|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course:** | **Software Design and Analysis** | **Course Code:** | **CS-3004** |
| **Program:** | **BS (Computer Science)** | **Semester:** | **Fall 2024** |
| **Duration:** | **40 Minutes** | **Total Marks:** | **20** |
| **Quiz Date:** | **27-Nov-24** | **Roll No.** |  |
| **Section:** | **BCS-5B** | **Name:** |  |
|  |  |  |  |
|  |  | | | |

**Question 1)** Refactor the following ShapeFactory to use the **Factory Method Pattern**. **(10 Marks)**

|  |  |
| --- | --- |
| class Shape {  public:  virtual void draw() = 0;  };  class Circle : public Shape {  public:  void draw() override {  cout << "Drawing Circle" << endl;  }  };  class Rectangle : public Shape {  public:  void draw() override {  cout << "Drawing Rectangle" << endl;  }  };  class ShapeFactory {  public:  static Shape\* getShape(const string& type) {  if (type == "Circle") {  return new Circle();  }  else if (type == "Rectangle") {  return new Rectangle();  }  }  }; | **Solution:**  class ShapeFactory {  public:  virtual Shape\* createShape() = 0;  };  class CircleFactory : public ShapeFactory {  public:  Shape\* createShape() override {  return new Circle();  }  };  class RectangleFactory : public ShapeFactory {  public:  Shape\* createShape() override {  return new Rectangle();  }  }; |

**Question 2)** Identify and mention the design pattern that ensures there is only one instance of Database in the application and refactor the code.  **(10 Marks)**

|  |  |
| --- | --- |
| class Database {  public:  void query(const string& sql) {  cout << "Executing query: " << sql << endl;  }  }; | **Solution: Singleton Pattern**  class Database {  private:  static Database\* instance; // Static pointer to the single instance  Database() {  cout << "Database instance created" << endl;  }  public:  static Database\* getInstance() {  if (instance == nullptr) {  instance = new Database();  }  return instance;  }  void query(const string& sql) {  cout << "Executing query: " << sql << endl;  }  };  // Initialize the static member  Database\* Database::instance = nullptr; |