Sample Module-4 Questions

Strings

1. Justify String are immutable.
2. Differentiate between String and StringBuffer.
3. Differentiate between StringBuffer and StringBuilder.
4. Differentiate between capacity and length in StringBuffer?
5. Explain compareTo() and equals() method in String.
6. Find the output:

a.
```
class Demo {
public static void main(String args[]) {
        String s="Sugar";
        s.concat("Spoon");
        System.out.println(s);
}
}
```

b.
```
class Demo {
public static void main(String args[]) {
        String s1 = "Sample";
        String s2 = new String(s1);
        System.out.println(s1.equals(s2));
        System.out.println(s1 == s2);
}
```

c.
```
public class Test{
      public static void main(String args[]){
              String x = "Java";
              int y = 8;
              System.out.println(x += y);
       }
```

```
}
```

d. How many string objects will be created for the following code to be executed?

```
String s1 = new String("Java");
String s2 = "Java";
String s3 = "Java";
```

7. Write a java program to check a given string is palindrome or not.
8. Write a java program to capitalize the first letter of each word in an inputted sentence from the keyboard.
9. Write a java program to reverse a string.
10. Write a java program to convert a string into uppercase and lowercase.
11. Write a java program to replace a substring with another string.
12. Write a program that receives two strings and check whether they are equal or not.

Collection Frameworks

1. Differentiate between ArrayList and LinkedList.
2. Differentiate between ArrayList and Vector.
3. What is Iterator, ListIterator interface?
4. Differentiate between List and Set.
5. Differentiate between Set and Map.
6. What are the various interfaces under collection frameworks?
7. Find the output:

```
import java.util.*;
class Test{
   public static void main(String []args){
      PriorityQueue<Integer> pq= new PriorityQueue<Integer>();
      pq.add(10);
      pq.add(20);
      pq.add(1);
```

```
            pq.add(3);
            System.out.println(pq.size());
            System.out.println(pq);
        }
    }
```

8. **Write a Java program to create two linked lists by adding some integers. Merge the two linked lists to form a new linked list. Print the resultant linked list.**

Ans:

```
import java.util.LinkedList;

public class MergeLinkedLists {
    public static void main(String[] args) {
        // Create first linked list and add elements
        LinkedList<Integer> list1 = new LinkedList<>();
        list1.add(10);
        list1.add(20);
        list1.add(30);

        // Create second linked list and add elements
        LinkedList<Integer> list2 = new LinkedList<>();
        list2.add(40);
        list2.add(50);
        list2.add(60);

        // Merge the two linked lists
        LinkedList<Integer> mergedList = new LinkedList<>(list1);
        mergedList.addAll(list2);

        // Print the merged linked list
        System.out.println("Merged Linked List:");
```

```
        for (int num : mergedList) {
            System.out.print(num + " ");
        }
    }
}
```

Output:

```
Merged Linked List:
10 20 30 40 50 60
```

9. Write a Java program to create a linked list by adding names of students. Iterate the linked list in reverse order.

Ans:

```
import java.util.LinkedList;

import java.util.ListIterator;

public class ReverseLinkedList {

   public static void main(String[] args) {

      LinkedList<String> studentList = new LinkedList<>();  // Create a LinkedList of student type

      studentList.add("Amit");                  // Add student names to the list

      studentList.add("Bhairu");

      studentList.add("Chaman");

      studentList.add("Dibba");

      studentList.add("Ellema");

      System.out.println("Original Linked List:");
```

```
        for (String name : studentList) {        // Printing the original list
            System.out.print(name + " ");
        }
        // Reverse iteration using ListIterator
        System.out.println("\n\nLinked List in Reverse Order:");
        ListIterator<String> iterator = studentList.listIterator(studentList.size());
        while (iterator.hasPrevious()) {
            System.out.print(iterator.previous() + " ");
        }
    }
}
```

Output:

Original Linked List:

Amit Bhairu Chaman Dibba Ellema

Linked List in Reverse Order:

Ellema Dibba Chaman Bhairu Amit

10. **Write a program to create an ArrayList that stores 5 integers 100,200,300,400,500. Display the ArrayList using Iterator interface. Perform the following operations on the ArrayList:**
    a. **Add an integer 600 at the end of the list.**
    b. **Retrieve the value present at index 3.**

c. **Update the value of index 2 to 250.**

d. **Check whether the element 200 present in the list**

Ans:

```
import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListOperations {
    public static void main(String[] args) {
        // Step 1: Create an ArrayList and add 5 integers
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(100);
        numbers.add(200);
        numbers.add(300);
        numbers.add(400);
        numbers.add(500);

        // Step 2: Display the ArrayList using Iterator
        System.out.println("Original ArrayList:");
        Iterator<Integer> iterator = numbers.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }

        // Step 3a: Add 600 at the end of the list
        numbers.add(600);
        System.out.println("\n\nAfter adding 600:");
        System.out.println(numbers);

        // Step 3b: Retrieve the value at index 3
        int valueAtIndex3 = numbers.get(3);
        System.out.println("\nValue at index 3:"+ valueAtIndex3);
```

```
        // Step 3c: Update the value at index 2 to 250
        numbers.set(2, 250);
        System.out.println("\nAfter updating index 2 to 250:");
        System.out.println(numbers);

        // Step 3d: Check if 200 is present in the list
        boolean contains200 = numbers.contains(200);
        System.out.println("\nIs 200 present in the list? " + contains200);
    }
}
```

Output:

```
Original ArrayList:
100 200 300 400 500

After adding 600:
[100, 200, 300, 400, 500, 600]

Value at index 3: 400

After updating index 2 to 250:
[100, 200, 250, 400, 500, 600]

Is 200 present in the list? true
```

11. Write a program to create the LinkedList by adding 6 names of the students [Alex, Sipser, Charlie, Smith, Jones, Andrew]. Perform the following operations:

a. Display the LinkedList in forward and backward directions.

b. Add a student name "Rony" at the first node of the linkedlist and display it.

c. Add a student name "John" at the last node of the linkedlist and display the list.

d. Remove the student Sipser from the collection.

