

MANAGMENT DEVELOPMENT INSTITUTE

Diploma in Computer Science

Java I I

Final Exam Time: 2hrs

9th December 2024

SECTION 1

**Answer all questions in this section. All questions carry equal marks.**

Good Luck!

1. The granddaddy of everything in Java is said to be.
   * 1. Class
     2. Main method
     3. Node
     4. An Object
2. Which class defined toString() function which can be overridden by every subclass.
   * 1. Object class
     2. String class
     3. ArrayList class
     4. Student class
3. The ArrayList<> can store any type including user-defined types. This is because the ArrayList class defined a type parameter. All this is possible because of\_\_
   * 1. Object Oriented Programming
     2. Functional Programming
     3. Generic Programming
     4. None of the above
4. Enums are used to define\_\_ types for variables.
   * 1. Dynamic
     2. Predefine
     3. Static
     4. Node of the above
5. Study the code snippets below to answer the following questions(5, 6, and 7).

class Car {

String name;

String engine;

int seats;

int wheels;

}

With Object-Oriented Programming, we can define our type by creating a class that represents a real-world entity. For example, the Car class refers to what types?

* 1. User define type
  2. String data type
  3. Object
  4. Constructor

1. The four variables defined at the class level are also called\_\_ variable. a. Car
   1. Class
   2. Object
   3. Instance
2. The Car class does not explicitly define a constructor, but when creating a car object, Java will create a default construct under the hook. What is this constructor called?
   1. No-args constructor
   2. Super constructor
   3. Class constructor
   4. Object constructor
3. A function/sub-routine is a species of code that executes whenever called or invoked.
   1. true
   2. false
4. A stack data structure that stores items in a\_\_ order.
   1. last in last out
   2. last in first out
   3. first in first out
   4. first in last out
5. What data structure does the stack utilize to store items internally?
   1. object
   2. list
   3. array
   4. trees
6. A data structure that stores items/elements using an index base is called.
   1. an array
   2. linked list
   3. trees
   4. all of the above
7. Which operation in LinkedList does not require changing its structure. a. insertion
   1. deletion
   2. displaying
   3. none of the above
8. Study the algorithm below and figure out its time and space complexity(questions; 13 and 14).

int[] temp = new int[this.data.length \* 2];

for (int i = 0; i < data.length - 1; i++) {

temp[i] = data[i];

}

The above code doubles an Array. What is the space complexity of the algorithm? a. O(1)

* 1. O(n)
  2. both a and b
  3. none of the above

1. What is the time complexity of traversing through the data array?
   1. constant time O(1)
   2. The time is relative to the number of items in the data array, O(N)
   3. time-complexity not applicable
   4. none of the above
2. The below snippet is a Node class it has a value of type int and a next pointer pointing to a next Node.

private class Node {

int value;

Node next;

}

What type of a LinkedList?

* 1. singly
  2. doubly
  3. circular
  4. a and b

1. All the rest are principles of Object Oriented Programming except.
   1. implements
   2. abstraction
   3. encapsulation
   4. polymorphism
2. A recursive function is a function that calls itself, repetitively until a base case is reached.
   1. true
   2. false
   3. both a and b
   4. none the above
3. A data structure where insertion happens at the end and removal happens at the front.
   1. Set
   2. HashMap
   3. Stack
   4. Queue
4. In a LinkedList, the next pointer(s) are referential variables stored in the stack memory that point to an actual object stored in the heap memory. a. false

b. true

1. Suppose end = 3; val will be inserted at which index.

data[end++] = val;

* 1. at index 2
  2. at index 3
  3. at index 4
  4. none of the above

1. Study the code below.

static int sum(int a, int b){

return a + b;

}

static double sum(double a, int b){

return a + b;

}

What programming technique allows you to write two or more functions/methods with the same but different method signatures?

* 1. method overridden
  2. method overloading
  3. recursion
  4. generic programming

1. Study the code below and choose the right option.

public void insert(int val) {

if (isFull()) {

throw new IndexOutOfBoundsException("Queue is full cannot add item.");

}

data[end++] = val;

end = end % data.length;

size++;

}

What will happen when the condition evaluates to true?

* 1. runtime error
  2. compile time error
  3. both a and b
  4. none of the above

1. If a class extends a class it becomes\_\_
   1. super class
   2. sub class
   3. an object
   4. a constructor
2. Objects are stored in the heap memory while their references are stored in\_\_ memory.
   1. virtual
   2. stack
   3. array
   4. static
3. Below is a CircularQueue with two constructors. One is a no-args constructor and the other defines an argument. When creating a circular queue object it will be mandatory to pass an argument as the size of the internal data array.

public class CircularQueue {

/\*

....

\*/

public CircularQueue() {

data = new int[DEFAULT\_SIZE];

}

public CircularQueue(int size) {

data = new int[size];

}

}

1. yes
2. no
3. When working with data structures and performing certain operations will result in a change in the structure of the data structure. A peek operation on a Stack will result in a change in its structure.

|  |
| --- |
| public Task peek(){  if (isEmpty()) {  System.out.println("Cannot peek from an empty stack!");  return null;  }  return data[pointer];  } |

* + 1. yes
    2. no

1. When a variable is defined within the body of a function is said to be in a\_\_ scope.
   * 1. global
     2. local
     3. function
     4. internal
2. A data structure that stores items in a key-value pair in Java is said to be\_\_ a. Set
   * 1. Queue
     2. Map
     3. ArrayList
3. All the following are linear data structures except.
   * 1. Tree
     2. ArrayList
     3. Stack
     4. LinkedList
4. Iterating through a linear data structure, the time complexity of an algorithm depends on its size.
   * 1. true
     2. false