OBJECT ORIENTED PROGRAMMING

Lab 9: Advanced topics (2)

### Exercise 1:

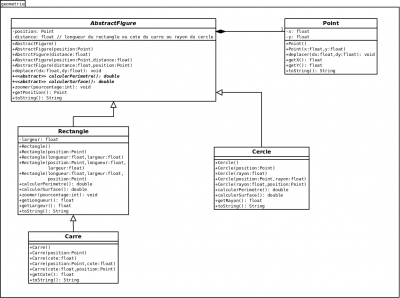
Create an abstract class " AbstractFigure " allowing to pool the elements common to any geometric figure (disc, square, etc.). In particular, you will provide for the 2 abstract methods “calculateSurface ()” and “calculatePerimeter ()”.

Define the UML model of this set of classes.

Add the class "Rectangle" (also derived from " AbstractFigure ") but more generic than the specialization class "Square".

Write and test the AbstractFigure, Disk, Rectangle, Square classes

Answers

****

**File2:**

**package lab9.geometry;**

**public abstract class GeometricFigure {**

**private Point initialPoint;**

**private int distance;**

**public GeometricFigure(Point point, int distance) {**

**this.initialPoint = point;**

**this.distance = distance;**

**}**

**public abstract double calculateSurface();**

**public abstract double calculatePerimeter();**

**public Point getPoint() {**

**return initialPoint;**

**}**

**public int getDistance() {**

**return distance;**

**}**

**@Override**

**public String toString() {**

**return "GeometricFigure [point=" + initialPoint + ", distance=" + distance + "]";**

**}**

**}**

### Exercise 3:

We consider a Drawing class composed of several geometric figures. The figures can be disks, squares or rectangles. We will not consider the case of figures which are superimposed. One of the methods of the “Drawing” class will return the sum of the surfaces of the different figures that compose it. Define the UML class diagram for this problem.

Write the classes "Drawing" and "TestDessin".

