Tutorial/Lab 3 - Abstract Classes and Interfaces

Aim

This tutorial/lab aims to further the understanding of the topics covered in Week 3's lecture, such as Abstract classes, Interface and deep copy.

Exercise 3.1 ArrayList Shuffling

Write the following method that shuffles an ArrayList of numbers:

public static void shuffle(ArrayList<Number> list)

Hints:
1. Create the ArrayList first
2. In the shuffle() method, you may consider the Fisher-Yates shuffle algorithm, which is efficient for shuffling.
3. Generate a random index within the range of the list size using Math.random().
4. Swap the element at index 'i' with the element at the randomly generated index to perform the shuffle.

Exercise 3.2 Comparable Interface

Define a class named ComparableCircle that extends Circle and implements Comparable. Implement the compareTo method to compare the circles on the basis of area. Write a test class to find the larger of two instances of ComparableCircle objects

Hints:
1. Define a class named ComparableCircle that extends Circle and implements Comparable interface. You need to implement the compareTo method to compare the circles based on their radii.
2. Implement the compareTo method in the ComparableCircle class to compare circles based on their radii. You should compare the radii of two circles and return a positive integer if the radius of the current circle is greater, a negative integer if it's smaller, and zero if they are equal.
3. Write a test class to verify the functionality of the ComparableCircle class. Create two ComparableCircle objects and compare their sizes using the max method.

Exercise 3.3 Deep Copy

Rewrite the MyStack class in Lecture 2.2 (page 43) to perform a deep copy of the list field.

Hints:
1. Implement the Cloneable interface in the MyStack class to enable cloning functionality.
2. Override the clone method in the MyStack class to create a deep copy of the list field.

3. Within the clone method, clone each object in the list to ensure a deep copy is made.

Exercise 3.4 Abstract class vs. Interface

1. Consider the following code snippet. Will it compile successfully? If not, what is the reason?

```
1  public interface Animal {
2      String name; // Data field representing the name of the animal
3      void makeSound(); // Abstract method to make the animal sound
4  }
```

2. Consider the following code snippet. Will it compile successfully? If not, what is the reason?

```
1  public interface MyInterface2 {
2      void method1(); // Abstract method without implementation
3      void method2(); // Abstract method without implementation
4      void method3() {
5          // Concrete method with implementation
6          System.out.println("Method 3 implementation");
7      }
8  }
9
```

3. What is the issue in the following code snippet. How it should be solved?

```
1  public abstract class MyClass {
2      public MyClass() {
3          System.out.println("Abstrac class constructor");
4      }
5
6      public static void main(String[] args) {
7          MyClass obj = new MyClass();
8      }
9  }
10
```