John Wang Best Buy Assignment

Exercise 1

This is an interesting garbage collection question. There are different outputs on different scenarios.

runnning environment	running mode	output
vs	debug	0
vs	release	0
Command line	debug	0
Command line	release	10

In command line, we need to redirect the output to a text file, and then display the text file.

On general, it isn't good practice to create a destructor or call GC.Collect explicitly because it might cause performace issue. If an object needs to do cleanup work for some unmanaged objects, such as files, connections, it should implement IDisposable to do properly.

The vs/release mode actually is similar to vs/debug, so we can group the above results as 2 results. debug gives 0 as output, while release gives 10 as output.

Though we can define destructor for a class, and we can call garbage collection explicitly, The time the garbage collection really happens is not controled by application code, rather by .net.

In debug mode, including the vs/release mode, garbage collection usually happens at the code exit. So the destructor ~SomeSubscriber() got called after the output statement "Console.WriteLine(SomeSubscriber.Count.ToString());". That is why the output is 0.

While in command line release mode, the garbage collection is triggered by "GC.Collect();" statement and when the objects are out of scope, so destructor got called before the output statement "Console.WriteLine(SomeSubscriber.Count.ToString());". Due to there are 10 objects of subscriber to be destructed, and the SomeSubscriber.Count is static variable which exists exactly one place in memory for the duration of the program, so SomeSubscriber.Count got accumulated. That is why the output is 10.

****

Exercise 2

CoinJar

I used oo code to do the exercise. There are 2 types of objects involved, Jar and Coin.

Please see class diagram at file “CoinJar Class Diagram.pdf”.

Jar has public method ResetJar() and PrintJar(). I used properties for CoinxxxCounter, Currentamount, and CurrentVolume. used constant maxVolume for the jar volume.

For coin, I defined interface ICoin, and abstract class Coin which is derived from interface ICoin.

Abstract class Coin defined fields, instead of properties, for diameter and thickness of coin. That is because we only need to set the diameter and thickness at constructor level and we only access diameter and thickness within the class. By the way, I got coin specifications from wiki.

Abstract class Coin defined some private methods, such as,

    private decimal GetCircleAreaByRadius(decimal radius)

private decimal GetCircleAreaByDiameter(decimal diameter)
private decimal GetCylinderVoulme(decimal diameter, decimal high)
private decimal ConvertCubicMmToFluidOunce(decimal cubicMm)

They are only used inside the class.

For all coins with different values defined classes, Coin001 for 1-cent coin, Coin005 for 5-cent coin, ..., Coin100 for $1 coin. All these Coinxxx classes are derived from the base class Coin, and implemented abstract method AddToJar based on the face value.

This design fits "loose coupling and high cohesion" principle well. When US want to disuse 1-cent coins like Canada, we can just safely remove the class Coin001. When US want to add $2 coins like Canada, we can define Coin200 derived from Coin. It won't affect other coinxxx classes.

The reason to choose decimal for money is because decimal provides 28-29 decimal digits of accuracy, while double only has 16 digits of accuracy.

I also provided unit test and integration test code in project CoinJarTest. Unit tests include testing Coinxxx constructor, AddToJar method, and Jar class ResetJar method. The integration tests include filling up the jar with 1-cent coins, filling up the jar with 5-cent coins, ... filling up the jar with random value coins.

The code and test code worked as expected.

I copied all source files into this pdf file. But you can still got vs solution files from folder CoinJar if you need to open it in vs.

// Coin Jar source code

// Please see class diagram at file “CoinJar Class Diagram.pdf”

//ConJar Project

// Jar.cs begins

```
using System;

namespace CoinJar
{
    /// <summary>
    /// Class Jar with capacity of 32 US fluid ounces
    /// </summary>
    public class Jar
    {
        /// <summary>
        /// capacity of the jar
        /// </summary>
        public const int maxVolume = 32;

        /// <summary>
        /// Current voulme filled by coins. The remaining volume is maxVolume-CurrentVolume
        /// </summary>
        public decimal CurrentVolume { get; set; }

        /// <summary>
        /// Current amount of the coins
        /// </summary>
        public decimal CurrentAmount { get; set; }
```

```
/// <summary>
/// Counter of 1-cent coins
/// </summary>
public int Coin001Counter { get; set; }

/// <summary>
/// Counter of 5-cent coins
/// </summary>
public int Coin005Counter { get; set; }

/// <summary>
/// Counter of 10-cent coins
/// </summary>
public int Coin010Counter { get; set; }

/// <summary>
/// Counter of 25-cent coins
/// </summary>
public int Coin025Counter { get; set; }

/// <summary>
/// Counter of 50-cent coins
/// </summary>
public int Coin050Counter { get; set; }

/// <summary>
/// Counter of $1 coins
/// </summary>
public int Coin100Counter { get; set; }

/// <summary>
/// Reset the jar
/// </summary>
public void ResetJar()
{
    CurrentAmount = 0;
    CurrentVolume = 0;
    Coin001Counter = 0;
    Coin005Counter = 0;
    Coin010Counter = 0;
    Coin025Counter = 0;
```

```
            Coin050Counter = 0;
            Coin100Counter = 0;
        }

        /// <summary>
        /// print the jar
        /// </summary>
        public void PrintJar()
        {
            Console.WriteLine("\r\nPrint CoinJar as of {0}", DateTime.Now);
            Console.WriteLine("\tCurrent Amount {0:C}", CurrentAmount);
            Console.WriteLine("\tCoin of 1 Cent Counter {0}", Coin001Counter);
            Console.WriteLine("\tCoin of 5 Cents Counter {0}", Coin005Counter);
            Console.WriteLine("\tCoin of 10 Cents Counter {0}", Coin010Counter);
            Console.WriteLine("\tCoin of 25 Cents Counter {0}", Coin025Counter);
            Console.WriteLine("\tCoin of 50 Cents Counter {0}", Coin050Counter);
            Console.WriteLine("\tCoin of 1 Dollars Counter {0}", Coin100Counter);
            Console.WriteLine("\tCurrent Voulme {0}", CurrentVolume);
        }


    }
}
```

