1. Task 1

```
public class BankAccount
{
    2 references
    private string accountNumber;
    3 references
    private double balance;
    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public BankAccount(string accountNumber, double balance)
    {
        this.accountNumber = accountNumber;
        if (balance <= 0)
        {
            Console.WriteLine("Initial balance must be greater than 0.");
        }

        this.balance = balance;
    }
    1 reference
    public string AccountNumber
    {
        get { return accountNumber; }
    }

    7 references
    public double Balance
    {
        get { return balance; }
        private set
        {
            if (value <= 0)
            {
                Console.WriteLine("Balance must always remain greater than 0.");
            }
            balance = value;
        }
    }
```

```
    1 reference | Windsurf: Refactor | Explain | Generate Documentation |
    public void Deposit(double amount)
    {
        if (amount <= 0)
        {
            Console.WriteLine("Deposit amount must be greater than 0.");
            return;
        }

        Balance += amount;
        Console.WriteLine($"Successfully deposited {amount}. New balance: {Balance}");
    }
    1 reference | Windsurf: Refactor | Explain | Generate Documentation |
    public void Withdraw(double amount)
    {
        if (amount <= 0)
        {
            Console.WriteLine("Withdraw amount must be greater than 0.");
            return;
        }

        if (amount > Balance)
        {
            Console.WriteLine("Insufficient balance.");
            return;
        }

        Balance -= amount;
        Console.WriteLine($"Successfully withdrew {amount}. Remaining balance: {Balance}");
    }
}
```

```
0 references | Windsurf: Refactor | Explain
class Program
{
    0 references | Windsurf: Refactor | Explain | Generate Documentation |
    static void Main(string[] args)
    {
        BankAccount account = new BankAccount("ACC12345", 5000);
        Console.WriteLine($"Account Number: {account.AccountNumber}");
        Console.WriteLine($"Initial Balance: {account.Balance}");
        account.Deposit(1500);
        account.Withdraw(2000);
        Console.WriteLine($"Final Balance: {account.Balance}");
    }
}
```

Output:

```
Account Number: ACC12345
Initial Balance: 5000
Successfully deposited 1500. New balance: 6500
Successfully withdrew 2000. Remaining balance: 4500
Final Balance: 4500
PS D:\College\Application Development\Week 4\Workshop5>
```

2. Task 2

```
2 references | Windsurf: Refactor | Explain
public class Vehicle
{
    5 references
    public string Brand { get; set; }
    3 references
    public int Speed { get; set; }

    2 references | Windsurf: Refactor | Explain | Generate Documentation |
    public void Start()
    {
        Console.WriteLine($"{Brand} is starting...");
    }

    2 references | Windsurf: Refactor | Explain | Generate Documentation |
    public void Stop()
    {
        Console.WriteLine($"{Brand} is stopping...");
    }

    6 references | Windsurf: Refactor | Explain | Generate Documentation |
    public virtual void DisplayInfo()
    {
        Console.WriteLine($"Brand: {Brand}");
        Console.WriteLine($"Speed: {Speed} km/h");
    }
}

2 references | Windsurf: Refactor | Explain
public class Car : Vehicle
{
    2 references
    public int Seats { get; set; }

    4 references | Windsurf: Refactor | Explain | Generate Documentation |
    public override void DisplayInfo()
    {
        base.DisplayInfo();
        Console.WriteLine($"Seats: {Seats}");
    }
}
```

```
2 references | Windsurf: Refactor | Explain
public class Motorcycle : Vehicle
{
    2 references
    public bool HasCarrier { get; set; }

    4 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void DisplayInfo()
    {
        base.DisplayInfo();
        Console.WriteLine($"Has Carrier: {HasCarrier}");
    }
}

0 references | Windsurf: Refactor | Explain
class Program
{
    0 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    static void Main(string[] args)
    {
        Car car = new Car();
        car.Brand = "Toyota";
        car.Speed = 180;
        car.Seats = 5;

        Motorcycle bike = new Motorcycle();
        bike.Brand = "Yamaha";
        bike.Speed = 120;
        bike.HasCarrier = true;

        Console.WriteLine("<<< Car Info >>>");
        car.Start();
        car.DisplayInfo();
        car.Stop();

        Console.WriteLine("<<< Motorcycle Info >>>");
        bike.Start();
        bike.DisplayInfo();
        bike.Stop();
    }
}
```

Output:

```
<<< Car Info >>>
Toyota is starting...
Brand: Toyota
Speed: 180 km/h
Seats: 5
Toyota is stopping...
<<< Motorcycle Info >>>
Yamaha is starting...
Brand: Yamaha
Speed: 120 km/h
Has Carrier: True
Yamaha is stopping...
PS D:\College\Application Development\Week 4\Workshop5>
```

3. Task 3

3.1.

```
class Printer
{
    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void Print(string message)
    {
        Console.WriteLine(message);
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void Print(int number)
    {
        Console.WriteLine(number);
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void Print(string message, int count)
    {
        for (int i = 0; i < count; i++)
        {
            Console.WriteLine(message);
        }
    }
}

0 references | Windsurf: Refactor | Explain
class Program
{
    0 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    static void Main(string[] args)
    {
        Printer printer = new Printer();
        printer.Print("Hello, World!");
        printer.Print(42);
        printer.Print("Repeat this message", 3);
    }
}
```

