**The object-oriented programming**

**Exam questions**

|  |  |
| --- | --- |
| 1 | Describe the fundamental concepts of object-oriented programming |
| 2 | Describe the definitions of objects and classes |
| 3 | Write about the concept of inheritance and its specifications |
| 4 | Write about the concept of encapsulation and its specifications |
| 5 | Write about the concept of polymorphism and its specifications |
| 6 | Write about the concept of abstraction and its specifications |
| 7 | Describe the creation of constructors and destructors in classes |
| 8 | Describe the structs and enumerators. Compare them with each other |
| 9 | Write about the constructor overloading and how it can be used with multiple classes |
| 10 | Write about Static members and methods. Create examples where a static method is used |
| 11 | Describe Sealed and partial classes. Write about their specifications |
| 12 | Specify Array and List collections. Explain the differences between arrays and lists in C# |
| 13 | Write about the Dictionary and how a Dictionary in C# works. Provide an example of how to add, retrieve, and check for an item in a dictionary |
| 14 | Describe Queue and Stack. Write about their specifications and differences between a Queue and a Stack |
| 15 | Describe LINQ with Collections. Explain how LINQ can be used to query and manipulate collections in C# |
| 16 | Write about designing the Windows Forms application. Give examples of properties and specifications of these kinds of applications |
| 17 | List and describe five common controls used in Windows Forms applications |
| 18 | Write about Form Properties. Describe key properties of the Form class in Windows Forms |
| 19 | Characterize different types of dialog boxes available in Windows Forms and how they are used |
| 20 | Describe the implementation of input validation in Windows Forms |
| 21 | Define Classes and Objects. Write a C# class called Book with properties Title, Author, and ISBN. Show how you would create an instance of this class |
| 22 | Implement Inheritance. Create a base class Vehicle with common properties and methods. Then, derive two classes from it, Car and Bicycle, adding specific properties or methods to each |
| 23 | Implement Polymorphism. Given a method that takes a Vehicle object as a parameter, explain how it can accept objects of Car or Bicycle due to polymorphism. Provide a code example |
| 24 | Define Abstract Classes and Interfaces. Create an abstract class Shape with an abstract method CalculateArea(). Then, define an interface IColorable with a method Color(). Implement these in a Circle class and explain the difference in usage between the abstract class and the interface |
| 25 | Implement Constructor Overloading. Write two constructors for a Rectangle class, one that takes no parameters and another that takes *length* and *width* |
| 26 | Implement Method Overloading and Overriding. In a class hierarchy involving Animal and Dog, demonstrate method overloading and overriding. Provide code snippets where Dog overrides a method from Animal and also overloads one of its own methods |
| 27 | Create Static Memners. Explain the purpose of static members in a class. Create a class Calculator with a static method Add that takes two numbers and returns their sum |
| 28 | Implement List creation and manipulation. Write a C# program to create a List and add 5 integers to it. Show how to iterate through the list and print each value |
| 29 | Implement Stack for LIFO. Create a Stack and show how to push and pop elements. Use a simple example to demonstrate the Last-In-First-Out (LIFO) behavior of a stack |
| 30 | Implement Queue Operations. Demonstrate how to create a Queue and perform basic operations like enqueue, dequeue, and peek. Show how you would handle the queue being empty before a dequeue operation |