**Задание 1.**

***1. Разработать абстрактный класс*** «геометрическая фигура» с методами площади, периметра.

***2. Сделать наследников*** с реализацией методов (конструкторы, площадь, периметр, toString). Примеры наследников: треугольник, параллелограмм, трапеция, круг, ромб, прямоугольник, окружность, эллипс (минимум 3 наследника). Реализовать в них методы абстрактных классов.

***3. Протестировать.***

***4. Разработать класс «Составная фигура»***, который внутри содержит массив геометрический фигур (экземпляров класса «геометрическая фигура». Для данного класса сделать метод нахождения общей площади и общего периметра всех фигур. + необходимые методы инициализации, добавления, удаления фигур в массив (добавления будет достаточно). toString, выводящий все фигуры в данном классе. «Составная фВыигура никак не должна взаимодействовать с наследниками фигуры, а только с классом «Геометрическая фигура».

***abstract class GeometricFigure***

package com.company;  
  
//Абстрактный класс «геометрическая фигура»  
public abstract class GeometricFigure *{* // имя фигуры  
 protected String name;  
  
 // конструкторы  
 //1. конструктор без параметров  
 public GeometricFigure*() {* name = "";  
 *}* //2. конструктор с одним параметром  
 public GeometricFigure*(*String *name) {* this.name = *name*;  
 *}* // методы  
 // 1. метод получения площади  
 public abstract double S*()*;  
  
 // 2. метод получения периметра  
 public abstract double P*()*;  
  
 // 3. метод представления класса в виде строки  
 @Override  
 public String toString*() {* return name;  
 *}  
}*

***class Square extends GeometricFigure***

package com.company;  
  
// класс квадрат  
public class Square extends GeometricFigure *{* // поля квадрата  
 private double a;  
  
 // конструкторы  
 //1. конструктор без параметров  
 public Square*() {* a = 0;  
 *}* //2. конструктор с двумя параметрами  
 public Square*(*String *name*, double *a) {* super*(name)*;  
 this.a = *a*;  
 *}* // методы  
 // 1. метод получения площади - перегрузка метода S() базового класса  
 @Override  
 public double S*() {* return a \* a;  
 *}* // 2. метод получения периметра - перегрузка метода P() базового класса  
 @Override  
 public double P*() {* return a \* 4;  
 *}* // 3. метод представления класса в виде строки - перегрузка метода toString  
 @Override  
 public String toString*() {* return name + ": a = " + a;  
 *}  
}*

***class Rectangle extends GeometricFigure***

package com.company;  
  
// класс прямоугольник  
public class Rectangle extends GeometricFigure *{* // поля прямоугольника  
 private double a;  
 private double b;  
  
 // конструкторы  
 //1. конструктор без параметров  
 public Rectangle*() {* a = b = 0;  
 *}* //2. конструктор с 3 параметрами  
 public Rectangle*(*String *name*, double *a*, double *b) {* super*(name)*;  
 this.a = *a*;  
 this.b = *b*;  
 *}* //3. делегируем конструктор с 3 параметрами  
 public Rectangle*(*String *name*, double *x) {* this*(name*, *x*, *x)*;  
 *}* // методы  
 // 1. метод получения площади - перегрузка метода S() базового класса  
 @Override  
 public double S*() {* return a \* b;  
 *}* // 2. метод получения периметра - перегрузка метода P() базового класса  
 @Override  
 public double P*() {* return *(*a + b*)* \* 2;  
 *}* // 3. метод представления класса в виде строки - перегрузка метода toString  
 @Override  
 public String toString*() {* return name + ": a = " + a + "," + " b = " + b;  
 *}  
}*

***class Circle extends GeometricFigure***

package com.company;  
  
// класс окружность  
public class Circle extends GeometricFigure *{* // поля окружности  
 private double r; // радиус окружности  
  
 // конструкторы  
 //1. конструктор без параметров  
 public Circle*() {* r = 0;  
 *}* //2. конструктор с одним параметром  
 public Circle*(*double *r) {* this.r = *r*;  
 *}* //3. конструктор с двумя параметрами  
 public Circle*(*String *name*, double *r) {* super*(name)*;  
 this.r = *r*;  
 *}* // методы  
 // 1. метод получения площади - перегрузка метода S() базового класса  
 @Override  
 public double S*() {* return Math.*PI* \* r \* r;  
 *}* // 2. метод получения периметра - перегрузка метода P() базового класса  
 @Override  
 public double P*() {* return 2 \* Math.*PI* \* r;  
 *}* // 3. метод представления класса в виде строки - перегрузка метода toString  
 @Override  
 public String toString*() {* return name + ": r = " + r;  
 *}  
  
}*

***class Triangle extends GeometricFigure***

package com.company;  
  
// класс треугольник  
public class Triangle extends GeometricFigure *{* // поля треугольника  
 private double a; // сторона a  
 private double b; // сторона b  
 private double c; // сторона b  
  
  
 // 1. конструктор без параметров  
 public Triangle*() {* this.name = "name";  
 this.a = 1;  
 this.b = 1;  
 this.c = 1;  
 *}* // 2. конструктор с 4-мя параметрами  
 public Triangle*(*String *name*, double *a*, double *b*, double *c) {* this.name = *name*;  
 this.a = *a*;  
 this.b = *b*;  
 this.c = *c*;  
 *}* // методы  
 // 1. метод получения площади - перегрузка метода S() базового класса  
 @Override  
 public double S*() {* double *halfP* = P*()* / 2;  
 return Math.*sqrt(halfP* \* *(halfP* - a*)* \* *(halfP* - b*)* \* *(halfP* - c*))*;  
 *}* // 2. метод получения периметра - перегрузка метода P() базового класса  
 @Override  
 public double P*() {* return a + b + c;  
 *}* // 3. метод представления класса в виде строки - перегрузка метода toString  
 @Override  
 public String toString*() {* return name + ": a = " + a + ", b = " + b + ", c = " + c;  
 *}  
}*

