13.3

import java.io.\*;

import java.util.\*;

public class test {

public static void main(String[] args) {

ArrayList<Number> list = new ArrayList<Number>();

list.add(14);

list.add(24);

list.add(4);

list.add(42);

list.add(5);

System.out.println("Unsorted: " + list.toString());

sort(list);

System.out.println("Sorted: " + list.toString());

}

public static void sort(ArrayList<Number> list) {

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

Number currentMin = list.get(i);

int currentMinIndex = i;

for (int j = i + 1; j < list.size(); j++) {

if (currentMin.doubleValue() > list.get(j).doubleValue()) {

currentMin = list.get(j);

currentMinIndex = j;

}

}

if (currentMinIndex != i) {

list.set(currentMinIndex, list.get(i));

list.set(i, currentMin);

}

}

}

}

--------------------------------------------

13.5

import java.io.\*;

import java.util.\*;

public class test{

public static void main(String[] args) {

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

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

System.out.println("\nCircle 1: ");

print(circle1);

System.out.println("\nCircle 2: ");

print(circle2);

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("\nRectangle 1: ");

print(circle1);

System.out.println("\nRectangle 2: ");

print(circle2);

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);

}

}

class Circle extends GeometricObject {

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 String toString() {

return "\nRadius: " + radius + "\nArea: " + getArea() +

"\nDiameter: " + getDiameter() + "\nPerimeter: " + getPerimeter();

}

}

-------------------------------

13.9

import java.io.\*;

import java.util.\*;

public class test {

public static void main(String[] args) {

Circle c1 = new Circle(10);

Circle c2 = new Circle(10);

MyRectangle2D r1 = new MyRectangle2D(0, 0, 5, 5);

System.out.println("c1 equals c2? " + c1.equals(c2));

System.out.println("c1 equals r1? " + c1.equals(r1));

}

}

class Circle extends GeometricObject {

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 obj) {

return obj instanceof Circle && radius == ((Circle) obj).radius;

}

public void printCircle() {

System.out.println("The circle is created " + getDateCreated() +

" and the radius is " + radius);

}

}

-------------------------------------------

13.11

import java.io.\*;

import java.util.\*;

public class test {

public static void main(String[] args) {

Octagon octagon1 = new Octagon(5);

System.out.println("\nOctagon:\nArea: " + octagon1.getArea() +

"\nPerimeter: " + octagon1.getPerimeter());

System.out.println("Cloning Octagon...");

Octagon octagon2 = (Octagon)octagon1.clone();

int result = (octagon1.compareTo(octagon2));

if (result == 1)

System.out.println("Octagon is greather than its clone.");

else if (result == -1)

System.out.println("Octagon is less than its clone.");

else

System.out.println("Octagon is equal to its clone.");

}

}

public class Octagon extends GeometricObject

implements Cloneable, Comparable<Octagon> {

private double side;

public Octagon() {

}

public Octagon(double side) {

this.side = side;

}

public void setSide(double side) {

this.side = side;

}

public double getSide() {

return side;

}

public double getArea() {

return (2 + 4 / Math.sqrt(2)) \* side \* side;

}

public double getPerimeter() {

return 8 \* side;

}

public int compareTo(Octagon o) {

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

return 1;

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

return -1;

else

return 0;

}

public Object clone() throws CloneNotSupportedException {

return super.clone();

}

public String toString() {

return super.toString() + "\nArea: " + getArea() +

"\nPerimeter: " + getPerimeter();

}

}

--------------------------

13.15

import java.io.\*;

import java.math.BigInteger;

import java.util.\*;

public class test {

public static void main(String[] args) {

Rational r1 = new Rational(new BigInteger("400000"),

new BigInteger("200000"));

Rational r2 = new Rational(new BigInteger("200000"),

new BigInteger("300000"));

System.out.println(r1 + " + " + r2 + " = " + r1.add(r2));

System.out.println(r1 + " - " + r2 + " = " + r1.subtract(r2));

System.out.println(r1 + " \* " + r2 + " = " + r1.multiply(r2));

System.out.println(r1 + " / " + r2 + " = " + r1.divide(r2));

System.out.println(r2 + " is " + r2.doubleValue());

}

}

class Rational extends Number implements Comparable {

private BigInteger[] r = new BigInteger[2];

public Rational() {

this(new BigInteger("0"), new BigInteger("1"));

}

public Rational(BigInteger numerator, BigInteger denominator) {

BigInteger gcd = gcd(numerator, denominator);

r[0] = (denominator.compareTo(BigInteger.ZERO) > 0

? BigInteger.ONE :

new BigInteger("-1")).multiply(numerator.divide(gcd));

r[1] = denominator.divide(gcd);

}

private static BigInteger gcd(BigInteger n, BigInteger d) {

BigInteger n1 = n;

BigInteger n2 = d;

BigInteger gcd = BigInteger.ONE;

for (BigInteger k = BigInteger.ONE;

k.compareTo(n1) <= 0 && k.compareTo(n2) <= 0;

k = k.add(BigInteger.ONE)) {

if (n1.remainder(k).compareTo(BigInteger.ZERO) == 0 &&

n2.remainder(k).compareTo(BigInteger.ZERO) == 0)

gcd = k;

}

return gcd;

}

public BigInteger getNumerator() {

return r[0];

}

public BigInteger getDenominator() {

return r[1];