Answers

**File1:**

**package lab9.geometry;**

**public class Disk extends GeometricFigure{**

**public Disk(Point point, int distance) {**

**super(point, distance);**

**}**

**@Override**

**public double calculateSurface() {**

**return Math.PI \* getDistance() \* getDistance();**

**}**

**@Override**

**public double calculatePerimeter() {**

**return 2 \* Math.PI \* getDistance();**

**}**

**}**

**File2:**

**package lab9.geometry;**

**public abstract class GeometricFigure {**

**private Point initialPoint;**

**private int distance;**

**public GeometricFigure(Point point, int distance) {**

**this.initialPoint = point;**

**this.distance = distance;**

**}**

**public abstract double calculateSurface();**

**public abstract double calculatePerimeter();**

**public Point getPoint() {**

**return initialPoint;**

**}**

**public int getDistance() {**

**return distance;**

**}**

**@Override**

**public String toString() {**

**return "GeometricFigure [point=" + initialPoint + ", distance=" + distance + "]";**

**}**

**}**

**File3:**

**package lab9.geometry;**

**public class Picture {**

**private GeometricFigure[] figures;**

**private int size;**

**private int currentCount;**

**public Picture(int size) {**

**this.size = size;**

**figures = new GeometricFigure[size];**

**currentCount = 0;**

**}**

**public void addFigure(GeometricFigure figure) {**

**if (currentCount < size) {**

**figures[currentCount] = figure;**

**currentCount++;**

**}**

**}**

**public double sumSurface() {**

**double result = 0.0f;**

**for (int index = 0; index < currentCount; index++) {**

**result += figures[index].calculateSurface();**

**}**

**return result;**

**}**

**}**

**File4:**

**package lab9.geometry;**

**public class Point {**

**private int x;**

**private int y;**

**public Point(int x, int y) {**

**this.x = x;**

**this.y = y;**

**}**

**public Point() {**

**x = 0;**

**y = 0;**

**}**

**public String toString() {**

**return "Point [x=" + x + ", y=" + y + "]";**

**}**

**public int getX() {**

**return x;**

**}**

**public void setX(int x) {**

**this.x = x;**

**}**

**public int getY() {**

**return y;**

**}**

**public void setY(int y) {**

**this.y = y;**

**}**

**}**

**File5:**

**package lab9.geometry;**

**public class Rectangle extends GeometricFigure {**

**private int height;**

**public Rectangle(Point point, int distance, int height) {**

**super(point, distance);**

**this.height = height;**

**}**

**@Override**

**public double calculateSurface() {**

**return getDistance() \* height;**

**}**

**@Override**

**public double calculatePerimeter() {**

**return 2 \* (getDistance() + height);**

**}**

**}**

**File6:**

**package lab9.geometry;**

**/\*\***

**\* We don't need to implement the two calculations again because it works in the**

**\* same way for a square.**

**\***

**\*/**

**public class Square extends Rectangle {**

**public Square(Point point, int distance) {**

**super(point, distance, distance);**

**}**

**}**

**File7:**

**package lab9.geometry;**

**public class TestGeometricFigure {**

**public static void main(String[] args) {**

**Point point = new Point(1,2);**

**GeometricFigure disk = new Disk(point, 5);**

**System.out.println("Disk surface : " + disk.calculateSurface());**

**System.out.println("Disk perimeter : " + disk.calculatePerimeter());**

**GeometricFigure square = new Square(point, 5);**

**System.out.println("Square surface : " + square.calculateSurface());**

**System.out.println("Square perimeter : " + square.calculatePerimeter());**

**GeometricFigure rectangle = new Rectangle(point, 5, 6);**

**System.out.println("Rectangle surface : " + rectangle.calculateSurface());**

**System.out.println("Rectangle perimeter : " + rectangle.calculatePerimeter());**

**}**

**}**

**File7:**

**package lab9.geometry;**

**public class TestPicture {**

**public static void main(String[] args) {**

**Point point1 = new Point(1, 2);**

**GeometricFigure disk = new Disk(point1, 5);**

**Point point2 = new Point(3, 4);**

**GeometricFigure square = new Square(point2, 5);**

**Point point3 = new Point(5, 6);**

**GeometricFigure rectangle = new Rectangle(point3, 5, 6);**

**Picture picture = new Picture(5);**

**picture.addFigure(disk);**

**picture.addFigure(square);**

**picture.addFigure(rectangle);**

**System.out.println("Sum of surface : " + picture.sumSurface());**

**}**

**}**

### Exercise 2:

Define a "Counting" interface, having the 2 functional methods "increment ()" and "decrementert ()" common to the different counters.

Define the UML class diagram for this problem.

Write the "Counting" interface and modify the Counter, BoundedCounter, Cyclic Counter and Stopwatch (chronometer) classes already carried out, then Test.

Answers

**package lab9.chronometer;**

**public interface Counting {**

**void increment();**

**void decrement();**

**int getValue();**

**}**

**package lab9.chronometer;**

**public class Counter implements Counting{**

**/\*\***

**\* The current value of the counter should always be positive.**

**\*/**

**private int value;**

**public Counter(int value) {**

**this.value = value;**

**}**

**@Override**

**public String toString() {**

**return "Counter [value=" + value + "]";**

**}**

**public int getValue() {**

**return value;**

**}**

**protected void setValue(int value) {**

**this.value = value;**

**}**

**public void increment() {**

**value++;**

**}**

**public void decrement() {**

**if (value > 0) {**

**value--;**

**}**

**}**

**public void init() {**

**value = 0;**

**}**

**}**

**package lab9.chronometer;**

**public class BoundedCounter extends Counter implements Counting {**

**private int max;**

**public BoundedCounter(int value, int max) {**

**super(value);**

**this.max = max;**

**}**

**@Override**

**public void increment() {**

**if (getValue() < max) {**

**super.increment();**

**}**

**}**

**@Override**

**public String toString() {**

**return "BoundedCounter [value=" + getValue() + "]";**

**}**

**protected int getMax() {**

**return max;**

**}**

**}**

**package lab9.chronometer;**

**public class CyclicCounter extends BoundedCounter implements Counting{**

**public CyclicCounter(int value, int max) {**

**super(value, max);**

**}**

**@Override**

**public void increment() {**

**if (getValue() < getMax()) {**

**super.increment();**

**} else {**

**setValue(0);**

**}**

**}**

**@Override**

**public void decrement() {**

**if (getValue() > 0) {**

**super.decrement();**

**} else {**

**setValue(getMax());**

**}**

**}**

**@Override**

**public String toString() {**

**return "CyclicCounter [value=" + getValue() + "]";**

**}**

**}**

**package lab9.****chronometer;**

**public class Chromometer {**

**private Counting hour;**

**private Counting minute;**

**private Counting second;**

**public Chromometer() {**

**hour = new CyclicCounter(0, 99);**

**minute = new CyclicCounter(0, 59);**

**second = new CyclicCounter(0, 59);**

**}**

**public void increment() {**

**second.increment();**

**if (second.getValue() == 0) {**

**minute.increment();**

**if (minute.getValue() == 0) {**

**hour.increment();**

**}**

**}**

**}**

**public void decrement() {**

**second.decrement();**

**if (second.getValue() == 59) {**

**minute.decrement();**

**if (minute.getValue() == 59) {**

**hour.decrement();**

**}**

**}**

**}**

**@Override**

**public String toString() {**

**return "Chromometer [hour=" + hour.getValue() + ", minute=" + minute.getValue() + ", second=" + second.getValue() + "]";**

