1 The Circle Class

Download the `Circle` class and the `CircleApplication`. Complete the main method by creating two circle objects and calculate and print their areas and circumferences.

2 The Rectangle Class

Design and implement a class named `Rectangle` to represent a rectangle. The class contains:
Two double data **fields** named `width` and `height` that specify the width and height of the rectangle. The default values are 1 for both width and height.

- A no-argument constructor that initializes the width and height to 1.0.
- A constructor that creates a rectangle with the width and height specified by arguments to the constructor.
- The accessor (get methods) and mutator (set methods) methods for all data fields.
- A method named `getArea()` that returns the area of this rectangle.
- A method named `getPerimeter()` that returns the perimeter of this rectangle.

2.1 Test the Rectangle class

Write a program called `RectangleTester` that tests all the methods in the `RectangleTester` class. Your program should create at least two `Rectangle` objects and calls all the methods.

3 The MyPoint class

Design and implement a class named `MyPoint` to represent a point in the Cartesian plane. Each point is represented with an x- and a y-coordinate. The class contains:

- Two data fields `x` and `y` that represent the coordinates.
- A no-argument constructor that creates a point with coordinates `(0, 0)`.
- A constructor that constructs a point with coordinates specified by arguments to the constructor.
- Two get methods for the data fields `x` and `y`, respectively.
- A method named `distance` that returns the distance from this point to another point of the `MyPoint` type.
- A method named `distance` that returns the distance from this point to another point with specified `x` and `y`-coordinates.

3.1 Test the MyPoint class

Write a program called `PointTester` that tests all the methods in the `MyPoint` class. Your program should create at least two `MyPoint` objects and call all the methods.

4 A Time class

Write a class `Time` that maintains a time value using a 24 hour clock. The class stores a time as hours and minutes (e.g. the time $13:45$ is stored as 13 and 45). Use the design shown on the next page

4.1 Using the Time class

Write an application (A class with a main method) that allows the user to calculate the amount of money owed to worker who is paid by the hour (or part thereof). The user inputs

- the time the worker started working and
- the time he stopped working
- the hourly rate of pay

Your program then outputs the amount owing to the worker.

All times are entered in 24 hour format e.g. 13:26 is entered by the use as two separate integer: 13 and 26.

You must use the `Time` to manipulate all times.

Design of the Time class

```
class Time


//fields
int hour  //the hour (0-23)
int min   //the minutes past the hour (0-59)

//constructors
Time() //sets the time to midnight
Time(int h, int m) //sets the time to the specified
                   //hour and minute
Time(int m) //sets the time given the minutes after midnight (m).
            //e.g. for m = 125 the time is 2:05

//methods
int getHours() //returns the hour
int getMins()  //returns the minutes

void getTime12() //prints out the time in 12 hour format
                 //e.g. 1:05 pm
void getTime24() //prints out the time in 24 hour format
                 //e.g. 13 H 05

int GetTotalMins() // returns the time  as a number of mins after
                   //midnight e.g. returns 125 if time is 2:05

void setTime(int h, int m) //sets the time to the specified values

boolean isLater(myTime aTime) //checks whether the Time aTime
                              //is later than this time (the time
                              //stored in the fields  )

void update(int mins) //updates the time by mins.  For e.g. if
                      //present time is 2:05 and mins = 65,
                      //then the new time is 3:10.
```