CPT204 2024
Final Exam Question

Short Answer Questions

1. A primitive variable stores the actual values of primitive data types directly, while a reference variable stores the address of an object in memory.

2. Overriding is a feature in Java where a subclass provides a specific implementation of a method that is already provided by its parent class, whereas Overloading is a feature that allows a class to have more than one method having the same name, but with different parameter lists.

3. The outputs of the following code are false and false.

```
public class Test {
   public static void main(String[] args) {
      Object o1 = new Object();
      Object o2 = new Object();
      System.out.print((o1 == o2) + "and" + (o1.equals(o2)));
   }
}
```

4. An interface is a class-like construct that contains only constants, abstract methods, default methods, and static methods. In many ways, an interface is similar to an abstract class, but an abstract class can contain data fields.

5. The output from the following code is [New York, Dallas]

```
public class Main {
      public static void main(String[] args) {
            ArrayList<String> list = new ArrayList<String>();
            list.add("New York");
            ArrayList<String> list1 = (ArrayList<String>) (list.clone());
            list.add("Atlanta");
            list1.add("Dallas");
            System.out.println(list1);
      }
}
```

6. The program below would encounter a compilation error because the Fruit class does not implement the java.lang. Comparable interface and the Fruit objects are not comparable. This is due to the problem in Line 4.

```
1.public class Test {
2.       public static void main(String[] args) {
3.              Fruit[] fruits = {new Fruit(2), new Fruit(3), new Fruit(1)};
4.            Arrays.sort(fruits);
5.      }
6.}

class Fruit {
   private double weight;

   public Fruit(double weight) {
      this.weight = weight;
   }
}
```

7. A generic class or method permits you to specify allowable types of objects that the class or method can work with. If you attempt to use a class or method with an incompatible object, the compiler will detect the error.

8. The method header is left blank in the following code. Fill in a generic method header ______.

```
public class GenericMethodDemo {
   public static void main(String[] args ) {
      Integer[] integers = {1, 2, 3, 4, 5};
      String[] strings = {"London", "Paris", "New York", "Austin"};

      print(integers);
      print(strings);
   }

  public static <T>   void   print(T[]  list) {
      for (int i = 0; i < list.length; i++)
         System.out.print(list[i] + " ");
      System.out.println();
   }
}
```

9. In Java collections, the ArrayList is efficient for retrieving elements and for adding and removing elements at the end of the list. On the other hand, a LinkedList is better suited for adding and removing elements at any position within the list, although locating specific elements for operations is not as efficient.

10. Suppose that list1 is a list that contains the strings red, yellow, and green, and that list2 is another list that contains the strings red, yellow, and blue. After executing list1.remove(list2), list1 contains green.

11. Stacks store objects that are processed in a last-in, first-out fashion. Queues

store objects that are processed in a first-in, first-out fashion.

12. The output from the following code is 10 20 30. 40 50 60

```
public class Test {
   public static void main(String[] args) {
      PriorityQueue<Integer> queue =
         new PriorityQueue<Integer>(
           Arrays.asList(20, 40, 60, 10, 50, 30));

      while (!queue.isEmpty())
         System.out.print(queue.poll() + " ");
   }
}
```

13. Show the output of the following code [ABC, FGH]

```
Set<String> set = new LinkedHashSet<>();
set.add("ABC");
set.add("FGH");
System.out.println(set);
```

14. Complete the following code to create an empty hash set of integers:
Set<Integer> set = new HashSet<;>( )

15. In a library management system where you need to frequently check the availability and information of books with unique identifiers (e.g., the book ID), the most suitable data structure to store the books in the library would be a Map.

16. The Big O notation is a theoretical approach for analyzing the performance of an algorithm. It estimates how fast an algorithm's execution time increases as the input size increases, which enables you to compare two algorithms by examining their growth rates.

17. An algorithm with the O(n) time complexity is called a linear time algorithm and an algorithm with the O(log n) time complexity is called a logarithmic algorithm.

18. Use the Big O notation to estimate the time complexity of the following method: O(n)

```
public static void mD(int[] m) {
   for (int i = 0; i < m.length; i++) {
      System.out.print(m[i] + " ");
   }
```

19. The worst-case time complexity for selection sort, insertion sort, bubble sort, and quick sort algorithms is $O(n^2)$.

20. Suppose a list is [12, 9, 15, 4, 20, 11]. After the first pass of bubble sort, the list becomes [9,12,4,15,11,20]

21. Indicate the number of possible values that can be put in the '?' place to maintain the max heap 9/10/11/12/13

```
        13
       /  \
    "?"    10
   /  \
8       9
```

22. In a graph, if two vertices are connected by two or more edges, these edges are called parallel edge. A complete graph is the one in which every two pairs of distinct vertices is connected by an edge.