Output:


PS D:\College\Application Development\Week 4\Workshop5> dotnet run
Hello, World!
42
Repeat this message
Repeat this message
Repeat this message
PS D:\College\Application Development\Week 4\Workshop5>

3.2.

```
public class Teacher
{
    2 references
    public string Name { get; set; }

    3 references | Windsurf: Refactor | Explain | Generate Documentation |
    public virtual void Teaching()
    {
        Console.WriteLine("Teacher teaches in English");
    }

    2 references | Windsurf: Refactor | Explain | Generate Documentation |
    public void SalaryInfo()
    {
        Console.WriteLine("Salary information cannot be overridden.");
    }
}

1 reference | Windsurf: Refactor | Explain
public class NepaliTeacher : Teacher
{
    3 references | Windsurf: Refactor | Explain | Generate Documentation |
    public override void Teaching()
    {
        Console.WriteLine("Nepali Teacher teaches in Nepali language");
    }
}


1 reference | Windsurf: Refactor | Explain
public class EnglishTeacher : Teacher
{
    // No overriding method needed
}
```

```
0 references | Windsurf: Refactor | Explain
class Program
{
    0 references | Windsurf: Refactor | Explain | Generate Documentation |
    static void Main(string[] args)
    {
        Teacher t1 = new NepaliTeacher();
        t1.Name = "Mr. Sharma";

        Teacher t2 = new EnglishTeacher();
        t2.Name = "Mr. Jonsen";

        Console.WriteLine("<<< Nepali Teacher >>>");
        t1.Teaching();
        t1.SalaryInfo();

        Console.WriteLine("<<< English Teacher >>>");
        t2.Teaching();
        t2.SalaryInfo();
    }
}
```

Output:

```
<<< Nepali Teacher >>>
Nepali Teacher teaches in Nepali language
Salary information cannot be overridden.
<<< English Teacher >>>
Teacher teaches in English
Salary information cannot be overridden.
PS D:\College\Application Development\Week 4\Workshop5>
```

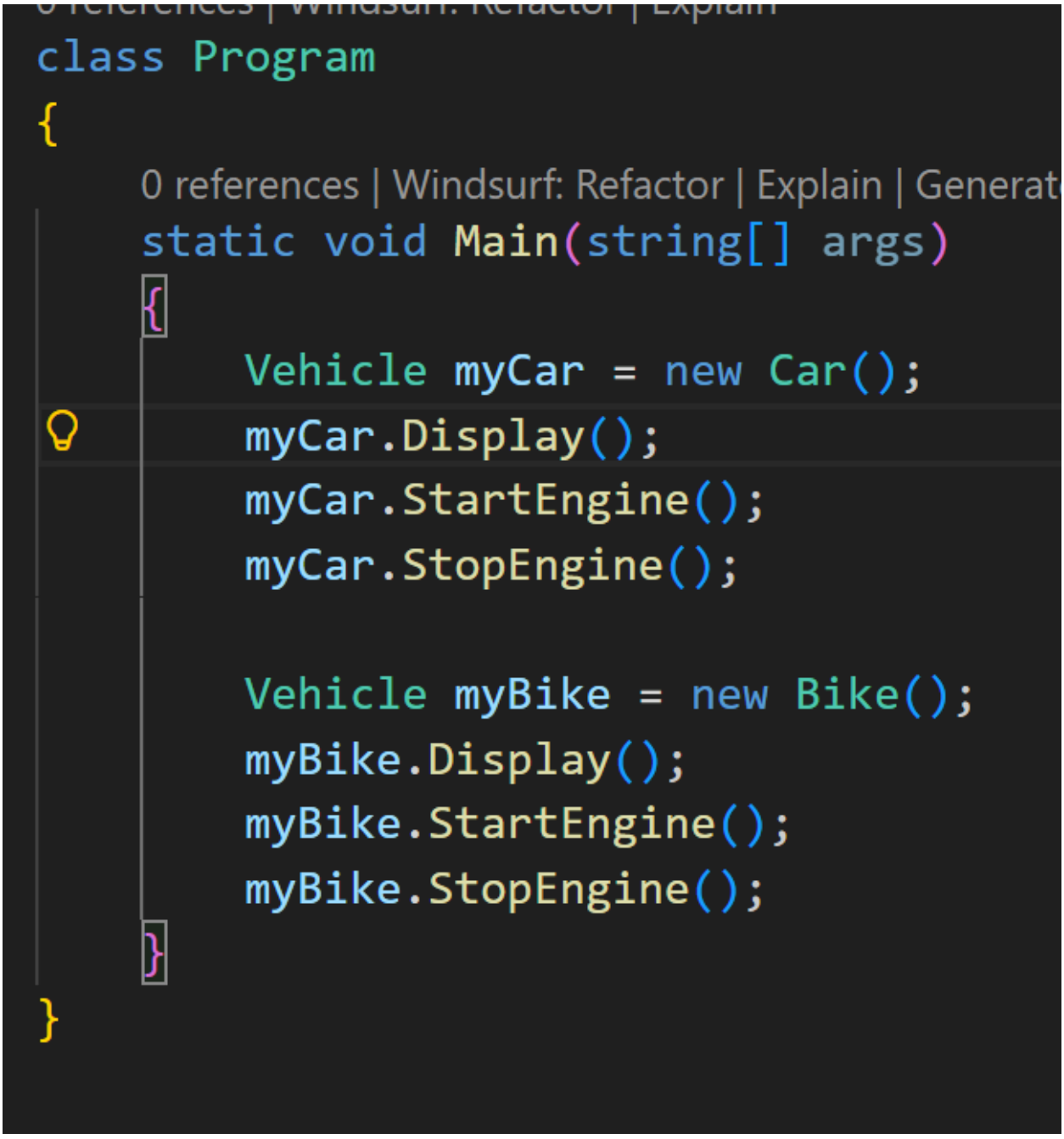
4. Task 4

```
abstract class Vehicle
{
    4 references
    public abstract void StartEngine();
    4 references
    public abstract void StopEngine();

    2 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void Display()
    {
        Console.WriteLine("This is a vehicle");
    }
}
1 reference | Windsurf: Refactor | Explain
class Car : Vehicle
{
    3 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void StartEngine()
    {
        Console.WriteLine("Car engine started.");
    }

    3 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void StopEngine()
    {
        Console.WriteLine("Car engine stopped.");
    }
}
1 reference | Windsurf: Refactor | Explain
class Bike : Vehicle
{
    3 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void StartEngine()
    {
        Console.WriteLine("Bike engine started.");
    }

    3 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void StopEngine()
    {
        Console.WriteLine("Bike engine stopped.");
    }
}
0 references | Windsurf: Refactor | Explain
```


