|  |  |
| --- | --- |
| **Programming 2**  Diploma in IT / DS / CSF  Year 1 (2022/23) Semester 2 | Week **7** |
| **2 hours** |
| **Practical 7 : Abstract Classes & Interfaces** | |

**Objectives**

At the end of this practical, the students should be able to:

* Implement Abstract Classes and Interfaces

|  |
| --- |
| **IMPORTANT**   * Create a folder, **week07.** * Create a new Console App project, **Snnnnnnnn\_AbstractShapeApp**, in the **Week07** folder created above *(note:* ***Snnnnnnnn*** *is your Student Number)*. * At the end of the session, copy the folder **Week07** folder (which contains all your work) to PRG2 network folder: **\\ictspace.ict.np.edu.sg\PRG2**. |

**ShapeApp**

1. Based on the Class Element diagram given below, implement the following classes:
2. Shape class
3. Circle class

|  |
| --- |
| ***Shape*** |
| -type:string  -color:string |
| +Shape()  +Shape(string,string)  *+FindArea():double*  *+FindPerimeter():double*  +ToString():string |

|  |
| --- |
| **Circle** |
| -radius:double |
| +Circle()  +Circle(string,double)  +FindArea():double  +FindPerimeter():double  +ToString():string |

1. In the **Program.cs**, do the following:
2. Create a List, **circleList**, to store Circle objects
3. Write an **InitCircleList(List<Circle> cList)** method to
4. Create 3 **Circle** objects with the information given below.

|  |  |  |
| --- | --- | --- |
|  | color | radius |
| circle1 | Red | 20.0 |
| circle2 | Green | 10.0 |
| 1circle3 | Blue | 30.0 |

1. Add the 3 Circle objects to the **cList**.
2. Call the **InitCircleList**(**circleList**) method in the program.
3. Display a simple menu as shown below and perform task according to the selected option until user chooses to exit.

|  |
| --- |
| ---------------- M E N U -------------------- [1] List all the circles [2] Display the areas of the circles  [3] Display the perimeters of the circles  [4] Change the size of a circle  [5] Add a new circle  [6] Delete a circle  [7] Display circles sorted by area  [0] Exit --------------------------------------------- Enter your option : |

Option 1: Display the details of all Circle objects in **circleList**. Display an error message if the list is empty.

|  |
| --- |
| Enter your option: 1  [1] Type: Circle Color: Red Radius: 20  [2] Type: Circle Color: Green Radius: 10  [3] Type: Circle Color: Blue Radius: 30 |

Option 2: Display the details and area of all Circle objects in **circleList**.

|  |
| --- |
| Enter your option: 2  Type: Circle Color: Red Radius: 20 Area: 1256.64  Type: Circle Color: Green Radius: 10 Area: 314.16  Type: Circle Color: Blue Radius: 30 Area: 2827.43 |

Option 3: Display the details and perimeters of all Circle objects in **circleList**.

|  |
| --- |
| Enter your option: 3  Type: Circle Color: Red Radius: 20 Perimeter: 125.66  Type: Circle Color: Green Radius: 10 Perimeter: 62.83  Type: Circle Color: Blue Radius: 30 Perimeter: 188.50 |

Option 4: Change the size of a circle, allow user to pick which circle to modify the radius.

|  |
| --- |
| Enter your option: 4  [1] Type: Circle Color: Red Radius: 20  [2] Type: Circle Color: Green Radius: 10  [3] Type: Circle Color: Blue Radius: 30  Enter circle number: 2  Enter new radius: 20  Radius successfully changed. |

Option 5: Add a new circle.

|  |
| --- |
| Enter your option: 5  Circle color: Black  Circle radius: 15  New Black circle with radius 15cm added. |

Option 6: Delete a circle.

|  |
| --- |
| Enter your option: 6  [1] Type: Circle Color: Red Radius: 20  [2] Type: Circle Color: Green Radius: 10  [3] Type: Circle Color: Blue Radius: 30  Enter circle number: 2  Circle removed. |

Option 7: Display circles sorted by area

|  |
| --- |
| Enter your option: 7  Type: Circle Color: Green Radius: 10 Area: 314.16  Type: Circle Color: Red Radius: 20 Area: 1256.64  Type: Circle Color: Blue Radius: 30 Area: 2827.43 |

**Hint:**

Modify the **Circle** class to implement the **interface** below so that the shapes can be sorted based on its size (i.e. radius or area).

|  |
| --- |
| interface IComparable<T>  {  int CompareTo(T obj);  } |

The return value of CompareTo() method is:

1 if the object is bigger than the other object

0 if they are the same

-1 if the other object is bigger

**EmployeeApp (Advanced)**

The class diagram for an application is given below.

|  |
| --- |
| ***Employee*** |
| -id:int  -name:string |
| +Employee()  +Employee(int,string)  *+CalculatePay():double*  +ToString():string |

|  |  |  |
| --- | --- | --- |
| **FullTimeEmployee** |  | **PartTimeEmployee** |
| -basicPay:double  -allowance:double |  | -dailyRate:double  -daysWorked:int |
| +FullTimeEmployee()  +FullTimeEmployee(int,string,double,double)  +CalculatePay():double  +ToString():string |  | +PartTimeEmployee()  +PartTimeEmployee(int,string,double,int)  +CalculatePay():double  +ToString():string |

* The (monthly) pay of a full time employee is calculated as shown below:

pay = basic pay + allowance

* The (monthly) pay of a part-time employee is calculated as shown below:

pay = days worked x daily rate

1. Create a Console project **Snnnnnnnn\_EmployeeApp** in Visual Studio 2022.
2. Implement the **Employee** class.
3. Implement the **FullTimeEmployee** class.
4. Implement the **PartTimeEmployee** class.
5. Write code in the superclass to implement the interface IComparable<T> so that the program can sort objects according to id.
6. Write code in the Program.cs to do the following:
7. create an employeeList to store FullTimeEmployee and PartTimeEmployee objects. Using the information below, create 2 objects and add them to employeeList.

|  |  |  |
| --- | --- | --- |
| employee1 | FullTimeEmployee | Id: 103  Name: John  Basic pay: 1500  Allowance: 100 |
| employee2 | PartTimeEmployee | Id: 101  Name: Mary  Daily rate: 50  Days worked: 10 |

1. sort employeeList by id.
2. display all the objects in employeeList using the ToString() method (using a for or foreach loop).

**Plagiarism Warning:**

**If a student is found to have submitted work not done by him/her, he/she will not be awarded any marks for this practical. Disciplinary action may also be taken.**

**Similar action will be taken for student who allows other student(s) to copy his/her work.**