National University of Computer & Emerging Sciences, Karachi
Spring-2025 School of Computing
Quiz No. 2
25th March 2025

Course Code: CS-1004	**Course Name: Object Oriented Programming**
Instructor Name / Names: Ms. Atiya Jokhio	
Section-C	**Student-ID:**

Time Allowed: 30 minutes. **Total Points**: 10

TYPE A

Question:1 **[3 points]**

a) Briefly answer the following questions:

i) Which function is used to write the raw bytes of an object into a binary file?

fout.write()

ii) If a class has private data members, how can you save its object using write()?

No problem; write() directly accesses memory.

iii) What is the main difference in the virtual destructor and pure virtual destructor

iv) if a class contains a pure virtual destructor, it must provide a function body for the pure virtual destructor.

Question:2 **[6 points]**

Design an abstract base class Player with a function displayType() that shows the type of player (implemented inside the base class) and a **pure virtual function** makeMove() to be overridden by all derived classes.

- Create three derived classes: Each class must implement its specific version of makeMove().
  - HumanPlayer
  - EasyAIPlayer
  - HardAIPlayer

- Demonstrate **runtime polymorphism** by:
  - Creating an array of Player*. (**Hint: Player* players[numPlayers];)**
  - Storing different types of players in the container.
  - Looping through the players and calling both displayType() and makeMove() using only Player* pointers.

You also need to ensure **proper memory management**.

Solution:

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

// Abstract Base Class
```

```
class Player {
public:
    void displayType() {
        cout << "Generic Player" << endl;
    }

    virtual void makeMove() = 0; // Pure virtual function

    virtual ~Player() {} // Virtual destructor for safe deletion
};

// Derived Class: HumanPlayer
class HumanPlayer : public Player {
public:
    void makeMove() override {
        cout << "Human player selects a cell manually.\n";
    }
};

// Derived Class: EasyAIPlayer
class EasyAIPlayer : public Player {
public:
    void makeMove() override {
        cout << "Easy AI randomly chooses an available cell.\n";
    }
};

// Derived Class: HardAIPlayer
class HardAIPlayer : public Player {
public:
    void makeMove() override {
        cout << "Hard AI uses strategic algorithm (like minimax) to select the best move.\n";
    }
};

int main() {
    const int numPlayers = 3;          // We have three types of players
    Player* players[numPlayers];       // Array of Player pointers

    // Create different types of players dynamically
    players[0] = new HumanPlayer();
    players[1] = new EasyAIPlayer();
    players[2] = new HardAIPlayer();

    // Loop through players and use polymorphism
    for (int i = 0; i < numPlayers; ++i) {
        players[i]->displayType();
```

```
        players[i]->makeMove();
        cout << "------------------------------\n";
    }

    // Proper memory cleanup
    for (int i = 0; i < numPlayers; ++i) {
        delete players[i];
    }

    return 0;
}
```