**Answer 1:**

public class Circle extends GeometricObject implements Comparable<Circle> {

private double radius;

public Circle() {

}

public Circle(double radius) {

this.radius = radius;

}

public Circle(double radius,

String color, boolean filled) {

this.radius = radius;

setColor(color);

setFilled(filled);

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

public double getArea() {

return radius \* radius \* Math.PI;

}

public double getDiameter() {

return 2 \* radius;

}

public double getPerimeter() {

return 2 \* radius \* Math.PI;

}

public boolean equals(Object o) {

return this.compareTo((Circle)o) == 0;

}

public int compareTo(Circle o) {

if (this.radius > o.radius)

return 1;

else if (this.radius < o.radius)

return -1;

else return 0;

}

public String toString() {

return super.toString() + "\nDate created: " + getDateCreated() +

"\nRadius: " + radius;

}

}

/////////////////////////

public class Rectangle extends GeometricObject implements Comparable<Rectangle> {

private double width;

private double height;

public Rectangle() {

}

public Rectangle(

double width, double height) {

this.width = width;

this.height = height;

}

public Rectangle(

double width, double height, String color, boolean filled) {

this.width = width;

this.height = height;

setColor(color);

setFilled(filled);

}

public double getWidth() {

return width;

}

public void setWidth(double width) {

this. width = width;

}

public double getheight() {

return height;

}

public void setheight(double height) {

this.height = height;

}

public double getArea() {

return width \* height;

}

public double getPerimeter() {

return 2 \* (width \* height);

}

public int compareTo(Rectangle o) {

if (getArea() > o.getArea())

return 1;

else if (getArea() < o.getArea())

return -1;

else return 0;

}

public boolean equals(Object o) {

return this.compareTo((Rectangle)o) == 0;

}

public String toString() {

return super.toString() + "\nWidth: " + width + "\nHeight: " + height

+ "\nArea: " + getArea() + "\nPerimeter: " + getPerimeter();

}

}

///////////////////////////////////////////////////////////////////////////////////////

public class GeometricObjectComparator implements Comparator<GeometricObject>, java.io.Serializable {

public int compare(GeometricObject o1, GeometricObject o2) {

double area1 = o1.getArea();

double area2 = o2.getArea();

if (area1 < area2)

return -1;

else if (area1 == area2)

return 0;

else return 1;

}

}

**Answer 2:**

public class Triangle extends GeometricObject {

private double side1;

private double side2;

private double side3;

public Triangle(double side1, double side2, double side3) throws IllegalTriangleException {

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

isValidTriangle();

}

public Triangle() {

this.side1 = 1;

this.side2 = 1;

this.side3 = 1;

}

public double getArea() {

double s = (side1 + side2 + side3) / 2.0;

return Math.pow(s \* (s - side1) \* (s - side2) \* (s - side3), 0.5);

}

public double getPerimeter() {

return side1 + side2 + side3;

}

public String toString() {

return "Triangle{" +"side1=" + side1 + ", side2=" + side2 + ", side3=" + side3 + '}';

}

public static boolean isTriangle(double side1, double side2, double side3) {

return ((side1 + side2 > side3) &&

(side1 + side3 > side2) &&

(side3 + side2 > side1));

}

public double getSide1() {

return side1;

}

public void setSide1(double side1) throws IllegalTriangleException {

this.side1 = side1;

isValidTriangle();

}

public double getSide2() {

return side2;

}

public void setSide2(double side2) throws IllegalTriangleException{

this.side2 = side2;

isValidTriangle();

}

public double getSide3() {

return side3;

}

public void setSide3(double side3) throws IllegalTriangleException {

this.side3 = side3;

isValidTriangle();

}

private void isValidTriangle() throws IllegalTriangleException{

if (!isTriangle(side1, side2, side3)) {

throw new IllegalTriangleException(side1, side2, side3);

}

}

public class IllegalTriangleException extends IllegalArgumentException

{

private double s1;

private double s2;

private double s3;

public IllegalTriangleException(double s1, double s2, double s3) {

super("Not a real triangle:" + " side1 = " + s1 + " side2 = " + s2 + " side3 = " + s3);

this.s1 = s1;

this.s2 = s2;

this.s3 = s3;

}

public double getS1() {

return s1;

}

public double getS2() {

return s2;

}

public double getS3() {

return s3;

}

}

}

**Answer 3:**

public static void main(String[] args) {

Circle circle1 = new Circle(15, "red", true);

Circle circle2 = new Circle(10, "blue", false);

System.out.println(Circle 1);

System.out.println(Circle 2);

print("\nThe larger of the two circles was ");

print(Circle.max(circle1, circle2));

Rectangle rectangle1 = new Rectangle(4, 5, "green", true);

Rectangle rectangle2 = new Rectangle(4.2, 5, "orange", true);

System.out.println(Rectangle 1);

System.out.println(Rectangle 2);

print("\nThe larger of the two rectangles was ");

print(Rectangle.max(rectangle1, rectangle2));

}

public static void print(String s) {

System.out.println(s);

}

public static void print(GeometricObject o) {

System.out.println(o);

}

}

**Answer 4:**

public static void main(String[] args) {

ComparableCircle 1 = new ComparableCircle(12.5);

ComparableCircle 2 = new ComparableCircle(18.3);

System.out.println(ComparableCircle1);

System.out.println(ComparableCircle2);

System.out.println((comparableCircle1.compareTo(comparableCircle2) == 1

? "\nComparableCircle1 " : "\nComparableCircle2 ") +

"is the larger of the two Circles");