Ans:

```
import java.util.LinkedList;
import java.util.ListIterator;
public class StudentLinkedList {
    public static void main(String[] args) {
        // Step 1: Create LinkedList and add 6 student names
        LinkedList<String> students = new LinkedList<>();
        students.add("Alex");
        students.add("Sipser");
        students.add("Charlie");
        students.add("Smith");
        students.add("Jones");
        students.add("Andrew");

        // a. Display the LinkedList in forward direction
        System.out.println("Forward Traversal:");
        for (String name : students) {
            System.out.print(name + " ");
```

```
}
// Display the LinkedList in backward direction
System.out.println("\n\nBackward Traversal:");
ListIterator<String> iterator = students.listIterator(students.size());
while (iterator.hasPrevious()) {
    System.out.print(iterator.previous() + " ");
}
// b. Add "Rony" at the first node
students.addFirst("Rony");
System.out.println("\n\nAfter adding 'Rony' at the beginning:");
System.out.println(students);
// c. Add "John" at the last node
students.addLast("John");
System.out.println("\nAfter adding 'John' at the end:");
System.out.println(students);
// d. Remove "Sipser" from the list
students.remove("Sipser");
System.out.println("\nAfter removing 'Sipser':");
System.out.println(students);
```

```
  }
}
```

Output:

Forward Traversal:

Alex Sipser Charlie Smith Jones Andrew

Backward Traversal:

Andrew Jones Smith Charlie Sipser Alex

After adding 'Rony' at the beginning:

[Rony, Alex, Sipser, Charlie, Smith, Jones, Andrew]

After adding 'John' at the end:

[Rony, Alex, Sipser, Charlie, Smith, Jones, Andrew, John]

After removing 'Sipser':

[Rony, Alex, Charlie, Smith, Jones, Andrew, John]

12. **Write a java program to create an ArrayList with elements: [111, 222, 333, 444, 555] and traverse the list using Iterator. Remove the first and last element of the list and display the final list.**

Ans:

```
import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListExample {
    public static void main(String[] args) {
```

```
        // Step 1: Create and populate the ArrayList
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(111);
        numbers.add(222);
        numbers.add(333);
        numbers.add(444);
        numbers.add(555);

        // Step 2: Traverse using Iterator
        System.out.println("Original ArrayList:");
        Iterator<Integer> iterator = numbers.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }

        // Step 3: Remove first and last elements
        if (!numbers.isEmpty()) {
            numbers.remove(0); // Remove first element
        }
        if (!numbers.isEmpty()) {
            numbers.remove(numbers.size() - 1); // Remove last element
        }

        // Step 4: Display final list
        System.out.println("\n\nFinal ArrayList after removing first and last elements:");
        System.out.println(numbers);
    }
}
```

Output:

```
Original ArrayList:
111 222 333 444 555
```

```
Final ArrayList after removing first and last elements:
[222, 333, 444]
```

13. Write a java program to create a String type Stack. Insert 5 string elements into the stack and traverse the elements using Iterator. Perform 2 deletion operations and display the final list of elements present in the stack.

Ans:

```
import java.util.Stack;
import java.util.Iterator;

public class StringStackDemo {
    public static void main(String[] args) {
        // Step 1: Create a Stack of Strings
        Stack<String> stack = new Stack<>();

        // Step 2: Insert 5 string elements
        stack.push("Apple");
        stack.push("Banana");
        stack.push("Cherry");
        stack.push("Date");
        stack.push("Elderberry");

        // Step 3: Traverse using Iterator
        System.out.println("Original Stack:");
        Iterator<String> iterator = stack.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }

        // Step 4: Perform 2 deletion operations (pop)
```

```
        if (!stack.isEmpty()) stack.pop(); // Removes "Elderberry"
        if (!stack.isEmpty()) stack.pop(); // Removes "Date"

        // Step 5: Display final stack
        System.out.println("\n\nFinal Stack after 2 deletions:");
        System.out.println(stack);
    }
}
```

Output :

```
Original Stack:
Apple Banana Cherry Date Elderberry

Final Stack after 2 deletions:
[Apple, Banana, Cherry]
```

14. Explain PriorityQueue. Write a java program to create a priority queue with five string type elements. Display the head element of the queue and traverse the queue using Iterator.

Ans:

```
import java.util.PriorityQueue;
import java.util.Iterator;

public class StringPriorityQueue {
    public static void main(String[] args) {
        // Step 1: Create a PriorityQueue of Strings
        PriorityQueue<String> queue = new PriorityQueue<>();
```

```
        // Step 2: Add five string elements
        queue.add("Banana");
        queue.add("Apple");
        queue.add("Mango");
        queue.add("Cherry");
        queue.add("Orange");

        // Step 3: Display the head element
        System.out.println("Head of the queue: " + queue.peek());

        // Step 4: Traverse the queue using Iterator
        System.out.println("\nTraversing the queue:");
        Iterator<String> iterator = queue.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
    }
}
```

Output:

```
Head of the queue: Apple

Traversing the queue:
Apple Banana Mango Cherry Orange
```