Isaac Ray Shoebottom

CS 1073 (FR02A)

Assignment 7

3429069

# Section A

## Source Code (Main.java):

/\*\*

\* This class drives the program, it controls the console input/output and passes arguments given to the classes

\* @author Isaac Shoebottom (3429069)

\*/

public class Main {

public static void main(String[] args) {

byte userChoice;

double biggestCorralArea = 0;

String biggestCorralType = "not applicable";

java.util.Scanner scan = new java.util.Scanner(System.in);

do {

System.out.print(

"What would you like to do?\n" +

"1 - Get info for rectangular enclosure\n" +

"2 - Get info for polygon enclosure\n" +

"3 - Quit\n" +

"Enter your choice: ");

userChoice = scan.nextByte();

if (userChoice == 1) {

System.out.print("Width in meters: ");

double tempWidth = scan.nextDouble();

System.out.print("Length in meters: ");

double tempLength = scan.nextDouble();

RectangularCorral corral = new RectangularCorral(tempWidth, tempLength);

System.out.print("The area is: "); System.out.printf("%.3f", corral.getArea()); System.out.print(" square meters\n");

System.out.print("The cost is: "); System.out.printf("%.2f", corral.getTotalFenceCost()); System.out.print("$\n");

if (biggestCorralArea < corral.getArea()) {

biggestCorralArea = corral.getArea();

biggestCorralType = "rectangle";

}

}

if (userChoice == 2) {

System.out.print("Length of sides: ");

double tempLength = scan.nextDouble();

System.out.print("Number of sides: ");

long tempSides = scan.nextLong();

PolygonalCorral corral = new PolygonalCorral(tempLength, tempSides);

System.out.print("The area is: "); System.out.printf("%.3f", corral.getArea()); System.out.print(" square meters\n");

System.out.print("The cost is: "); System.out.printf("%.2f", corral.getTotalFenceCost()); System.out.print("$\n");

if (biggestCorralArea < corral.getArea()) {

biggestCorralArea = corral.getArea();

biggestCorralType = "polygon";

}

}

} while (userChoice != 3);

System.out.println("The corral with the largest area is a " + biggestCorralType);

System.out.print("It's area is : "); System.out.printf("%.3f", biggestCorralArea); System.out.print(" square meters");

}

}

## Source Code (PolygonalCorral.java):

/\*\*

\* This class describes a polygonal corral with each side of equal length. It takes a length and a number of sides.

\* @author Isaac Shoebottom (3429069)

\*/

public class PolygonalCorral {

/\*\*

\* The unit price is how much the fence costs per meter

\*/

final double unitPrice = 9.50;

/\*\*

\* The length is how long each side of the polygonal corral is in meters

\*/

double length;

/\*\*

\* The number of sides in the polygonal corral

\*/

long numberOfSides;

/\*\*

\* The polygonal corral method contains the length and number of sides

\* @param length The length of the sides of the polygonal corral in meters

\* @param numberOfSides The number of sides of the polygonal corral

\*/

PolygonalCorral (double length, long numberOfSides) {

this.length = length;

this.numberOfSides = numberOfSides;

}

/\*\*

\* Method to get the length of the polygonal corrals sides

\* @return The length of the corrals sides in meters

\*/

public double getLength() {

return length;

}

/\*\*

\* Method to get the number of sides of the polygonal corral

\* @return The number of sides of the polygonal corral

\*/

public long getNumberOfSides() {

return numberOfSides;

}

/\*\*

\* Method to get the unit price of a meter of fence

\* @return The price of a meter of fence

\*/

public double getUnitPrice() {

return unitPrice;

}

/\*\*

\* Method to get the total cost of the polygonal fence

\* @return The cost of the polygonal fence

\*/

public double getTotalFenceCost() {

return (length\*numberOfSides\*unitPrice);

}

/\*\*

\* Method to get the area of the polygonal corral

\* @return The area of the polygonal corral in meters squared

\*/

public double getArea() {

double radians = (180/(double)numberOfSides)\*(Math.PI/180);

double apothem = length/(2\*Math.tan(radians));

return (0.5\*(length\*numberOfSides)\*apothem);

}

}

## Source Code (RectangularCorral.java):

/\*\*

\* This class describes a rectangular corral with a width and length

\* @author Isaac Shoebottom (3429069)

\*/

public class RectangularCorral {

/\*\*

\* The width of the rectangular corral

\*/

double width;

