Homework 3: Abstract Data Type (ADT) and List ADT

Bài 1. Tạo kiểu dữ liệu complex number như sau:

```
//Tạo đối tượng Complex
public class Complex {

	private float real;  //phần thực
	private float image; //phần ảo

	public Complex(float r, float i) {
		// Hàm khởi tạo
	}

	public Complex add(Complex c) {
		// Hoàn thành hàm cộng số phức hiện tại với số phức c
	}

	public Complex minus(Complex c) {
		// Hoàn thành hàm trừ số phức hiện tại cho số phức c
	}

	public Complex time(Complex c) {
		// Hoàn thành hàm nhân số phức hiện tại với số phức c
	}

	public float realpart() {
		return real;
	}

	public float imagepart() {
		return image;
	}

	@Override
	public String toString() {
		// Hoàn thành hàm in ra số phức dạng a + bi
	}
}
```

Sử dụng kiểu dữ liệu comlex number để:

- Nhập dãy *n* số phức.
- In ra số phức ở vị trí thứ *v* trong dãy.
- Tính tổng của *n* số phức.
- (*) Tự đề xuất một ứng dụng sử dụng kiểu số phức đã tạo ở trên.

Bài 2. Tạo kiểu dữ liệu Shape

```
//Tạo đối tượng Shape
public abstract class Shape {

	protected String name;

	protected abstract double getVolume();
	protected abstract double getArea();
	protected abstract double getPerimeter();

	protected void setName(String name) {
		this.name = name;
	}
```

```
    protected String getName() {
        return this.name;
    }
}
```

Sử dụng kiểu dữ liệu Sphere để tạo các kiểu dữ liệu Circle, Sphere, Rectangle thừa kế kiểu dữ liệu Sphere như sau

```
//Tạo đối tượng Circle thừa kế Shape
public class Circle extends Shape {

    private double radius = 0;

    public Circle(double radius) {
        // Hàm khởi tạo
    }

    @Override
    protected double getVolume() {
        return 0;
    }

    @Override
    protected double getArea() {
        // Hoàn thành thân hàm
    }

    @Override
    protected double getPerimeter() {
        // Hoàn thành thân hàm
    }
}


 //Tạo đối tượng Sphere thừa kế Shape
public class Sphere extends Shape {

    private double radius = 0;

    public Sphere(double radius) {
        // Hàm khởi tạo
    }

    @Override
    protected double getVolume() {
        // Hoàn thành thân hàm tính thể tích
    }

    @Override
    protected double getArea() {
        // Hoàn thành thân hàm tính diện tích xung quanh
    }

    @Override
    protected double getPerimeter() {
        // Hoàn thành thân hàm lấy bán kính
    }
}


 //Tạo đối tượng Rectangle thừa kế Shape
public class Rectangle extends Shape {

    private double width = 0;
    private double height = 0;
```

```
        public Rectangle(double width, double height) {
            // Hàm khởi tạo
        }

        @Override
        protected double getVolume() {
            // Hoàn thành thân hàm
        }

        @Override
        protected double getArea() {
            // Hoàn thành thân hàm
        }

        @Override
        protected double getPerimeter() {
            // Hoàn thành thân hàm
        }
    }
```

Sử dụng các kiểu dữ liệu đã tạo ở trên để:

- Tạo một danh sách các hình khối.
- In ra danh sách gồm tên các hình khối và với diện tích kèm theo, nếu là Sphere thì in ra thể tích, còn lại Circle và Rectangle thì in ra chu vi.
- (*) Tự đề xuất một ứng dụng sử dụng các kiểu dữ liệu đã tạo.

Bài 3. Tạo danh sách (List) bằng kiểu dữ liệu mảng (array) như sau:

```
//Tạo giao diện ListInterface kế thừa giao diện Iterable
  public interface ListInterface<T> extends Iterable<T>{
        public void add(T data);  //Thêm phần tử vào danh sách
        public T get(int i);      //Lấy giá trị phần tử thứ i
        public void set(int i, T data);  //Đặt data vào vị trí i của danh sách
        public void remove(T data);    //Loại phần tử data khỏi danh sách
        public void isContain(T data); //Kiểm tra phần tử data có trong danh sách
        public int size();             //Kích thước danh sách
        public boolean isEmpty();      //Danh sách có rỗng hay không
  }


//Tạo lớp SimpleArrayList cài đặt giao diện ListInterface
  public class SimpleArrayList<T> implements ListInterface<T> {
        private T[] array;
        private int n = 0;
        private int defaultSize = 100;

        public SimpleArrayList() {
            array = (T[]) new Object[defaultSize];
        }
        public SimpleArrayList(int capacity) {
            // Hàm dựng với kích thước mảng là capacity
        }
        public void add(T data) {
            // Hoàn thành thân hàm}
        public T get(int i) {
```

```
            // Hoàn thành thân hàm
        }
        public void set(int i, T data) {
            // Hoàn thành thân hàm
        }
        public void remove(T data) {
            // Hoàn thành thân hàm
        }
        public boolean isContain(T data) {
            // Hoàn thành thân hàm
        }
        public int size() {
            // Hoàn thành thân hàm
        }
        public boolean isEmpty() {
            // Hoàn thành thân hàm
        }
        public Iterator<T> iterator() {
            //Trả về toàn bộ danh sách
            // Hoàn thành thân hàm
        }
    }
```

Bài 4. Tạo kiểu danh sách móc nối (SimpleLinkedList) theo mẫu sau.

```
public class SimpleLinkedList<T> {
    class Node {
        T data;
        Node next;
    }
    private Node top = null;
    private Node bot = null;
    private int n = 0;
    public void add(T data) {
        // Thêm phần tử vào đầu danh sách
    }
    public void addBot(T data) {
        // Thêm phần tử vào cuối danh sách
    }
    public T get(int i) {
        // Lấy phần tử ở vị trí thứ i
        return null;
    }
    public void set(int i, T data){
        // Gán giá trị ở vị trí i bằng data
    }
    public boolean isContain(T data) {
        // Kiểm tra trong danh sách có chứa phần tử data hay không?
        return false;
    }
    public int size() {
        // Trả lại thông tin số phần tử có trong danh sách
        return 0;
    }
    public boolean isEmpty() {
        // Kiểm tra danh sách có rỗng hay không?
        return true;
    }
    public T removeTop() {
```

```
            // Xóa phần tử ở đầu danh sách, trả lại giá trị data của phần tử đó
        return null;
        }
        public T removeBot() {
            // Xóa phần tử ở cuối danh sách, trả lại giá trị data của phần tử đó
        return null;
        }
        public void remove(T data) {
            // Xóa tất cả các phần tử có giá trị bằng data
        }
    }
```

Bài 5. (*) Sử dụng các cấu trúc dữ liệu ở bài 3, 4 để viết chương trình đếm số lần xuất hiện của các từ trong một văn bản cho trước nhập vào từ bàn phím hoặc từ file văn bản.

Tạo đối tượng là WordCount gồm 2 thuộc tính là word và count. Đối tượng WordCount nạp chồng phương thức equals(Object o) để có thể sử dụng phương thức isContain đã xây dựng ở các cấu trúc dữ liệu trên, hoặc có thể sử dụng phương thức indexOf của các đối tượng cài đặt giao diện List.
