**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

ЛАБОРАТОРНАЯ РАБОТА № 6

**Дисциплина:** Технология кроссплатформенного программирования

**Тема:** Абстрактные классы и интерфейсы

**Выполнил(а): студент(ка) группы**

Хайруллина Алина Ринатовна

(Фамилия И.О.)

**Дата, подпись**

(Дата) (Подпись)

**Проверил:**

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись**

(Дата) (Подпись)

**Замечания:**

**Москва 2023**

**ЛАБОРАТОРНАЯ РАБОТА № 6**

**Применение абстрактных классов.**

Создать не менее трёх классов, которые будут описывать различные геометрические фигуры предварительно согласовав их с преподавателем (например, треугольник, квадрат и круг).

Предусмотреть наличие у фигур таких характеристик, как цвет, линейные размеры, площадь, возможно объём.

В программе должен быть абстрактный класс с названием Figure, на основе которого создаются производные классы фигур.

В главном методе реализовать создание объектов производных классов и продемонстрировать, как получить доступ к объектам через объектные переменные подклассов и через объектную переменную абстрактного суперкласса.

**Применение интерфейсов.**

Для фигур из предыдущего задания создать интерфейсы.

Создать суперкласс, обеспечивающий ввод с клавиатуры вида арифметической операции (например, умножение) и вывод заголовка результата, который будет отображаться на экране.

Создать подкласс, который наследует суперкласс и реализует интерфейсы.

В подклассе предусмотреть вычисление результата, получаемого применением вводимой арифметической операции к величинам площадей фигур, и отображение результата на экране после отображения заголовка результата.

Листинг

import java.util.HashMap;

abstract class Figure {

// Dictionary of colors and codes

public static final HashMap<String, String> colors = new HashMap<>(); static {

colors.put("Default", "\u001B[0m"); colors.put("Black", "\u001B[40m");

colors.put("Red", "\u001B[41m");

colors.put("Green", "\u001B[42m"); colors.put("Yellow", "\u001B[43m"); colors.put("Blue", "\u001B[44m"); colors.put("Purple", "\u001B[45m"); colors.put("Cyan", "\u001B[46m");

colors.put("White", "\u001B[47m");

};

public final double PI = 3.14159265359; public String name = "Figure";

abstract double area();

public static String colorToCodeStyle(String color) { if (colors.get(color) == null)

return colors.get("Default"); return colors.get(color);

}

}

/\*

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.figure;

/\*\*

* @author 1

\*/

class Cone extends Figure implements Color, Print { private double r;

private double h; private String color;

Cone(double r, double h) { this.name = "Cone"; this.r = r;

this.h = h;

this.color = "Purple";

}

Cone(double r, double h, String color) { this.name = "Cone";

this.r = r; this.h = h;

this.color = color;

}

double base() { return 2 \* PI \* r;

}

double area() { return PI \* r \* h;

}

double volume() {

return (PI \* r \* r \* h) / 3;

}

public String getColor() { return color;

}

void setR(double r) { this.r = r;

}

void setH(double h) { this.h = h;

}

public void print() {

System.out.println("Area of Cone is " + area()); System.out.println("Volume of Cone is " + volume()); System.out.println("Base of Cone is " + base()); System.out

.println("Color of " + Figure.colorToCodeStyle(getColor()) + this.name + Figure.colors.get("Default")

+ " is "

+ getColor()); System.out.println();

}

}

/\*

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.figure;

class Sphere extends Figure implements Color, Print { private double r;

private String color;Sphere(double r) { this.name = "Sphere"; this.r = r;

this.color = "Red";

}

Sphere(double r, String color) { this.name = "Sphere";

this.r = r; this.color = color;

}

double diameter() { return 2 \* r;

}

double area() {

return 4 \* PI \* r \* r;

}

double volume() {

return (4 \* PI \* r \* r \* r) / 3;

}

public String getColor() { return color;

}

void setR(double r) { this.r = r;

}

public void print() {

System.out.println("Area of Sphere is " + area()); System.out.println("Volume of Sphere is " + volume()); System.out.println("Diameter of Sphere is " + diameter()); System.out.println(

"Color of " + Figure.colorToCodeStyle(getColor()) + this.name + Figure.colors.get("Default") + " is "

+ getColor()); System.out.println();

}

}

/\*

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.figure;

/\*\*

* @author 1

\*/

class Square extends Figure implements Color, Print { private double a;

private String color;

Square(double a) { this.name = "Square"; this.a = a;

this.color = "Green";

}

Square(double a, double b, String color) { this.name = "Square";

this.a = a; this.color = color;

}

double area() { return a \* a;

}

double perimeter() { return 4 \* a;

}

public String getColor() { return color;

}

void setA(double a) { this.a = a;

}

public void print() {

System.out.println("Area of Square is " + area());

System.out.println("Perimeter of Square is " + perimeter()); System.out.println(

"Color of " + Figure.colorToCodeStyle(getColor()) + this.name + Figure.colors.get("Default") + " is "

+ getColor()); System.out.println();

}

}

/\*

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.figure;

/\*\*

* @author 1

\*/

class SubClass extends SuperClass implements Print {

private Figure a; private Figure b;

// set figures

public void setFigures(Figure a, Figure b) { this.a = a;

this.b = b;

}

@Override

public void print() { System.out.println("Figure a: " + a.area()); System.out.println("Figure b: " + b.area()); setOperator();

switch (getOperaion()) {

case '+' -> System.out.println("Answer: " + (a.area() + b.area()));

case '-' -> System.out.println("Answer: " + (a.area() - b.area()));

case '/' -> System.out.println("Answer: " + (a.area() / b.area()));

case '\*' -> System.out.println("Answer: " + (a.area() \* b.area())); default -> System.out.println("Wrong operator");

}

}

}

package com.mycompany.figure; import java.util.Scanner;

class SuperClass { private char operator;

public void setOperator() {

Scanner sc = new Scanner(System.in); System.out.println("Enter operator (+, -, / or \*): ");

this.operator = sc.nextLine().charAt(0); System.out.println("Result: ");

sc.close();

}

public char getOperaion() { return this.operator;

}

}

package com.mycompany.figure;

interface Print { void print();

}

package com.mycompany.figure;

interface Color { String getColor();

}

package com.mycompany.figure;

class test {

public static void main(String[] args) {

Figure f;

f = new Sphere(5, "Black"); ((Sphere) f).setR(2);

((Print) f).print();

f = new Cone(7, 4, "Purple"); ((Cone) f).setR(5);

((Cone) f).setH(8);

((Print) f).print();

f = new Square(3, 9, "Red");

((Square) f).setA(3);

((Print) f).print();

SubClass s;

s = new SubClass();

s.setFigures(new Square(8), new Cone(4, 3));

s.print();

}

}

