Interface

Mục tiêu

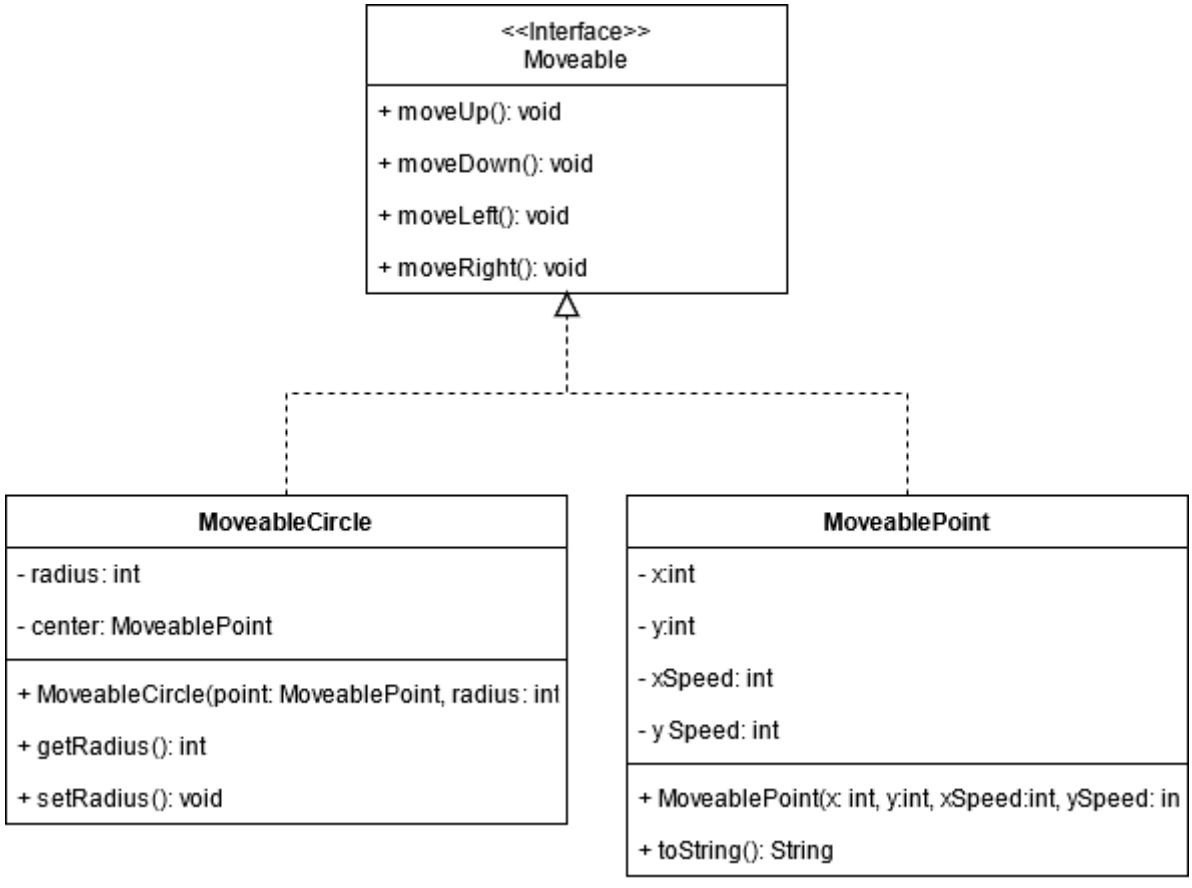
Kết thúc bài thực hành này, sinh viên có khả năng:

- ✓ Biết cách xác định và định nghĩa các hành vi phổ quát (giống nhau) của các đối tượng bằng cách sử dụng Interface
- ✓ Biết cách định nghĩa Interface và định nghĩa các lớp hiện thực Interface
- ✓ Biết cách xác định được kiểu dữ liệu của đối tượng ở thời điểm thực thi
- ✓ Phân biệt được sự khác nhau giữa Abstract Class và Interface và biết cách vận dụng vào những tình huống cụ thể
- ✓ Biết cách vận dụng khả năng đa thừa kế trong Interface

Dùng phương pháp lập trình hướng đối tượng, các em hãy giải quyết các bài tập sau đây:

Bài 1: (5 điểm)

Hãy xây dựng interface Moveable và các lớp hiện thực interface này là MoveablePoint và MoveableCircle theo sơ đồ cây thừa kế như sau:



Mô tả:

❖ Interface Moveable
- Thể hiện khả năng di chuyển của một lớp đối tượng
- Chứa các phương thức thuần ảo: moveUp, moveDown, moveLeft, moveRight có nghĩa là di chuyển một đối tượng đi lên, đi xuống, sang trái, và sang phải

