snippets and answer the questions.*

1. What will be the output of the following code? (2 marks)

```
#include <iostream>
using namespace std;

class Test {
    static int count;
public:
    Test() { count++; }
    static void showCount() { cout << count; }
};

int Test::count = 0;

int main() {
    Test t1, t2;
    Test::showCount();
    return 0;
}
```

2. Identify errors in the following code and correct them: (3 marks)

```
class Rectangle {
    int width, height;
public:
    Rectangle(int w, int h) : width(w), height(h) {}
    int area() { return width * height; }
};

int main() {
    Rectangle r;
    cout << r.area();
    return 0;
}
```

3. What is the problem with the following code and how would you fix it? (2 marks)

```
class Student {
    char *name;
public:
    Student(char *n) {
        name = new char[strlen(n)+1];
        strcpy(name, n);
    }
    void display() { cout << name; }
};
```

4. What will be the output of this program? (3 marks)

```
#include <iostream>
using namespace std;

class Base {
public:
    virtual void show() { cout << "Base\n"; }
};

class Derived : public Base {
public:
    void show() { cout << "Derived\n"; }
};

int main() {
    Base *b;
    Derived d;
    b = &d;
    b->show();
    return 0;
}
```

---

### Answer Key

*(For instructor use only)*

## Section A: Multiple Choice Questions

1. b) Compilation
2. c) Encapsulation
3. b) private
4. d) : (colon)
5. b) Constructor
6. b) Compile-time polymorphism
7. b) The member belongs to the class, not individual objects
8. c) ::
9. b) Code reusability
10. b) Dog is Animal

## Section B: Short Answer Questions

1. A class is a blueprint or template that defines the attributes and behaviors of objects. An object is an instance of a class. Example: Class: Car, Object: My specific red Toyota car.
2. Access specifiers control the visibility and accessibility of class members. Public members are accessible from anywhere, private members are only accessible within the class, and protected members are accessible within the class and its derived classes.

3. Function overloading involves multiple functions with the same name but different parameters within the same scope (compile-time polymorphism). Function overriding involves redefining a base class function in a derived class with the same signature (runtime polymorphism).
4. A static member belongs to the class rather than individual objects. It's used when you need a single copy shared by all objects. Example: A counter that tracks how many objects of a class have been created.
5. Constructor overloading means having multiple constructors with different parameter lists. Example: A Date class with constructors Date(), Date(int d), and Date(int d, int m, int y).
6. A virtual function is a member function declared in a base class that can be redefined in derived classes. It enables runtime polymorphism by allowing the correct function to be called based on the actual object type rather than the reference/pointer type.
7. Encapsulation provides data security by bundling data with methods that operate on that data and restricting direct access to the data. Data can only be accessed or modified through well-defined interfaces (methods), allowing for validation, control, and maintenance without affecting external code.

## Section C: Code Analysis

1. Output: 2 (The static count variable is shared by all objects and incremented with each constructor call)
2. Error: The default constructor is not defined but called in main. Correction: Either provide a default constructor or create the object with parameters.
3. Problem: Memory leak - the allocated memory for name is never freed. Fix: Add a destructor that deletes[] name.
4. Output: Derived (Due to the virtual function, the show() method of the Derived class is called even though the pointer is of Base type)