}

}

**Answer 5**

public class {

public static void main(String[] args) {

geoShapes[1] = new Square();

geoShapes[2] = new Square(0,0,25);

for (int i = 0; i < geoShapes.length; i++) {

System.out.println("shape #" + (i + 1) + " area = " + geoShapes[i].getArea());

if (geoShapes[i] instanceof Colorable) {

System.out.println("How to color: "+((Colorable)geoShapes[i]).howToColor());

}

}

}

}

class Square extends GeometricObject implements Colorable {

private double x;

private double y;

private double side;

Square() {

this(0,0,10);

}

Square(double x, double y, double side) {

this.x = x;

this.y = y;

this.side = side;

}

public String howToColor() {

return "Color all four sides.";

}

public double getX() {

return x;

}

public void setX(double x) {

this.x = x;

}

public double getY() {

return y;

}

public void setY(double y) {

this.y = y;

}

public double getSide() {

return side;

}

public void setSide(double side) {

this.side = side;

}

}

interface Colorable {

String howToColor();

}

**Answer 6:**

public class {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] array = getArray();

System.out.print("Enter the index of the array: ");

try {

System.out.println("The corresponding element value is " +

array[input.nextInt()]);

}

catch (ArrayIndexOutOfBoundsException ex) {

System.out.println("Out of Bounds.");

}

}

public static int[] getArray() {

int[] array = new int[100];

for (int i = 0; i < array.length; i++) {

array[i] = (int)(Math.random() \* 100) + 1;

}

return array;

}

}

**Answer 7:**

public class {

public static void main(String[] args) {

String allInOne = "";

for (String arg : args) {

allInOne += arg;

}

String[] tokens = MyString1.split(allInOne, "[+-\*/]");

if (tokens.length != 3) {

System.out.println("Usage: java Calculator \"operand1 operator operand2\"");

System.exit(0); }

int result = 0;

try {

switch (tokens[1].charAt(0)) {

case '+':

result = Integer.parseInt(tokens[0]) + Integer.parseInt(tokens[2]);

break;

case '-':

result = Integer.parseInt(tokens[0]) - Integer.parseInt(tokens[2]);

break;

case '\*':

result = Integer.parseInt(tokens[0]) \* Integer.parseInt(tokens[2]);

break;

case '/':

result = Integer.parseInt(tokens[0]) / Integer.parseInt(tokens[2]);

}

} catch (NumberFormatException ex) {

System.out.println("ERROR "+ ex.getMessage());

System.exit(0);

}

System.out.println(tokens[0] + ' ' + tokens[1] + ' ‘ + tokens[2] + " = " + result);

}

public static boolean isNumeric(String s) {

for (char ch : s.toCharArray()) {

if (ch < '0' || ch > '9') return false;

}

return true;

}

}

**Answer 8:**

public class InvalidRadiusException extends Exception {

public InvalidRadiusException (String message) {

super("Exception: " + message);

}

}

**Answer 9:**

public class {

public static void main(String[] args) throws Exception {

if (args.length < 1) {

System.out.println("Use: java Loan.java");

System.exit(1);

}

File file = new File(args[0]);

if (!file.exists()) {

System.out.println("please include the source file");

System.exit(2);

}

Scanner in = new Scanner(file);

long charCount = 0L;

int lines = 0;

int words = 0;

while(in.hasNext()) {

String line = in.nextLine();

String[] wordArray = line.split(" ");

charCount += line.length();

lines += 1;

words += wordArray.length;

}

System.out.printf("File %s has%n" + "%d characters%n" + "%d words%n" + "%d lines%n%n", args[0], charCount, words, lines);

}

}

**Answer 10:**

public class {

public static void main(String[] args) {

ArrayList<Double> assistant = new ArrayList<>();

ArrayList<Double> associate = new ArrayList<>();

ArrayList<Double> full = new ArrayList<>();

try {

java.net.URL url = new java.net.URL(

"http://cs.armstrong.edu/liang/data/Salary.txt");

Scanner input = new Scanner(url.openStream());

while (input.hasNext()) {

String[] line = (input.nextLine()).split(" ");

processData(assistant, associate, full, line[2],

new Double(line[3]));

}

}

catch (java.net.MalformedURLException ex) {

System.out.println("Invalid URL");

System.exit(0);

}

catch (java.io.IOException ex) {

System.out.println("I/O Errors: no such file");

System.exit(1);

}

double totalAssistant = getTotal(assistant);

double totalAssociate = getTotal(associate);

double totalFull = getTotal(full);

double totalFaculty = (totalAssistant + totalAssociate + totalFull);

System.out.println("\n Total salary");

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

System.out.printf("Assistant professors: $%.2f\n", totalAssistant);

System.out.printf("Associate professors: $%.2f\n", totalAssociate);

System.out.printf("Full professors: $%.2f\n", totalFull);

System.out.printf("All faculty: $%.2f\n", totalFaculty);

System.out.println("\n Average salary");

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

System.out.printf("Assistant professors: $%.2f\n",

(totalAssistant / assistant.size()));

System.out.printf("Associate professors: $%.2f\n",

(totalAssociate / associate.size()));

System.out.printf("Full professors: $%.2f\n",

(totalFull / full.size()));

System.out.printf("All faculty: $%.2f\n",

(totalFaculty / (assistant.size() + associate.size() + full.size())));

}

public static double getTotal(ArrayList<Double> list) {

double total = 0;

for (int i = 0; i < list.size(); i++) {

total += list.get(i);

}

return total;

}

public static void processData(ArrayList<Double> a, ArrayList<Double> b,

ArrayList<Double> c, String rank, double salary) {

if (rank.equals("assistant")){

a.add(salary);

}

else if (rank.equals("associate"))

b.add(salary);

else if (rank.equals("full"))

c.add(salary);

}

}