class Program
{
0 references | Windsurf: Refactor | Explain | Generat
static void Main(string[] args)
{
Vehicle myCar = new Car();
myCar.Display();
myCar.StartEngine();
myCar.StopEngine();
Vehicle myBike = new Bike();
myBike.Display();
myBike.StartEngine();
myBike.StopEngine();
}
}

Output:

```
PS D:\College\Application Development\Week 4\Workshop5> dotnet run
This is a vehicle
Car engine started.
Car engine stopped.
This is a vehicle
Bike engine started.
Bike engine stopped.
PS D:\College\Application Development\Week 4\Workshop5>
```

5. Task 5

5.1.

```
abstract class ElectronicDevice
{
    2 references
    private string brand;
    2 references
    private double price;

    3 references
    public string Brand
    {
        get { return brand; }
        set { brand = value; }
    }

    3 references
    public double Price
    {
        get { return price; }
        set { price = value; }
    }

    2 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public ElectronicDevice(string brand, double price)
    {
        Brand = brand;
        Price = price;
    }

    2 references
    public abstract void ShowInfo();
}
```

5.2.

```
class Laptop : ElectronicDevice
{
    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public Laptop(string brand, double price) : base(brand, price)
    {
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void TurnOnBattery()
    {
        Console.WriteLine("Laptop battery is now ON.");
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void ShowInfo()
    {
        Console.WriteLine($"Laptop Brand: {Brand}, Price: {Price}");
    }
}

4 references | Windsurf: Refactor | Explain
class Smartphone : ElectronicDevice
{
    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public Smartphone(string brand, double price) : base(brand, price)
    {
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void EnableCamera()
    {
        Console.WriteLine("Smartphone camera is now ON.");
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public override void ShowInfo()
    {
        Console.WriteLine($"Smartphone Brand: {Brand}, Price: {Price}");
    }
}
```

5.3.

```
2 references | Windsurf: Refactor | Explain
class ElectronicsStore
{
    3 references
    private List<ElectronicDevice> devices = new List<ElectronicDevice>();

    2 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void AddDevice(ElectronicDevice device)
    {
        devices.Add(device);
    }

    0 references | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void RemoveDevice(ElectronicDevice device)
    {
        devices.Remove(device);
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | ×
    public void ShowAllDeviceDetails()
    {
        foreach (var device in devices)
        {
            device.ShowInfo();
            if (device is Laptop laptop)
            {
                laptop.TurnOnBattery();
            }
            else if (device is Smartphone smartphone)
            {
                smartphone.EnableCamera();
            }
        }
    }
}
```

5.4.

```
0 references | Windsurf: Refactor | Explain
class Program
{
    0 references | Windsurf: Refactor | Explain | Generate Documentation |
    static void Main(string[] args)
    {
        ElectronicsStore store = new ElectronicsStore();

        Laptop laptop = new Laptop("Dell", 1200.00);
        Smartphone smartphone = new Smartphone("Samsung", 800.00);

        store.AddDevice(laptop);
        store.AddDevice(smartphone);

        store.ShowAllDeviceDetails();
    }
}
```

Output:

```
Laptop Brand: Dell, Price: 1200
Laptop battery is now ON.
Smartphone Brand: Samsung, Price: 800
Smartphone camera is now ON.
PS D:\College\Application Development\Week 4\Workshop5>
```