8.1P - semester test example

Object-Oriented Programming (Swinburne University of Technology)

Swinburne University of Technology

Object Oriented Programming

Doubtfire Submission

Semester Test

Submitted By:
Raj Negi
101696057
2020/05/08 16:40

Tutor:
Shamara Gibson

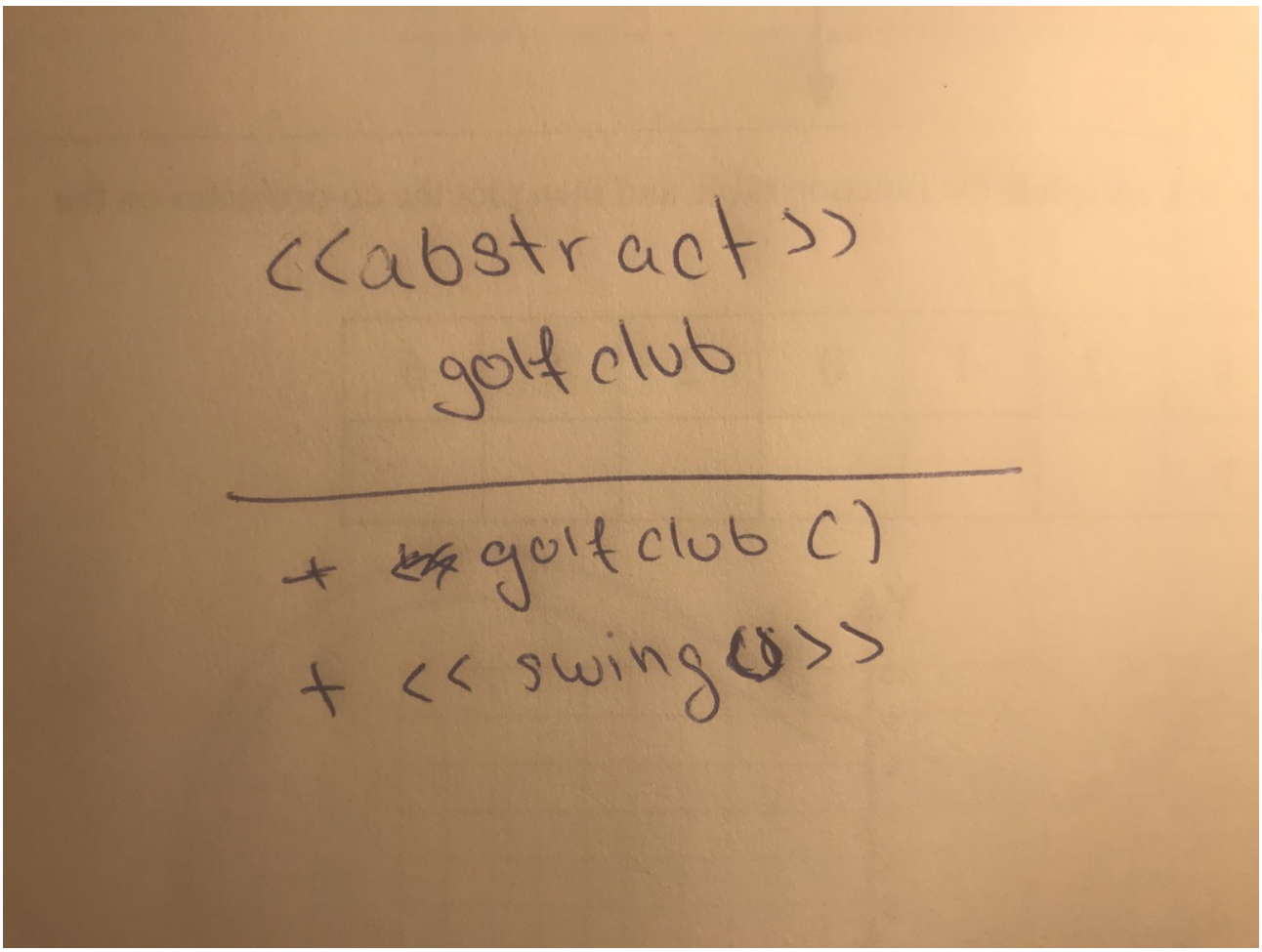
May 8, 2020

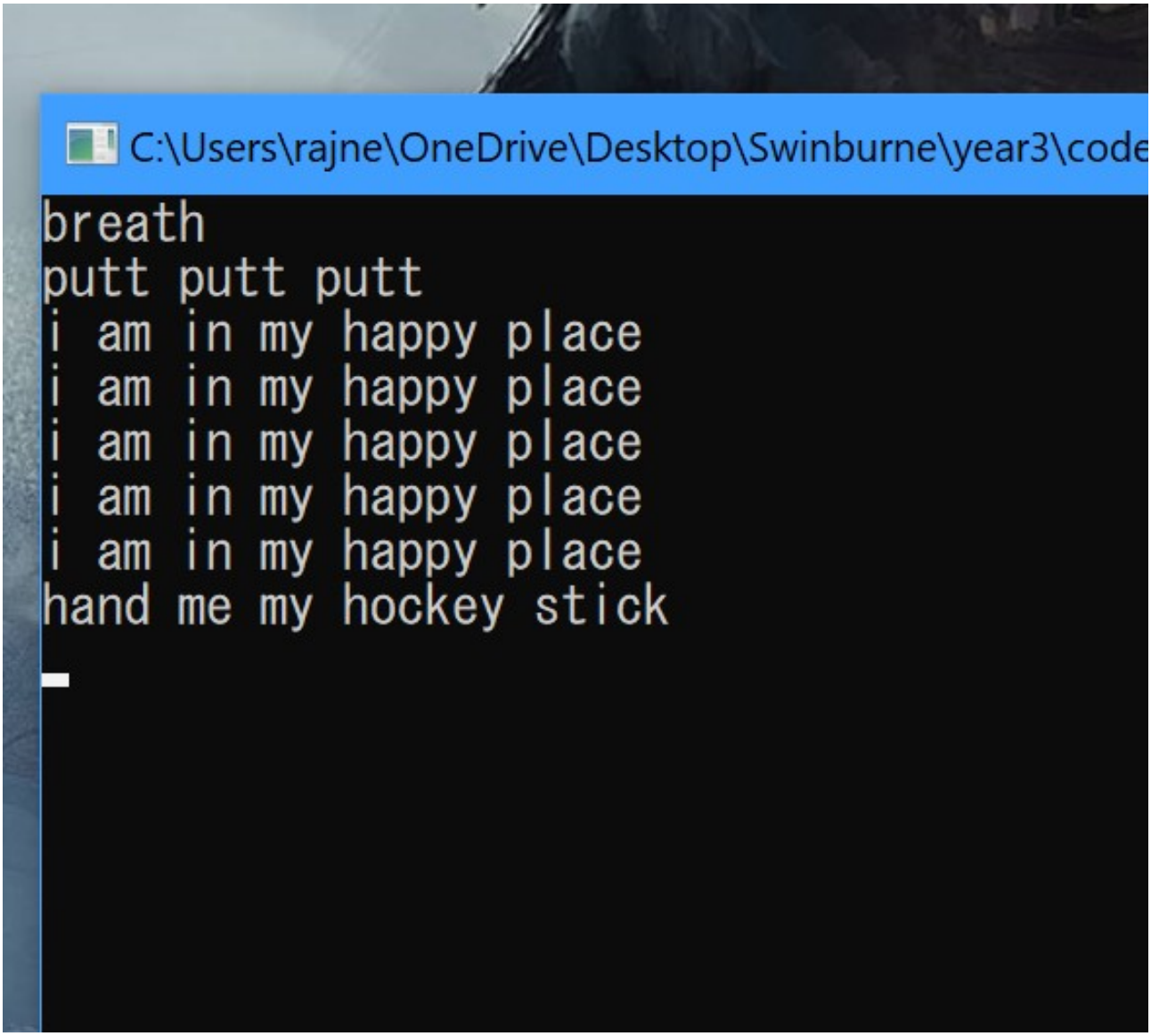
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SemesterTest
{
    public abstract class GolfClub
    {
        public  GolfClub()
        {

        }
        public abstract void Swing();

    }
}
```





- **Encapsulation**
  Prevents data from being accessed by other classes outside its class. This is done by making the fields private inside its class and only allowing the methods in the same class to access it. For other classes to access them they must call public methods
    - Almost all the tasks
      We use (private datatype _dataname) to only all the certain dataname to be accessible by the methods in the same class

- **Abstraction**
  Objects should only expose operations relevant for other objects and hide internal implementation details
    - A shape can have the properties (color, width, height, radius) but the class (circle) only needs to use the property radius and color to define its methods. Abstraction theirfore only provides the information needed to a circle.

- **Inheritance**
  A child class can inherit methods from its parent class and can implement its own objects to it
    - A parent class (shape) can have color, properties which can be used by its child classes (rectangle, circle) to define its own properties

- **Polymorphism**
  Child classes can use a class exactly like its parent, but each child class keeps its own methods.
    - Task 4.1
      We use polymorphism, as the child classes (rectangle, circle,) have their on fields (width, height, radius) but use the methods defined in the parent class (shape)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SemesterTest
{
    class Golfer
    {
       private List<GolfClub> _club;