}

public Rational add(Rational secondRational) {

BigInteger n = (r[0].multiply(secondRational.getDenominator())).add(

r[1].multiply(secondRational.getNumerator()));

BigInteger d = r[1].multiply(secondRational.getDenominator());

return new Rational(n, d);

}

public Rational subtract(Rational secondRational) {

BigInteger n = (r[0].multiply(secondRational.getDenominator())).subtract(

r[1].multiply(secondRational.getNumerator()));

BigInteger d = r[1].multiply(secondRational.getDenominator());

return new Rational(n, d);

}

public Rational multiply(Rational secondRational) {

BigInteger n = r[0].multiply(secondRational.getNumerator());

BigInteger d = r[1].multiply(secondRational.getDenominator());

return new Rational(n, d);

}

public Rational divide(Rational secondRational) {

BigInteger n = r[0].multiply(secondRational.getDenominator());

BigInteger d = r[1].multiply(secondRational.getNumerator());

return new Rational(n, d);

}

public String toString() {

if (r[1].compareTo(BigInteger.ONE) == 0)

return r[0] + "";

else

return r[0] + "/" + r[1];

}

public boolean equals(Object other) {

if (((this.subtract((Rational)(other))).getNumerator()).compareTo(

BigInteger.ZERO) == 0)

return true;

else

return false;

}

public int intValue() {

return (int)doubleValue();

}

public float floatValue() {

return (float)doubleValue();

}

public double doubleValue() {

return this.getNumerator().doubleValue() /

this.getDenominator().doubleValue();

}

public long longValue() {

return (long)doubleValue();

}

public int compareTo(Object o) {

BigInteger zero = BigInteger.ZERO;

BigInteger n = this.subtract((Rational)o).getNumerator();

if (n.compareTo(zero) > 0)

return 1;

else if (n.compareTo(zero) < 0)

return -1;

else

return 0;

}

}

----------------------------------

13.19

import java.io.\*;

import java.math.BigInteger;

import java.util.\*;

public class test {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter a decimal number: ");

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

Rational r1 = new Rational(new BigInteger(decimal[0]), BigInteger.ONE);r

Rational r2 = new Rational(new BigInteger(decimal[1]), new BigInteger(

String.valueOf((int)Math.pow(10, decimal[1].length()))));

System.out.println("The fraction number is " +

(decimal[0].charAt(0) == '-' ? (r1).subtract(r2) : (r1).add(r2)));

}

}

class Rational extends Number implements Comparable {

private BigInteger[] r = new BigInteger[2];

public Rational() {

this(new BigInteger("0"), new BigInteger("1"));

}

public Rational(BigInteger numerator, BigInteger denominator) {

BigInteger gcd = gcd(numerator, denominator);

r[0] = (denominator.compareTo(BigInteger.ZERO) > 0

? BigInteger.ONE :

new BigInteger("-1")).multiply(numerator.divide(gcd));

r[1] = denominator.divide(gcd);

}

private static BigInteger gcd(BigInteger n, BigInteger d) {

BigInteger n1 = n;

BigInteger n2 = d;

BigInteger gcd = BigInteger.ONE;

for (BigInteger k = BigInteger.ONE;

k.compareTo(n1) <= 0 && k.compareTo(n2) <= 0;

k = k.add(BigInteger.ONE)) {

if (n1.remainder(k).compareTo(BigInteger.ZERO) == 0 &&

n2.remainder(k).compareTo(BigInteger.ZERO) == 0)

gcd = k;

}

return gcd;

}

public BigInteger getNumerator() {

return r[0];

}

public BigInteger getDenominator() {

return r[1];

}

public Rational add(Rational secondRational) {

BigInteger n = (r[0].multiply(secondRational.getDenominator())).add(

r[1].multiply(secondRational.getNumerator()));

BigInteger d = r[1].multiply(secondRational.getDenominator());

return new Rational(n, d);

}

public Rational subtract(Rational secondRational) {

BigInteger n = (r[0].multiply(secondRational.getDenominator())).subtract(

r[1].multiply(secondRational.getNumerator()));

BigInteger d = r[1].multiply(secondRational.getDenominator());

return new Rational(n, d);

}

public Rational multiply(Rational secondRational) {

BigInteger n = r[0].multiply(secondRational.getNumerator());

BigInteger d = r[1].multiply(secondRational.getDenominator());

return new Rational(n, d);

}

public Rational divide(Rational secondRational) {

BigInteger n = r[0].multiply(secondRational.getDenominator());

BigInteger d = r[1].multiply(secondRational.getNumerator());

return new Rational(n, d);

}

public String toString() {

if (r[1].compareTo(BigInteger.ONE) == 0)

return r[0] + "";

else

return r[0] + "/" + r[1];

}

public boolean equals(Object other) {

if (((this.subtract((Rational)(other))).getNumerator()).compareTo(

BigInteger.ZERO) == 0)

return true;

else

return false;

}

public int intValue() {

return (int)doubleValue();

}

public float floatValue() {

return (float)doubleValue();

}

public double doubleValue() {

return this.getNumerator().doubleValue() /

this.getDenominator().doubleValue();

}

public long longValue() {

return (long)doubleValue();

}

public int compareTo(Object o) {

BigInteger zero = BigInteger.ZERO;

BigInteger n = this.subtract((Rational)o).getNumerator();

if (n.compareTo(zero) > 0)

return 1;

else if (n.compareTo(zero) < 0)

return -1;

else

return 0;

}

}