**}**

**}**

**package lab9.chronometer;**

**public class TestChronometer {**

**public static void main(String[] args) {**

**Chromometer chromometer = new Chromometer();**

**System.out.println(chromometer.toString());**

**for (int i = 1; i <= 5000; i++) {**

**chromometer.increment();**

**}**

**System.out.println(chromometer.toString());**

**for (int i = 1; i <= 3000; i++) {**

**chromometer.decrement();**

**}**

**System.out.println(chromometer.toString());**

**}**

**}**

### Exercise 4:

Take the different examples and implement dynamic polymorphism in the following situations:

* the counters: define a single test class for the 3 counters by systematically using the Counter class when declaring the variables (Counter, Terminal Counter, Cyclic Counter type objects).
* Start over with the Counting interface.
* Use the Counting interface for the stopwatch test.

Answers

**The Soulution for this question included in the answer of the previous question.**

### Exercise 5:

Define a Java interface for the phone book problem and test.

Answers

**package lab9.phonebook;**

**public class Contact {**

**private String name;**

**private String number;**

**private String email;**

**public Contact(String name, String number, String email) {**

**this.name = name;**

**this.number = number;**

**this.email = email;**

**}**

**public String toString() {**

**return "Contact [name=" + name + ", number=" + number + ", email=" + email + "]";**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getNumber() {**

**return number;**

**}**

**public void setNumber(String number) {**

**this.number = number;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**}**

**package lab9.phonebook;**

**public interface PhonebookInterface {**

**public void add(Contact contact);**

**public Contact searchByName(String name);**

**public Contact searchByNumber(String number);**

**public void modify(String name, String newNumber);**

**public void remove(String name);**

**}**

**package lab9.phonebook;**

**public class PhonebookTable implements PhonebookInterface {**

**private Contact[] contacts;**

**private int currentSize = 0;**

**public PhonebookTable(int size) {**

**contacts = new Contact[size];**

**}**

**public void add(Contact contact) {**

**if (currentSize != contacts.length) {**

**contacts[currentSize] = contact;**

**currentSize++;**

**}**

**}**

**public Contact searchByName(String name) {**

**for (int index = 0; index < currentSize; index++) {**

**if (contacts[index].getName().equals(name)) {**

**return contacts[index];**

**}**

**}**

**return null;**

**}**

**public Contact searchByNumber(String number) {**

**for (int index = 0; index < currentSize; index++) {**

**if (contacts[index].getNumber().equals(number)) {**

**return contacts[index];**

**}**

**}**

**return null;**

**}**

**public void modify(String name, String newNumber) {**

**Contact contact = searchByName(name);**

**if (contact != null) {**

**contact.setNumber(newNumber);**

**}**

**}**

**public void remove(String name) {**

**int index;**

**// Remove the contact**

**for (index = 0; index < currentSize; index++) {**

**if (contacts[index].getName().equals(name)) {**

**contacts[index] = null;**

**break;**

**}**

**}**

**// Move the next contacts**

**for (int newIndex = index; index < currentSize - 1; index++) {**

**contacts[newIndex] = contacts[newIndex + 1];**

**}**

**if (index != currentSize) {**

**currentSize--;**

**}**

**}**

**@Override**

**public String toString() {**

**String result = "All contacts in phonebook : \n";**

**for (int index = 0; index < currentSize; index++) {**

**result += contacts[index].toString() + "\n";**

**}**

**return result;**

**}**

**}**

**package lab9.phonebook;**

**public class TestPhonebook {**

**public static void main(String[] args) {**

**PhonebookInterface phonebook = new PhonebookTable(50);**

**Contact paul = new Contact("Paul", "123456", "paul@gmail.com");**

**Contact jean = new Contact("Jean", "234567", "jean@hotmail.com");**

**phonebook.add(paul);**

**phonebook.add(jean);**

**Contact foundPaul = phonebook.searchByName("Paul");**

**System.out.println(foundPaul.toString());**

**Contact foundJean = phonebook.searchByNumber("234567");**

**System.out.println(foundJean.toString());**

**System.out.println("-----------------------------");**

**System.out.println("After modification : ");**

**phonebook.modify("Paul", "222222");**

**Contact foundPaul2 = phonebook.searchByName("Paul");**

**System.out.println(foundPaul2.toString());**

**System.out.println(phonebook.toString());**

**System.out.println("-----------------------------");**

**phonebook.remove("Paul");**

**System.out.println("After removing : ");**

**System.out.println(phonebook);**

**}**

**}**