       public Golfer()
       {
            _club = new List<GolfClub>();
       }

        public bool Holding()
        {
            if (_club.Count()!=0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public void PickUp (GolfClub Club)
        {
            _club.Add(Club);
        }

        public void Swing (GolfClub Club)
        {
            foreach (GolfClub s in _club )
            {
                if (s==Club)
                {
                    s.Swing();
                }
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SemesterTest
{
    class Putter : GolfClub
    {


        public Putter()
        {

        }

        public override void Swing ()
        {
            Console.WriteLine("putt putt putt");
        }




    }


}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SemesterTest
{
   public class SandWedge : GolfClub
    {

        private int _count;

        public SandWedge()
        {
            _count = 5;

        }
        public override void Swing()
        {
            while (_count > 0)
            {
                Console.WriteLine("i am in my happy place ");
                _count -= 1;
            }
            if (_count ==0)
            {
                Console.WriteLine("hand me my hockey stick");
            }
        }



        public void Throw()
        {
            _count = 0;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SemesterTest
{
    class Program
    {
        public static void Main()
        {
            Golfer Raj = new Golfer();
            Putter Club1 = new Putter();
            SandWedge Club2 = new SandWedge();

            Raj.PickUp(Club1);
            Raj.PickUp(Club2);

            if (Raj.Holding()==true)
            {
                Console.WriteLine("breath");
                Raj.Swing(Club1);
                Raj.Swing(Club2);
            }
            else
            {
                Console.WriteLine("where the caddy");
            }

            Console.ReadKey();

        }
    }
}
```