Наследование классов. Графический интерфейс AWT.

Цель работы

Лабораторная работа посвящена построению приложений с использованием механизма наследования и графического интерфейса библиотеки java.awt.

Указания к работе

Графический пользовательский интерфейс (GUI) – основной способ взаимодействия конечных пользователей с java-приложением. Для разработки прикладного программного обеспечения на языке Java, а точнее графического интерфейса приложений, обычно используются пакеты AWT и Swing.

AWT (для доступа загружается пакет java.awt) содержит набор классов, позволяющих выполнять графические операции и создавать оконные элементы управления, подобно тому, как это делается в VBA и Delphi

Варианты заданий

Первый

1. Создайте класс Rectangle, представляющий прямоугольник, экземпляры которого обладают четырьмя полями целого типа (x1,y1) (левый верхний угол), (x2, y2) (правый нижний угол). Для данного класса создать три конструктора, которые инициализируют поля следующим образом:
 конструктор принимает 4 параметра целого типа и присваивает их значения полям (x1, y1), (x2, y2);
 конструктор принимает 2 параметра целого типа – ширину и высоту прямоугольника, а левый верхний угол прямоугольника помещает в координату (0,0);
 конструктор не принимает никаких параметров – создает вырожденный прямоугольник с координатами углов (0,0) и (0,0).
В классе Rectangle создать метод rect_print(), выдающий текущее состояние экземпляра прямоугольника (значение полей). Создать метод move (int dx, int dy), перемещающий прямоугольник по горизонтали на заданное dx, по вертикали на заданное dy. Создать метод Union (подумать какие входные параметры), возвращающий объединение этого прямоугольника с другим прямоугольником (возвращается наименьший прямоугольник, содержащий оба прямоугольника).
Для проверки работоспособности класса Rectangle создайте в отдельном файле

класс Test, содержащий функцию main(...).
Протестируйте в ней поведение экземпляров класса Rectangle следующим образом: создайте три объекта Rectangle тремя различными созданными конструкторами, выведите состояние всех трех объектов. Воспользуйтесь вызовом функции move(...) с различными значениями параметров для каждого объекта и выведите новое положение созданных прямоугольников.
Протестируйте работу функции Union на одном примере.

2. Расширьте класс Rectangle новым классом DrawableRect, у которого есть метод прорисовки draw(Graphics g) и поле outColor с типом данных Color (из пакета java.awt.*). Это поле служит для задания цвета границы прямоугольника. Для отображения прямоугольника в пакете java.awt.* существует специальный класс Graphics (его нужно импортировать в вашу программу с помощью import java.awt.*;). У экземпляров этого класса есть метод по установлению значения цвета рисуемого объекта (например, setColor(Color.red)) и метод рисования прямоугольника по 2 координатам (x,y), ширине w и высоте h drawRect(x,y,h,w) (рисует только границы прямоугольника, внутренность не закрашена).

3. Расширьте класс DrawableRect новым классом ColoredRect, в котором есть поле inColor с типом Color. Метод прорисовки draw(Graphicsg) перекрывается следующим образом: прямоугольник отображается двумя цветами – граница цветом outColor, внутренность – inColor. Метод fillRect(x,y,h,w) (рисует закрашенный прямоугольник). Тестирование унаследованных от Rectangle классов DrawableRect и ColoredRect производить не нужно.

Второй

1. Создайте класс Circle, представляющий круг, экземпляры которого обладают двумя полями целого типа (x1,y1) (центр), и один вещественного типа (r) (радиус). Для данного класса создать три конструктора, которые инициализируют поля следующим образом:

- конструктор принимает 2 параметра целого типа и один вещественного присваивает их значения полям (x1, y1), (r);
- конструктор принимает 1 параметр вещественного типа – радиус окружности, а центр окружности помещает в координату (0,0);
- конструктор не принимает никаких параметров – создает вырожденную окружность с координатами (0,0) и радиус (0).

В классе Circle создать метод circle_print(), выдающий текущее состояние экземпляра окружности (значение полей). Создать метод move (int dx, int dy), перемещающий окружность по горизонтали на заданное dx, по вертикали на заданное dy. Создать метод Union (подумать какие входные параметры), возвращающий объединение этой окружности с другой окружностью (возвращается наименьшая окружность, вписанная в большую, их центры совпадают).

