X

Week 10: Assignment 10

The due date for submitting this assignment has passed.

Due on 2025-04-02, 23:59 IST.

Assignment submitted on 2025-03-27, 20:18 IST

1) Which of the following is the primary purpose of the Singleton pattern? ***1 point***

- To allow multiple instances of a class.
- To restrict a class to a single instance and provide global access to it.
- To create objects based on a condition.
- To improve inheritance in a class hierarchy.

Yes, the answer is correct.
Score: 1

Accepted Answers:
To restrict a class to a single instance and provide global access to it.

2) Analyze the following C++ implementation of the Singleton pattern: ***1 point***

```
class Singleton {
private:
    static Singleton* instance;
    Singleton() {}

public:
    static Singleton* getInstance() {
        if (!instance) {
            instance = new Singleton();
        }
        return instance;
    }
};

Singleton* Singleton::instance = nullptr;

int main() {
    Singleton* obj1 = Singleton::getInstance();
    Singleton* obj2 = Singleton::getInstance();
    if (obj1 == obj2) {
        std::cout << "Same instance";
    }
    return 0;
}
```

What is the output of this program?

- (selected) Same instance
- Different instance
- Compilation error
- Undefined behavior

Yes, the answer is correct.
Score: 1

Accepted Answers:
Same instance

3) To write a Java implementation of the Singleton pattern that ensures thread safety. Which of the following is a correct implementation? ***1 point***

- Use synchronized blocks in getInstance().
- Use the volatile keyword with the instance variable.
- Use a static block for initialization.
- (selected) All of the above.

Yes, the answer is correct.
Score: 1

Accepted Answers:
All of the above.

4) To write a C++ implementation of the Factory pattern to create objects of classes Circle and Square, both inheriting from an abstract base class Shape. Which of the following correctly defines the createShape() method? ***1 point***

- (selected) Shape* createShape(const std::string& type)
- Circle* createShape(const std::string& type)
- Square* createShape(const std::string& type)
- Shape createShape(const std::string& type)

Yes, the answer is correct.

Score: 1

Accepted Answers:
Shape createShape(const std::string& type)*

5) What is the primary purpose of the Observer pattern? ***1 point***

- To create a single global instance of a class.
- To allow one object to notify multiple objects of a change in its state.
- To encapsulate algorithms in separate classes.
- To improve memory management in object-oriented design.

Yes, the answer is correct.
Score: 1

Accepted Answers:
To allow one object to notify multiple objects of a change in its state.

6) To write a Java program to demonstrate the Observer pattern using: ***1 point***

- An Observer interface.
- A concrete Subject class to notify observers.
- Multiple concrete observer classes that react to state changes.

Which of the following correctly implements the notify() method in the Subject class?

- Iterate through the list of observers and call their update() method.
- Use a static method to update all observers.
- Use inheritance to notify all observers.
- Store observer states in a database.

Yes, the answer is correct.
Score: 1

Accepted Answers:
Iterate through the list of observers and call their update() method.

7) Which of the following statements is true about Singleton, Factory, and Observer patterns? ***1 point***

- The Singleton pattern is used for global access, the Factory pattern for object creation, and the Observer pattern for state management.
- The Factory pattern is a creational pattern, while Singleton and Observer are structural patterns.
- The Observer pattern is always implemented using threads.
- The Singleton pattern ensures that an object is immutable.

Yes, the answer is correct.
Score: 1

Accepted Answers:
The Singleton pattern is used for global access, the Factory pattern for object creation, and the Observer pattern for state management.

8) Which of the following use cases combines multiple design patterns? ***1 point***

- A logging system using Singleton for a logger instance and Observer for notifying log subscribers.
- A database system using Factory for creating connections and Observer for managing connection pools.
- Both A and B.
- Design patterns cannot be combined.

Yes, the answer is correct.
Score: 1

Accepted Answers:
Both A and B.

9) Consider the following C++ implementation where the Adapter Pattern is used to make a legacy OldPrinter class compatible with a modern PrinterInterface. ***1 point***

```
class PrinterInterface {
public:
    virtual void print() = 0;
};
class OldPrinter {
public:
    void legacyPrint() {
        std::cout << "Legacy printing...\n";
    }
};
class PrinterAdapter : public PrinterInterface {
private:
    OldPrinter* oldPrinter;
public:
    PrinterAdapter(OldPrinter* printer) : oldPrinter(printer) {}

    void print() override {
        oldPrinter->legacyPrint();
    }
};
int main() {
    OldPrinter oldPrinter;
    PrinterInterface* printer = new PrinterAdapter(&oldPrinter);
    printer->print();
```

```
    delete printer;
    return 0;
}
```

Which of the following statements is true about the adapter pattern in the above code?

- The PrinterAdapter class allows OldPrinter to be used directly without any modifications to the OldPrinter class.
- The PrinterAdapter class replaces the OldPrinter class completely.
- The PrinterAdapter class extends PrinterInterface and provides a new method legacyPrint().
- The PrinterAdapter class requires OldPrinter to inherit from PrinterInterface for the code to compile.

Yes, the answer is correct.
Score: 1

Accepted Answers:
The PrinterAdapter class allows OldPrinter to be used directly without any modifications to the OldPrinter class.

10) Which of the following is a primary purpose of the Proxy Pattern? ***1 point***

- To allow an object to send notifications to all its observers.
- To provide a surrogate or placeholder for another object to control access and interactions.
- To enable the creation of multiple independent instances of a class.
- To ensure that a class only has one instance throughout the application.

Yes, the answer is correct.
Score: 1

Accepted Answers:
To provide a surrogate or placeholder for another object to control access and interactions.