23. There are two popular ways to traverse a graph, namely depth-first search and breath-first-search

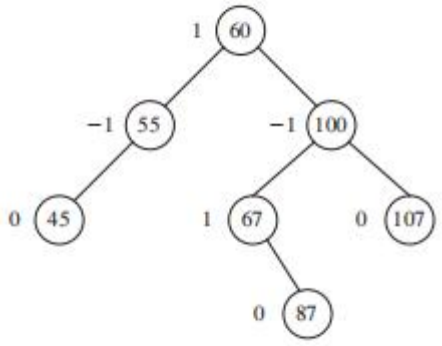
24. In a network of cities connected by roads of varying lengths, if you want to connect all the cities in the most cost-effective way possible, you would use the Prim's algorithm. However, if you want to find the shortest route from one specific city to another, you would use the Dijkstra's algorithm.

25. The time complexity for inserting and deleting an element into a binary search tree are $O(\log n)$ and $O(\log n)$ respectively, assuming that the tree is balanced.

26. In a binary tree, a preorder traversal visits the root node first, followed by the left subtree, and then the right subtree. Conversely, a postorder traversal visits the left subtree first, followed by the right subtree, and finally the root node..

27. In AVL tree, A node is said to be left-heavy if its balance factor is -1, and it is said to be right-heavy if its balance factor is +1.

28. Given the original AVL tree below, after adding element 42, the LL rotation would be performed to rebalance the tree.



29. Open Addressing and Separate Chaining methods are usually taken to deal with hashing collision where two different keys are mapped to the same index in a hash table.

30. In hash tables, linear probing resolves collisions by sequentially checking the next available slots, while quadratic probing resolves collisions by checking slots at increasing squared distances from the original hash.

Coding Questions

Q1. A real estate analyst is comparing the prices of recently sold luxury homes in a prestigious neighborhood. The prices of these homes are as follows: $1,250,000.1, $1,750,000.3, $2,100,000.6, $1,900,000.8, and $2,300,000.2.

Complete the following method that find the maximum element in an ArrayList of generic elements, return null if the ArrayList is empty, and and return the maximum value of the ArrayList that was passed to the function if it is not null.
public static <E extends Comparable<E>> E max(ArrayList<E> list)

Test cases:
ArrayList<Double> prices2 = new ArrayList<>();
System.out.println(maxPrice(prices2)); // -> null

ArrayList<Double> prices1 = new ArrayList<>(Arrays.asList(1250000.1, 1750000.3, 2100000.6, 1900000.8, 2300000.2));
System.out.println(maxPrice(prices1)); // -> 2300000.2

White board (where students code answers):

```
import java.util.ArrayList;
import java.util.Arrays;
```

```
public class RealEstateAnalyzer {

    public static void main(String[] args) {

        // Printing Test case with an empty list

        // Printing Test case with highest prices

    }

    public static <E extends Comparable<E>> E max(ArrayList<E> list) {

        // Empty list case
        return null;


        // Highest price case
        return highestPrice;
    }
}
```

Q2. You are tasked with enhancing a flight boarding system for an airline. The airline categorizes passengers into three classes: "First Class," "Business Class," and "Economy Class." The following three priority queues represent the passengers waiting to board:

First Class Queue: {"Alice", "Bob", "Charlie", "Diana", "Ethan", "Fiona"}
Business Class Queue: {"Charlie", "Diana", "Grace", "Ian"}
Economy Class Queue: {"Alice", "Grace", "Ethan", "Hannah", "Ian"}

You job is to find the exclusive First Class passengers who are only in "First Class" and not in "Business Class" or "Economy Class", by completing the 3 tasks indicated in the following white board.

White board (where students code answers):

```
import java.util.Arrays;
import java.util.HashSet;
import java.util.PriorityQueue;
import java.util.Set;
```

```
public class FlightBoardingSystem {

    public static void main(String[] args) {
        // Task 1: Create PriorityQueues for each class of passengers

        // Task 3: Call findExclusiveFirstClassPassengers() method and print
        out the exclusive First Class passengers

    }


    public static Set<String> findExclusiveFirstClassPassengers(
            PriorityQueue<String> firstClassQueue,
            PriorityQueue<String> businessClassQueue,
            PriorityQueue<String> economyClassQueue) {

       // Task 2: Implement this method to find exclusive First Class
          passengers

        return exclusiveFirstClass;
    }
}
```