/\*\*

\* The length of the rectangular corral

\*/

double length;

/\*\*

\* The price of fence per meter

\*/

final double unitPrice = 9.50;

/\*\*

\* The rectangular corral method contains the width and the height of the corral

\* @param width The width of the rectangular corral

\* @param length The length of the rectangular corral

\*/

RectangularCorral (double width, double length) {

this.width = width;

this.length = length;

}

/\*\*

\* Method to get the length of the rectangular corral

\* @return The length of the rectangular corral

\*/

public double getLength() {

return length;

}

/\*\*

\* Method to get the width of the rectangular corral

\* @return The width of the rectangular corral

\*/

public double getWidth() {

return width;

}

/\*\*

\* Method to get the price of fence per meter

\* @return The price of fence per meter

\*/

public double getUnitPrice() {

return unitPrice;

}

/\*\*

\* Method to get the total cost of the rectangular fence

\* @return The total cost of the rectangular fence

\*/

public double getTotalFenceCost() {

return ((length+width)\*2\*unitPrice);

}

/\*\*

\* Method to get the area of a rectangular corral

\* @return The area of a rectangular corral

\*/

public double getArea() {

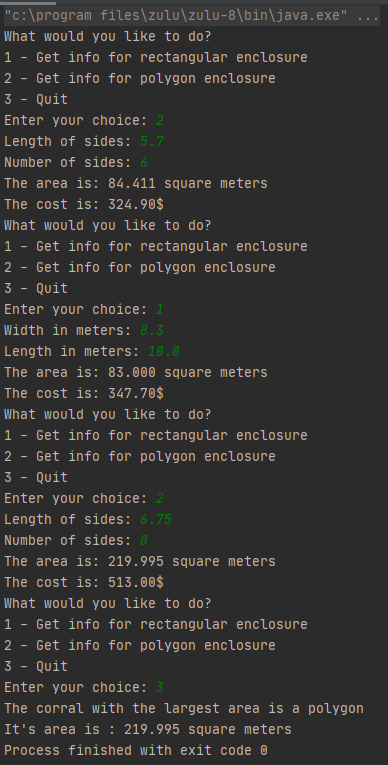
return (length\*width);

}

}

# Section B

## Output:



# Section C

## Source Code (Main.java):

/\*\*

\* This class has two methods, one for returning a constructed string from the hieroglyph and one that drives the console input and output

\* @author Isaac Shoebottom (3429069)

\*/

public class Main {

public static void main(String[] args) {

java.util.Scanner scan = new java.util.Scanner(System.in);

int inputNumber;

do {

System.out.print("Please enter a number between 1 and 9 999 999: ");

inputNumber = scan.nextInt();

if (inputNumber < 1 | inputNumber > 9\_999\_999) {

System.out.println("Invalid input. You must enter a number between 1 and 9 999 999");

}

} while (inputNumber < 1 | inputNumber > 9\_999\_999);

System.out.println(inputNumber + " in Egyptian hieroglyphs is:");

System.out.print(printHieroglyphics(inputNumber/1\_000\_000, 'w')); inputNumber %= 1\_000\_000;

System.out.print(printHieroglyphics(inputNumber/100\_000, '&')); inputNumber %= 100\_000;

System.out.print(printHieroglyphics(inputNumber/10\_000, ')')); inputNumber %= 10\_000;

System.out.print(printHieroglyphics(inputNumber/1\_000, '\*')); inputNumber %= 1\_000;

System.out.print(printHieroglyphics(inputNumber/100, '@')); inputNumber %= 100;

System.out.print(printHieroglyphics(inputNumber/10, 'n')); inputNumber %= 10;

System.out.print(printHieroglyphics(inputNumber, '|'));

}

private static String printHieroglyphics(int number, char hieroglyph) {

if (number == 0 ) { return ""; }

byte counter = 0;

StringBuilder phrase = new StringBuilder(String.valueOf(hieroglyph));

if (number == 4 | number == 7 | number == 8) {

for (int i = number; i > 1; i--) {

counter++;

if (counter % 4 == 0) { phrase.append("\n"); }

phrase.append(hieroglyph);

}

} else {

for (int i = number; i > 1; i--) {

counter++;

if (counter % 3 == 0) { phrase.append("\n"); }

phrase.append(hieroglyph);

}

}

return (phrase.toString() + "\n");

}

}

# Section D

## Output: