THOMPSON RIVERS UNIVERSITY

COMP 1130
Computer Programming I
Final Exam Assignment B – 25 marks

Instructions

- questions for students with student numbers ending in **5, 6, 7, 8, or 9**
  - *if the incorrect assignment is submitted, this is a mark of zero (0)*
  - *if the submission is late, this is a mark of zero (0)*
  - *you can use previous exercises & solutions, lectures notes, and the internet…but* **the work is your own!**

- complete the questions below, with marks distributed against,
  - *solving the problem with a program that is efficient and well-structured, and formatted*
  - *proper & adequate documentation (program header & comments), and well-named identifiers*
  - *clear & informative user interaction (clear prompts)*
  - *sufficient output capture showing/proving the program solves the problem with various tests*

- questions have multiple parts, and full marks given when all parts are included in the solution
  (partial marks given for parts completed…*output capture still required to show functionality for completed work)*

[10 marks] Question 1

You have been asked to help in the development of an Education Geometry Studies application, to assist elementary and high-school students studying geometric shapes. Your part is to write a Java class to store the characteristics of the fundamental shapes: points, rectangles, right-angle triangles, and circles, to be used by your peer programmers writing other parts of the application.

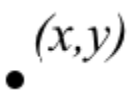
Part 1: Create a new class file called **Shape**, that contains the members for a geometric shape: point, rectangle, triangle, or circle. Include all the proper techniques in creating a class.

The class has the following members (attributes and methods),

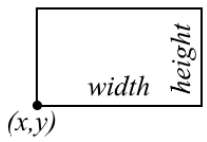
- private attributes (all numbers are **doubles**, unless otherwise indicated),
  - **name** – String, a custom name; such as “Tammy Triangle” or “Bubble the Circle” or “Bob”; default “no name”
  - **figure** – int, the type of shape/figure: 0-point, 1-rectangle, 2-triangle, 3-circle; default is 0 (*point*)
  - **x** – x coordinate for the shape of (x,y); default is zero (0)
  - **y** – y coordinate for the shape of (x,y); default is zero (0)
  - **width –** considered width of rectangle, base length of triangle, or diameter of circle; default is zero (0)
  - **height –** height of the shape; default is zero (0)
    - (x,y) and width, height are custom to the shape:
      - point uses values (x, y) – *the position of the point; width and height are zero*
      - rectangle uses values (x, y) and width, height – *(x,y) is bottom-left of shape*
      - triangle uses values (x,y) and width, height – *(x,y) is bottom-left, and width is base length, and height is regular height*
      - circle uses values (x1, y1) and width – *(x,y) is centre of circle, width is diameter, height is zero*

The shapes are assumed to represent:

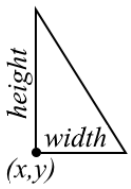
point



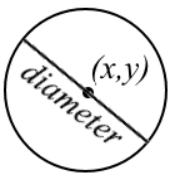
rectangle



right-angle triangle



circle



- public methods *(you may add other methods, that you consider helpful)*:
    - default constructor **Shape()** – that does nothing, since the attributes should already have default values
    - multiple constructors **Shape()** with parameters depending on the type of shape being defined
        - Shape(*name, figure, x, y*) – for figure point; if figure is not point, do nothing
        - Shape(*name, figure, x, y, width*) – for figure, circle; if figure not circle, do nothing
        - Shape(*name, figure, x, y, width, height*) – used for figure rectangle and triangle; if not rectangle or triangle, do nothing
    - multiple set() methods, (hint: could called by constructors)
        - set(*x, y*) – set (x,y) of shape
        - set(*x, y, width*) – set (x,y) and width
        - set(*x, y, width, height*) – set (x,y), width, and height
            - note: there are no methods to change attributes *name* and *figure*
    - **area()** – returns the area calculation of the shape,
        - point – no area, return zero
        - rectangle – return (width * height)
        - circle – return (π*(diameter/2)^2)
        - triangle – return ((base * height)/2)
    - **perimeter()** – returns the perimeter of rectangles and triangles, or circumference of circles,
        - point – no perimeter, return zero
        - rectangle – return ((width + height) * 2)
        - circle – return (π*diameter)
        - triangle – return (base + height + sqrt[base^2 + height^2]) ← this is Pythagoras' Theorem
    - **equals()** – returns a boolean (true/false) if the two objects are "equal,"
        - equality is evaluated only on: *figure* and *area*
          for example: *two rectangles are the same because they are rectangles and have the same area, even if the (x,y) are different, and the width, height values are swapped—since area calc. is the same*
    - **toString()** – returns a formatted description of the object
        - examples: "Tammy the Triangle is a triangle at (2.3,-4.005) with a diameter of 22.71"
          or "Bob is a rectangle at (10.0, 12.0) with a width of 100.5 and height of 55.0"

