# Week 3: Lab exercise

**Aim: Learn basic C# class syntax, adding fields, methods, constructor, properties, static members**

Note: You may choose to work in pairs. Please show completed task1, task 2, task 3 to your lab instructor before you leave (not applicable for Tuesday group as they haven't had the lecture yet).

**Task 1:**

1. Create a console project and name it a circle_demo
2. Add a class(c#) to the project name it as 'circle' (as shown in class)
3. Add following to your class (Circle.cs)

```
class Circle
{
private double radius = 0.0;

        Public double Area()
        {
                return 3.141592 * radius * radius;
        }
}
```

4. In your "main" function (program.cs file) add following: and run the program.

```
Circle c1;
c1= new Circle();
c1.radius =5.0;
Double c1_area= c1.Area();
Console.WriteLine(c1_area);
Console.ReadLine();
```

**You will get error. why? Fix it!**

5. Add a constructor to your class Circle

```
public Circle (double r)
        {
                radius = r;
        }
```

In your main function, add

```
c1= new Circle (5.0);
```

Run the program. Why are you getting '78.53' instead of getting '0'?

6. Add following in your class (Circle.cs)

```
private String name;
public Circle (String n, double r)
{
        name=n;
        radius=r;
}
```

Write code in your main function to call the above constructor. In other words, initialize an object using above constructor.

7. Getting user input: add following in your class “program” (program.cs file)

```
static Circle Create_Circle ()
{
        String name, temp;
        double radius;
        Console.Write("Please enter name for new Circle: ");
        name = Console.ReadLine(); Console.Write("Please
        enter radius: ");
        temp = Console.ReadLine();
        radius = double.Parse(temp);            // why you want to add this?
        return new Circle (name, radius);
}
```

In your main function add:

```
Circle c1 = Create_Circle ();

Console.Write("Circle " + c1.Name());

Console.WriteLine(" created with radius” + c1.Radius());

Console.WriteLine("Its area is " + c1.Area());

Console.ReadLine(); // Keeps window open.
```

To be able to run this code. You need to add Radius () and Name () To be able to run this code. You need to add Radius () and Name () methods (accessor methods). Add these accessor method as discussed in class.

8. Passing objects: add following in your class Circle (circle.cs)

```
public bool Is_Greater_Than (Circle other)
{
        if (this.Radius() > other.Radius())
{
        return true;
}
else
{
        return false;
}
}
```

In your main function: add

```
Circle Circle_A = Create_Circle();
Console.Write ("Circle " + Circle_A.Name() );
Console.WriteLine (" created with radius " + Circle_A.Radius());
Console.WriteLine ("Its area is " + Circle_A.Area());
Circle c2= Create_Circle();
Console.Write ("Circle " + c2.Name() );
Console.WriteLine (" created with radius " + c2.Radius());
Console.WriteLine ("Its area is " + c2.Area());
if (c1.Is_Greater_Than(c2))
{
Console.Write ("Circle " + c1.Name() + " is greater than ");
Console.WriteLine( "Circle " + c2.Name());
}
else if (c2.Is_Greater_Than(c1))
{
Console.Write ("Circle " + c2.Name() + " is greater than ");
Console.WriteLine( "Circle " + c1.Name());
}
```

```
else
{
Console.Write("Circle " + c1.Name() + " and Circle " + c2.Name());
Console.WriteLine (" are the same size.");
}
```

Run your program and find what does ‘this’ keyword do?

9. Review Static fields, methods and class from lecture slides.

**Task 2 (Marked task: 1 pts)**

Implement a class Account. An account has

- a balance field,
- methods to add and withdraw money,
- and a function to inquire the current balance.
- Pass a value into a constructor to set an initial balance.
- If no value is passed the initial balance should be set to $0.
- Add a method to the Account class to compute interest on the current balance.

Implement a class Bank. This bank has two objects

- checking
- and savings

of the type *Account*.

Implement four methods:

- deposit(double amount, String account)
- withdraw(double amount, String account)
- transfer(double amount, String account)
- printBalances()

Write suitable test cases to test above implementation (main method)

**Task 3 (Marked task: 1pts)**

Write a simple Vehicle class that has fields for (at-least) current speed, current direction, and owner name.

- Add a static field to your Vehicle class for the Vehicle Identification Number, and a non-static field that holds each vehicle's ID number.
- Write test cases (main method) for your Vehicle class that creates a few vehicles and prints out their field values.
- Add two constructors to Vehicle. A no-arg constructor and one that takes an initial owner's name. Test your design.
- Make the fields in your Vehicle class private and add properties for the fields. Which fields should have 'set' properties to change them and which should not?
- Add a *changeSpeed* method that changes the current speed of the vehicle to a passed-in value, and a stop method that sets the speed to zero.
- Add two turn methods to Vehicle. One that takes a number of degrees to turn (180 deg), and one that takes simply either a Vehicle.TURN_LEFT or a Vehicle.TURN_RIGHT constant. Define the two constants using const (Read about 'const' in C#).

**Task 4**

work on Email problem if you have not finished already 😊.