***class CompoundFigure***

package com.company;  
  
import java.util.Arrays;  
  
// класс «составная фигура»  
public class CompoundFigure *{* // поля составной фигуры  
 private GeometricFigure*[]* figures;  
  
 // конструкторы  
 //1. конструктор без параметров  
 public CompoundFigure*() {* figures = new GeometricFigure*[*0*]*;  
 *}* //2. конструктор с одним параметром  
 public CompoundFigure*(*int *figure) {* figures = new GeometricFigure*[figure]*;  
 *}* //setter  
 public void setFigure*(*GeometricFigure *figure*, int *pos) {* figures*[pos]* = *figure*;  
 *}* // методы  
 // 1. метод добавления новой фигуры в массив  
 public void addFigure*() {* figures = Arrays.*copyOf(*figures, figures.length + 1*)*;  
 *}* // 2. метод вывода параметров всех фигур  
 public void allOptionsFigure*() {* for *(*GeometricFigure *figure* : figures*) {* if *(figure* != null*)* System.*out*.println*(figure* + ", S = " + *figure*.S*()* + ", P = " + *figure*.P*())*;  
 *}  
 }* // 3. метод вывода общей площади составной фигуры  
 public double generalAreaFigure*() {* double *res* = 0;  
 for *(*GeometricFigure *figure* : figures*) {  
 res* += *figure*.S*()*;  
 *}* return *res*;  
 *}* // 4. метод вывода общего периметра составной фигуры  
 public double generalPerimeterFigure*() {* double *res* = 0;  
 for *(*GeometricFigure *figure* : figures*) {  
 res* += *figure*.P*()*;  
 *}* return *res*;  
 *}* // 5. метод представления класса в виде строки - перегрузка метода toString  
 @Override  
 public String toString*() {* StringBuilder *sb* = new StringBuilder*()*;  
  
 for *(*GeometricFigure *figure* : figures*) {* if *(figure* == null*)  
 sb*.append*(*"empty seat"*)*;  
 else  
 *sb*.append*(figure)*.append*(*"\n"*)*;  
 *}* return *sb*.toString*()*;  
 *}  
}*

***class ShowFigure***

package com.company;  
  
public class ShowFigure {  
 //метод тестирования фигуры  
 static void printGeometricFigure(GeometricFigure *figure*) {  
 System.*out*.println(*figure*);  
 System.*out*.println("S = " + *figure*.S());  
 System.*out*.println("P = " + *figure*.P());  
 }  
  
 public static void main(String[] *args*) {  
  
 System.*out*.println("1.Тестирование класса \"Геометрическая фигура\": ");  
 // содаём объект прямоугольник  
 Rectangle *rect* = new Rectangle("Rectangle", 3, 4);  
 *printGeometricFigure*(*rect*);//вывод площади и периметра  
 System.*out*.println();  
  
 //содаём объект треугольник  
 Triangle *triangle* = new Triangle("Triangle", 3, 4, 5);  
 *printGeometricFigure*(*triangle*);//вывод площади и периметра  
 System.*out*.println();  
  
 //содаём объект квадрат  
 Square *square* = new Square("Square", 3);  
 *printGeometricFigure*(*square*);//вывод площади и периметра  
 System.*out*.println();  
  
 //содаём объект окружность  
 Circle *circle* = new Circle("Circle", 3);  
 *printGeometricFigure*(*circle*);//вывод площади и периметра  
 System.*out*.println();  
  
 //Создание составной фигуры  
 System.*out*.println("2.Тестирование класса \"Составная фигура\": ");  
 CompoundFigure *comp* = new CompoundFigure(4);//Массив из 4-х фигур  
  
 //поместили прямоугольник в класс «Составная фигура», на 1-е место в массиве  
 *comp*.setFigure(*rect*, 0);  
  
 //поместили треугольник в класс «Составная фигура», на 2-е место в массиве  
 *comp*.setFigure(*triangle*, 1);  
  
 //поместили квадрат в класс «Составная фигура», на 3-е место в массиве  
 *comp*.setFigure(*square*, 2);  
  
 //поместили окружность в класс «Составная фигура», на 4-е место в массиве  
 *comp*.setFigure(*circle*, 3);  
  
 //вывод в консоль массива составной фигуры  
 System.*out*.println(*comp*);  
  
 //добавляем новую фигуру в массив  
 *comp*.addFigure();  
 System.*out*.println("Добавляем новую фигуру в массив фигур: ");  
 *comp*.setFigure(new Square("Square2", 8), 4);  
  
 //вывод всех параметров фигур массива с учетом новой добавленной фигурой  
 *comp*.allOptionsFigure();  
 System.*out*.println();  
  
 //Вывод в консоль общей площади и периметра всех фигур массива  
 System.*out*.println("Общая площадь S = " + *comp*.generalAreaFigure());  
 System.*out*.println("Общий периметр P = " + *comp*.generalPerimeterFigure());  
 }  
}

***Тест:***

