GLA University, 2019

BCSC0002: Object-Oriented Programming

Practice Set 1

1. What is Object-Oriented programming? How is it different from procedural programming?
2. How are data and methods organized in an object-oriented program?
3. What are the unique advantages of an object-oriented programming paradigm?
4. Distinguish between the terms -
    1. Objects and Classes
    2. Data Abstraction and Data Encapsulation
    3. Data Abstraction and Data Hiding
    4. Inheritance and Polymorphism
5. What do you understand by inheritance in Object-Oriented programming?
6. State whether `TRUE` or `FALSE`
    1. In conventional, procedure-oriented programming, all data are shared by all functions.
    2. The main emphasis of procedure-oriented programming is on algorithms rather than on data.
    3. One of the striking features or object-oriented programming is the division of programs into objects that represent real-world entities.
    4. Wrapping up of data of different types into a single unit is known as encapsulation.
    5. One problem with OOP is that once a class is created, it can never be changed.
    6. Inheritance means the ability to reuse the data values of one object by other objects.
    7. Polymorphism is extensively used in implementing Inheritance.
    8. Object-Oriented programs are executed much faster than conventional programs.
    9. Object-Oriented systems can scale up better from small to large audience.
    10. Object-Oriented approach cannot be used to create databases.
7. Describe the structure of a typical Java program.
8. Why do we need the import statement?
9. What is the task of the `main()` method in a Java program?
10. What is a token? List the various types of tokens supported by Java.
11. Why can't we use a keyword as a variable name?
12. Enumerate the rules for creating identifier in Java.
13. What are the conventions followed in Java for naming identifiers? Give examples.
14. What are separators? Describe the various separators used in Java.
15. What is a statement? How do the Java statements differ from those of C and C++?
16. Describe in detail the steps involved in implementing a standalone program.
17. What are command line arguments? How are they useful?

18. Java is a free-form language. Comment.
19. What are the basic steps in writing the Object-Oriented Application?
20. What are the important class relationships?
21. What are the reasons for implementing inheritance relationship?
22. List the eight basic data types in Java. Give examples.
23. What is the scope of a variable?
24. What is type casting? Why is it required in programming?
25. What is initialization? Why is it important?
26. What are literals? Give examples?
27. Determine whether the following are `TRUE` or `FALSE`
    1. When **if** statements are nested, the last **else** gets associated with the nearest **if** without an **else**.
    2. One **if** can have more than one **else** clause.
    3. A **switch** statement can always be replaced by a series of **if...else** statements.
    4. A program stops its execution when a **break** statement is encountered.
28. In what ways does a `switch` statement differ from an `if` statement?
29. Differentiate between
    1. `while` and `do ... while`
    2. `while` and `for`
    3. `break` and `continue`
    4. `for` and `for-each`
30. What is a `class`? How does it accomplish data hiding?
31. How do classes help us to organize our programs?
32. What are the three parts of a simple, empty class?
33. What are `objects`? How are they created from a class?
34. How do you define a method signature?
35. When do we declare a member of a class **static**?
36. How do we invoke (call) a constructor of a class?
37. What is inheritance and how does it help us create new classes quickly?
38. Describe different forms of inheritance with examples.
39. Describe the syntax of single inheritance in Java.
40. Compare and contrast method overloading and method overriding.
41. When do we declare a method or class **final**?
42. When do we declare a method or class **abstract**?
43. Discuss the different levels of access protection available in Java.
44. Design a class to represent a bank account. Include the following members
    1. Data Members (Fields)
        1. Name of the Depositor: String
        2. Type of Account: String (SB - Savings, CB - Current, SA - Salary, FD - Fixed Deposit)
        3. Account Number: long
        4. Balance Amount in the Account: Double
    2. Methods

    1. A method to assign initial values to the fields.
    2. To withdraw an amount after checking balance.
    3. To deposit an amount.
    4. To display the name and balance.

45. What is an array?
46. Why are arrays easier to use compared to a bunch of related variables?
47. Write a statement to declare and instantiate an array to hold marks obtained by students in different subjects in a class. Assume that there are up to 70 students in a class and there are 8 subjects in total.
48. How does the `String` class differ from the `StringBuffer` class?
49. What is an interface?
50. How do we tell Java that the class we are creating implements a particular interface?
51. What is the major difference between a class and an interface?
52. What are the similarities between interfaces and classes?
53. Describe the various forms of implementing interfaces. Give examples of Java code for each case.
54. Give an example where interface can be used to support multiple inheritance. Develop a standalone Java program for the example.
55. What is a package?
56. How do we tell Java that we want to use a particular package in a file?
57. How do we design a package?
58. How do we add a class or an interface to a package?
59. Discuss the various levels of access protection available for packages and their implications.
60. What is a `static import`? How is it useful?
61. What is an exception?
62. How do we define a `try` block?
63. How do we define a `catch` block?
64. List some of the most common types of exceptions that might occur in Java. Give code examples.
65. Is it essential to catch all types of exceptions.
66. How many `catch` blocks can we use with one `try` block.
67. Create a `try` block that is likely to generate three types of exceptions and then incorporate necessary `catch` blocks to catch and handle them appropriately.
68. What is a `finally` block? When and how is it used? Give suitable example.
69. Explain how exception handling mechanism can be used for debugging a program.
70. Define an exception called 'NoMatchException' that is thrown when a String is not equal to "India". Write a program that uses this exception.