Для проверки работоспособности класса Circle создайте в отдельном файле класс Test, содержащий функцию main(...).
Протестируйте в ней поведение экземпляров класса Circle следующим образом: создайте три объекта Circle тремя различными созданными конструкторами, выведите состояние всех трех объектов. Воспользуйтесь вызовом функции move(...) с различными значениями параметров для каждого объекта и выведите новое положение созданных окружностей.
Протестируйте работу функции Union на одном примере.

2. Расширьте класс Circle новым классом DrawableCircle, у которого есть метод прорисовки draw(Graphics g) и поле outColor с типом данных Color (из пакета java.awt.*). Это поле служит для задания цвета границы окружности. Для отображения окружности в пакете java.awt.* существует специальный класс Graphics (его нужно импортировать в вашу программу с помощью import java.awt.*;). У экземпляров этого класса есть метод по установлению значения цвета рисуемого объекта (например, setColor(Color.red)) и метод рисования окружности по 2 координатам центра окружности (x,y), и радиусу r drawCircle (x,y,r) (рисует только границы окружности, внутренность не закрашена).

3. Расширьте класс DrawableCircle новым классом ColoredCircle, в котором есть поле inColor с типом Color. Метод прорисовки draw(Graphicsg) перекрывается следующим образом: окружность отображается двумя цветами – граница цветом outColor, внутренность – inColor. Метод fillCircle (x,y,r) (рисует закрашенную окружность). Тестирование унаследованных от Circle классов DrawableCircle и ColoredCircle производить не нужно.

Третий

1. Создайте класс Ellipse, представляющий эллипс, экземпляры которого обладают двумя полями целого типа (x,y) (фокус), и два вещественного типа (a,b) (большая и малая полуось). Для данного класса создать три конструктора, которые инициализируют поля следующим образом:

- конструктор принимает два параметра целого типа и два вещественного типа присваивает их значения полям (x, y), (a,b);
- конструктор принимает 2 параметр вещественного типа – большая и малая полуось, а фокус эллипса помещает в координату (0,0);
- конструктор не принимает никаких параметров – создает вырожденный эллипс с координатами (0,0) и большая и малая полуось (0,0).

В классе Ellipse создать метод Ellipse_print(), выдающий текущее состояние экземпляра окружности (значение полей). Создать метод move (int dx, int dy), перемещающий окружность по горизонтали на заданное dx, по вертикали на заданное dy. Создать метод Union (подумать какие входные параметры), возвращающий объединение этой окружности с другой окружностью (возвращается наименьшая окружность, вписанная в большую, их центры

совпадают).
Для проверки работоспособности класса Ellipse создайте в отдельном файле класс Test, содержащий функцию main(...).
Протестируйте в ней поведение экземпляров класса Ellipse следующим образом: создайте три объекта Ellipse тремя различными созданными конструкторами, выведите состояние всех трех объектов. Воспользуйтесь вызовом функции move(...) с различными значениями параметров для каждого объекта и выведите новое положение созданных окружностей.
Протестируйте работу функции Union на одном примере.

2. Расширьте класс Ellipse новым классом DrawableEllipse, у которого есть метод прорисовки draw(Graphics g) и поле outColor с типом данных Color (из пакета java.awt.*). Это поле служит для задания цвета границы окружности. Для отображения окружности в пакете java.awt.* существует специальный класс Graphics (его нужно импортировать в вашу программу с помощью import java.awt.*;). У экземпляров этого класса есть метод по установлению значения цвета рисуемого объекта (например, setColor(Color.red)) и метод рисования окружности по 2 координатам центра окружности (x,y), и радиусу r drawEllipse (x,y,r) (рисует только границы окружности, внутренность не закрашена).

3. Расширьте класс DrawableEllipse новым классом ColoredEllipse, в котором есть поле inColor с типом Color. Метод прорисовки draw(Graphicsg) перекрывается следующим образом: окружность отображается двумя цветами – граница цветом outColor, внутренность – inColor. Метод fillEllipse(x,y,r) (рисует закрашенную окружность). Тестирование унаследованных от Circle классов DrawableEllipse и ColoredEllipse производить не нужно.