    - a *switch* statement on *figure* may be very helpful in the various methods, to determine which calculations to perform or which formatted string to return

Part 3: Create a program class to test the Shape class, by declaring a few objects of each type (points, rectangles, triangles, circles), changing their characters with the set() methods, calculating the areas and perimeters, and comparing objects.

Note: For area and perimeter calculations, ensure the answer is correct; don't just call the method and hope it is correct!

Have fun with this program, but make sure it is a well-written and well-prepared program.

Hints:

- consider the Point and Circle classes as examples for calling the various methods
- when writing the test main(), add sufficient display statements so the output is clear without having to refer to the main()

[10 marks] Question 2

For an astronomy course, you decide to write a program to produce a summary report of the solar system (yes, *Pluto is a planet, dammit!*). Using the concept of parallel arrays, and your knowledge of Java flow-of-control and methods, accomplish the following.

Part 1: Add to your program,

Include the following array declarations, **planet[]** and **orbitYR[]**, and initialisations,

```
  // name of planet
String[] planet  ={ "Mercury","Venus","Earth","Mars","Jupiter","Saturn","Uranus","Neptune","Pluto" };
  // diameter in kilometres
int[]    diamKM  ={ 4878,      12104,  12760,  6787,  139822,   120500,  51120,   49530,    2301  };
  // orbit time (in Earth years)
double[] orbitYR ={ 0.241,     0.616,  1.0,    1.88,  11.9,     29.5,    84.0,    165.0,    248.0 };
```

- declare a new **int** array **orbitDays[ ]** using the .length attribute of the **orbitYR[ ]** array
- populate the new array with the values from the orbitYR[ ] array, to convert element values from years to days (you will have to correctly cast to int),
    - `orbit in days = (orbit in years * 365)`

Part 2: using the **orbitDays[ ]** array and **planet[ ]** array, determine,

- the planet with minimum orbit, store to new identifiers: **minOrbit** and **nameMinOrbit**
- the planet with maximum orbit, store to new identifiers: **maxOrbit** and **nameMaxOrbit**

    - write a new method **indexMin()** that has a single double array parameter, and returns an int
    - the method returns the index of the maximum element value in the array
    - call this method, passing the orbitDays[] array, and use the returned index to help identify the element in orbitDays[] and planet[]

    - write a similar method **indexMax()**, that returns the index to help identify the element in orbitDays[] and planet[]

Part 3: using the outcomes from the above, to complete the program so it shows the summary report
(see sample report format below, and use all your knowledge of .printf() and String.format())

- display the *orbit in days*; ex: Earth orbit is 365 days, Mars orbit is 686 days
- planet name is shown in UPPERCASE
- diameter is shown in 1000's of kms in format 7.2f (2 decimal places); ex: Mercury's diam is `4.88`

Send this report to a file called **PlanetSummary.txt**.

The summary report format:

```
Planet       Diam(1000kms)    Orbit(days)
-----------------------------------------
MERCURY          4.88             87
VENUS           12.10             224
.
.
.
PLUTO            2.30             90520
-----------------------------------------
```

[5 marks] Question 3

Throughout the course, you have been asked to write specific programs or solve specific problems.
This is your opportunity to be creative and discuss something you found interesting in the course, but were not asked to complete in an exercise, assignment, or exam.

Write a complete well-written, well-documented, and personal program that shows a topic or concept you found intriguing. This is your program, and not something copied from the lecture notes or internet (which is easy to check with a google search, so no cheating!)

In your output capture document, write a brief statement (two or three sentences), as to why this topic is interesting for you.