Problem Statement 1:

1. Small Code Using abstract Keyword

```
abstract class Animal {     abstract void
sound(); // Abstract method


   void eat() { // Non-abstract method
      System.out.println("This animal eats food.");
   }
}
class Dog extends Animal {
   @Override
   void sound() {
      System.out.println("The dog barks.");
   }
}
public class AbstractExample {     public
static void main(String[] args) {
Dog dog = new Dog();        dog.sound();
dog.eat();
   }
}
```

---

2. Abstract Class with Abstract and Non-Abstract

Methods

```
abstract class Shape {     abstract void draw(); //
Abstract method     void description() { // Non-abstract
method
      System.out.println("This is a shape.");
   }
}
```

```
class Circle extends Shape {
   @Override
   void draw() {
      System.out.println("Drawing a Circle.");
   }
}
public class AbstractClassDemo {
   public static void main(String[] args) {
Circle circle = new Circle();
circle.draw();        circle.description();
   }
}
```

---

3. Interface and Abstract

Methods interface Vehicle {

```
void start(); // Abstract method
void stop();  // Abstract method
}
class Car implements Vehicle {
   @Override
   public void start() {
      System.out.println("Car is starting.");
   }
   @Override
   public void stop() {
      System.out.println("Car is stopping.");
   }
}
```

```
public class InterfaceDemo {     public
static void main(String[] args) {        Car
car = new Car();
      car.start();
car.stop();
   }
}
```

---

4. Print Array in Reverse Order

```
public class ReverseArray {     public
static void main(String[] args) {
int[] array = {1, 2, 3, 4, 5};

      System.out.println("Array in reverse order:");
for (int i = array.length - 1; i >= 0; i--) {
         System.out.print(array[i] + " ");
      }
   }
}
```

---

5. Override toString and Sort

Array import.util.Arrays; class Person

```
{    String name;
   int age;
   Person(String name, int age) {
this.name = name;
      this.age = age;
   }
   @Override     public
String toString() { return
```

```
"Person{name='" + name
+ "', age=" + age + "}";
   }
}
public class ToStringSortExample {
public static void main(String[] args) {
Person[] persons = {            new
Person("Alice", 30),            new
Person("Bob", 25),            new
Person("Charlie", 35)
      };
      Arrays.sort(persons, (p1, p2) -> Integer.compare(p1.age,
p2.age));        System.out.println("Sorted array of persons:");        for
(Person person : persons) {
         System.out.println(person);
      }
   }
}
```

---

6. Compare Strings Using Different Methods

```
public class StringComparison {
public static void main(String[] args) {
      String str1 = "Hello";
      String str2 = "Hello";
      String str3 = new String("Hello");
      // By "==" method
      System.out.println("Using '==' method:");
      System.out.println(str1 == str2); // true
      System.out.println(str1 == str3); // false
      // By "equals" method
```

```
        System.out.println("\nUsing 'equals' method:");
        System.out.println(str1.equals(str2)); // true
        System.out.println(str1.equals(str3)); // true
        // By "compareTo" method
        System.out.println("\nUsing 'compareTo' method:");
        System.out.println(str1.compareTo(str2)); // 0
        System.out.println(str1.compareTo("World")); // Negative value
    }
}
```

---

7. Concatenate Strings public class StringConcatenation { public static void main(String[] args) {

```
        String str1 = "Hello";
        String str2 = " World";
        String result = str1 + str2; // Using '+' operator
        System.out.println("Concatenated String: " + result);
    }
}
```

Problem Statement 2:

---

1. Create a File, Write Content, and Read From a File import.io.*; public class FileOperations { public static void main(String[] args) {

```
        String fileName = "example.txt";
        String content = "This is an example content.";
        // Write to a file
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
            writer.write(content);
```

```
            System.out.println("File written successfully.");
} catch (IOException e) {
            e.printStackTrace();
        }
        // Read from a file        try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
            String line;
            System.out.println("Reading file content:");
while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

2. Calculate the Average Value of Array

Elements public class ArrayAverage {

public static void main(String[] args) {

int[] numbers = {10, 20, 30, 40, 50};   int

sum = 0;

```
        for (int num : numbers) {
sum += num;
        }
        double average = (double) sum / numbers.length;
        System.out.println("The average is: " + average);
    }
}
```

---

3. Create and Print an ArrayList

```
import.util.ArrayList; public class
ArrayListExample {  public static void
main(String[] args) {  ArrayList<String>
list = new ArrayList<>();
list.add("Apple");  list.add("Banana");
list.add("Cherry");
  System.out.println("ArrayList elements:");
for (String item : list) {
    System.out.println(item);
  }
 }
}
```

---

4. Generic Class with doSwap() Method

```
public class Swapper {  public static <T>
void doSwap(T a, T b) {
  System.out.println("Before Swap: a = " + a + ", b = " + b);
T temp = a;
  a = b;
b = temp;
  System.out.println("After Swap: a = " + a + ", b = " + b);
 }
 public static void main(String[] args) {
doSwap(10, 20);  doSwap("Hello",
"World");
 }
}
```

---

5. Append Element to a Linked List import

.util.LinkedList; public class

LinkedListAppend { public static void

main(String[] args) { LinkedList<String>

list = new LinkedList<>();

list.add(" ");

list.add("Python");

System.out.println("Original LinkedList: " + list);

list.addLast("C++"); // Appending an element

System.out.println("Updated LinkedList: " + list);

}

}

---

6. Count Key-Value Mappings in a Map

import .util.HashMap; public class

MapSize { public static void

main(String[] args) {

HashMap<String, Integer> map = new HashMap<>();

map.put("Apple", 3);

map.put("Banana", 5);

map.put("Cherry", 2);

System.out.println("Number of key-value pairs: " + map.size());

}

}

---

7. Clone a TreeSet

```
import .util.TreeSet; public class
TreeSetClone {     public static void
main(String[] args) {
      TreeSet<String> originalSet = new
TreeSet<>();        originalSet.add("A");
originalSet.add("B");
      originalSet.add("C");
      System.out.println("Original TreeSet: " + originalSet);
      TreeSet<String> clonedSet = (TreeSet<String>) originalSet.clone();
      System.out.println("Cloned TreeSet: " + clonedSet);
   }
}
```