❖ Class MoveableCircle
- Biểu diễn các đối tượng hình tròn có thể di chuyển được
- Đặc trưng bởi các thuộc tính riêng: radius (bán kính), center (tâm hình tròn). Trong đó center là một đối tượng thuộc kiểu MoveablePoint, thể hiện mối quan hệ has-a (một MoveableCircle có tâm là một MoveablePoint)
- Các phương thức riêng của lớp:
  - MoveableCircle(): khởi tạo giá trị cho các thuộc tính
  - getRadius(): phương thức getter của thuộc tính radius
  - setRadius(): phương thức setter của thuộc tính radius

❖ Class MoveablePoint
- Biểu diễn các đối tượng điểm có thể di chuyển được
- Đặc trưng bởi các thuộc tính: x, y, xSpeed (khoảng cách theo phương ngang mà mỗi lần điểm có thể di chuyển được), ySpeed (khoảng cách theo phương đứng mà mỗi lần điểm có thể di chuyển được)
- Các phương thức riêng của lớp:
  - MoveablePoint(): khởi tạo giá trị cho các thuộc tính
  - toString(): thể hiện thông tin về đối tượng theo dạng MoveablePoint[x = ?,y = ?, xSpeed = ?, ySpeed = ?]

Bài làm:

```
package LabEx7;

public class Bai1 {
	public static void main(String[] args) {
		MoveablePoint point = new MoveablePoint(3, -5, 2, 4);
		MoveableCircle circle = new MoveableCircle(point, 4);

		// Thực hiện tùy ý một số bước di chuyển tọa độ của hình
tròn
		circle.moveUp();
		circle.moveLeft();
		circle.moveLeft();
```

```
            circle.moveRight();
            circle.moveDown();
            circle.moveUp();
      }
}

interface Moveable {

      public void moveUp();

      public void moveDown();

      public void moveLeft();

      public void moveRight();
}

class MoveablePoint implements Moveable {
      // Thành phần dữ liệu
      private int x;
      private int y;

      private int xSpeed;
      private int ySpeed;

      // Thành phần xử lý

      public MoveablePoint(int x, int y, int xSpeed, int ySpeed) {
            this.x = x;
            this.y = y;
            this.xSpeed = xSpeed;
            this.ySpeed = ySpeed;
      }

      @Override
      public void moveUp() {
            y = y + ySpeed;
      }

      @Override
      public void moveDown() {
            y = y - ySpeed;
      }

      @Override
      public void moveLeft() {
            x = x - xSpeed;
      }
```

```
	@Override
	public void moveRight() {
		x = x + xSpeed;
	}

	public String toString() {
		return "MoveablePoint[x = "+this.x+",y = "+this.y+", xSpeed
= "+this.xSpeed+", ySpeed = "+this.ySpeed+"]";
	}
}


class MoveableCircle implements Moveable {

	private int radius;
	private MoveablePoint center;


	public int getRadius() {
		return radius;
	}

	public void setRadius(int radius) {
		this.radius = radius;
	}


	public MoveableCircle(MoveablePoint point, int radius) {
		this.center = point;
		this.radius = radius;
	}

	@Override
	public void moveUp() {
		this.center.moveUp();

	}

	@Override
	public void moveDown() {
		this.center.moveDown();

	}

	@Override
	public void moveLeft() {
		this.center.moveLeft();

	}
```

```
	@Override
	public void moveRight() {
		this.center.moveRight();
	}
}
```

Bài 2: (5 điểm)

Xây dựng một lớp Octagon (hình bát giác) kế thừa từ lớp GeometricObject (hình đa giác) được cho như bên dưới và hiện thực các interface Comparable và Cloneable do Giảng viên cung cấp.

Mô tả lớp trừu tượng GeometricObject:

GeometricObject
color: String # filled: boolean # dateCreated: java.util.Date
+ GeometricObject() + GeometricObject(color: String, filled: boolean) + getColor(): String + setColor(color: String): void + isFilled(): boolean + setFilled(filled: boolean): void + getDateCreated(): java.util.Date + toString(): String (GeometricObject[color=red, filled = true, dateCreated = “09/05/2020 7:36:47”]) + getArea(): double + getPerimeter(): double

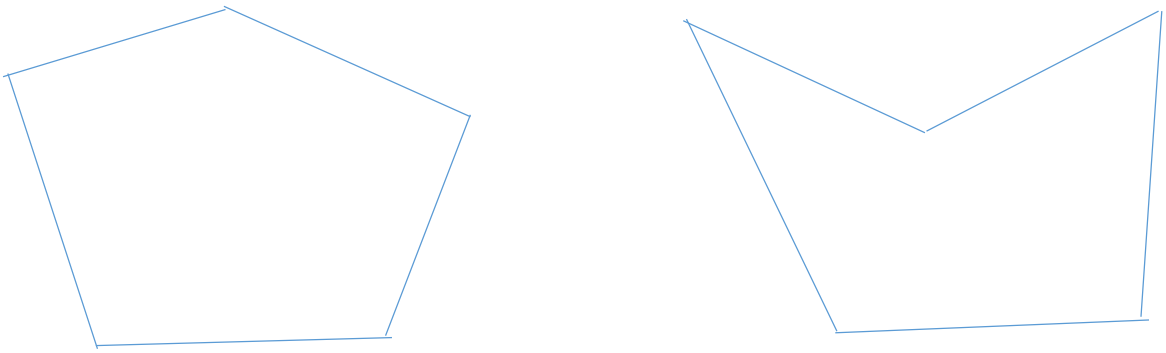
Giả sử rằng 8 cạnh của hình bát giác đều bằng nhau, khi đó diện tích của hình được tính theo công thức:

$$\text{Diện tích} = (2 + 4\sqrt{2}) * \text{cạnh} * \text{cạnh}$$

Sau khi xây dựng các lớp Octagon và GeometricObject hiện thực các interface Comparable và Cloneable nói trên, hãy tạo ra trong chương trình một đối tượng Octagon với độ dài cạnh

bằng 5, tính chu vi và diện tích của hình. Sau đó, tạo ra một đối tượng Octagon mới bằng cách sử dụng phương thức clone() và so sánh hai đối tượng này bằng phương thức compareTo()

Ví dụ: hai hình ngũ giác có độ dài cạnh bằng nhau nhưng là hai hình khác nhau



Bài làm:

```
package LabEx7;
import java.util.Date;

public class Bai2 {
	public static void main(String[] args) {
		Octagon octagon1 = new Octagon(5);
		System.out.println("Chu vi của hình: " +
octagon1.getPerimeter());
		System.out.println("Diện tích của hình: " +
octagon1.getArea());

		// Tạo ra hình octagon2 là nhân bản của hình 1
		Octagon octagon2 = octagon1.clone();

		// So sánh 2 hình
		if (octagon1.compareTo(octagon2) == 0)
			System.out.println("Octagon1 == Octagon2");
		else if (octagon1.compareTo(octagon2) == 1)
			System.out.println("Octagon1 > Octagon2");
		else System.out.println("Octagon1 < Octagon2");
	}
}

interface Cloneable<T> {
	// Tham số: không có tham số truyền vào
```

```
	// Kết quả trả về: Một đối tượng T mới có nội dung giống nội dung
của đối tượng được
	// nhân bản nhưng khác tham chiếu
	public T clone();
}

interface Comparable<T> {
	// Tham số: o - đối tượng được so sánh
	// Kết quả trả về: 0 - bằng nhau, -1 - nhỏ hơn, 1 - lớn hơn
	public abstract int compareTo(T o);
}


abstract class GeometricObject{
	// Thành phần dữ liệu
	protected String color;
	protected boolean filled;
	protected Date dateCreated; // Thuộc tính kiểu tham chiếu/kiểu
đối tượng

	// Thành phần xử lý
	public GeometricObject() {
		this.color = "";
		this.filled = false;
		this.dateCreated = new Date();
	}

	public GeometricObject(String color, boolean filled) {
		this.color = color;
		this.filled = filled;
		this.dateCreated = new Date();
	}

	public String getColor() {
		return this.color;
	}

	public void setColor(String color) {
		this.color = color;
	}


	public boolean isFilled() { // getter
		return this.filled;
	}

	public void setFilled(boolean filled) {
		this.filled = filled;
	}
```

```
	public Date getDateCreated() { // getter
		return this.dateCreated;
	}

	public String toString() {
		return "GeometricObject[color="+this.color+",
filled="+this.filled+", created date="+this.dateCreated+"]";
	}

	abstract double getArea();
	abstract double getPerimeter();
}



class Octagon extends GeometricObject implements Comparable<Octagon>,
Cloneable<Octagon> {
	private double side;

	public Octagon(double side) {
		this.side = side;
	}

	@Override
	public double getArea() {
		// Tính diện tích hình bát giác
		return (2 + 4 * Math.sqrt(2)) * side * side;
	}

	@Override
	public double getPerimeter() {
		// Tính chu vi hình bát giác
		return side * 8;
	}

	@Override
	public int compareTo(Octagon o) {
		// Nhỏ hơn: return -1
		// Bằng nhau: return 0
		// Lớn hơn: return 1

		// So sánh chu vi
		if (this.getPerimeter() > o.getPerimeter()) return 1;
		else if (this.getPerimeter() < o.getPerimeter()) return -1;
		else { // Nếu chu vi 2 hình bằng nhau thì so sánh diện tích
			// So sánh diện tích
			if (this.getArea() > o.getArea()) return 1;
			else if (this.getArea() < o.getArea()) return -1;
```

```
                else return 0;
            }
        }

        @Override
        public Octagon clone() {
            // Nhân bản nghĩa là tạo ra một đối tượng hình bát giác mới
có độ dài cạnh = độ dài cạnh của hình bát giác đang xét
            Octagon octagon = new Octagon(this.side);
            return octagon;
        }

}
```