// Jar.cs ends

// ICoin.cs begins

```
namespace CoinJar
{
    interface ICoin
    {
        /// <summary>
        /// Get volume of the coin in fluid ounces
        /// </summary>
        /// <returns></returns>
```

```
        decimal GetVolume();
    }
}
```

// ICoin.cs ends

// Coin.cs begins

```
using System;

namespace CoinJar
{
    /// <summary>
    /// Base class for coin.
    /// </summary>
    public abstract class Coin : ICoin
    {
        /// <summary>
        /// diameter of coin in mm.
        /// </summary>
        public decimal diameter;

        /// <summary>
        /// thickness of coin in mm.
        /// </summary>
        public decimal thickness;

        /// <summary>
        /// Get volume of the coin in fluid ounces
        /// </summary>
        /// <returns></returns>
        public decimal GetVolume()
        {
            //calculate volume of the coin. convert it into US fluid ounces
            // 1 millimeter cubed = 3.38140227 × 10-5 US fluid ounces
```

```
            decimal v = ConvertCubicMmToFluidOunce(GetCylinderVoulme(diameter, thickness));

            return v;
        }

        /// <summary>
        /// Abstract method to add the coin to the jar.
        /// Due to different coins have different specification and value, we need to implement it in derived sub-class
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public abstract bool AddToJar(Jar jar);

        // Get circle area by radius
        private decimal GetCircleAreaByRadius(decimal radius)
        {
            //circle area = radius * radius * PI
            return radius * radius * Convert.ToDecimal(Math.PI);
        }


        // Get circle area by daimeter
        private decimal GetCircleAreaByDiameter(decimal diameter)
        {
            decimal r = diameter / 2.0m;
            return GetCircleAreaByRadius(r);
        }

        // Get cylinder volume by diameter and high
        private decimal GetCylinderVoulme(decimal diameter, decimal high)
        {
            return GetCircleAreaByDiameter(diameter)*high;
        }

        // Convert  cubic mm to fluid ounces
        private decimal ConvertCubicMmToFluidOunce(decimal cubicMm)
        {
            // 1 millimeter cubed = 3.38140227 × 10-5 US fluid ounces
            return cubicMm* 0.0000338140227m;
        }
    }
}
```

// Coin.cs ends

// Coin001.cs begins

```
namespace CoinJar
{
    /// <summary>
    /// Coin class of 1-cent coin.
    /// </summary>
    public class Coin001: Coin
    {
        /// <summary>
        /// Constructor to set coin diameter and thickness
        /// </summary>
        public Coin001()
        {
            diameter = 19.00m;
            thickness = 1.55m;
        }

        /// <summary>
        /// Add the coin to a jar
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public override bool AddToJar(Jar jar)
        {
            decimal v = GetVolume();

            if (jar.CurrentVolume + v < Jar.maxVolume)
            {
                jar.Coin001Counter++;
                jar.CurrentAmount += 0.01m;
                jar.CurrentVolume += v;
                return true;
```

```
            }
            else
            {
                return false;
            }
        }
    }
}
```

// Coin001.cs ends

// Coin005.cs begins

```
namespace CoinJar
{
    /// <summary>
    /// Coin class of 5-cent coin.
    /// </summary>
    public class Coin005 : Coin
    {
        /// <summary>
        /// Constructor to set coin diameter and thickness
        /// </summary>
        public Coin005()
        {
            diameter = 21.21m;
            thickness = 1.95m;
        }

        /// <summary>
        /// Add the coin to a jar
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public override bool AddToJar(Jar jar)
        {
```

```
            decimal v = GetVolume();

            if (jar.CurrentVolume + v < Jar.maxVolume)
            {
                jar.Coin005Counter++;
                jar.CurrentAmount += 0.05m;
                jar.CurrentVolume += v;
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

// Coin005.cs ends

// Coin010.cs begins

```
namespace CoinJar
{
    /// <summary>
    /// Coin class of 10-cent coin.
    /// </summary>
    public class Coin010 : Coin
    {
        /// <summary>
        /// Constructor to set coin diameter and thickness
        /// </summary>
        public Coin010()
        {
            diameter = 17.91m;
            thickness = 1.35m;
        }
```

```
        /// <summary>
        /// Add the coin to a jar
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public override bool AddToJar(Jar jar)
        {
            decimal v = GetVolume();

            if (jar.CurrentVolume + v < Jar.maxVolume)
            {
                jar.Coin010Counter++;
                jar.CurrentAmount += 0.10m;
                jar.CurrentVolume += v;
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

// Coin010.cs ends

// Coin025.cs begins

```
namespace CoinJar
{
    /// <summary>
    /// Coin class of 25-cent coin.
    /// </summary>
    public class Coin025 : Coin
    {
```

```
        /// <summary>
        /// Constructor to set coin diameter and thickness
        /// </summary>
        public Coin025()
        {
            diameter = 24.26m;
            thickness = 1.75m;
        }

        /// <summary>
        /// Add the coin to a jar
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public override bool AddToJar(Jar jar)
        {
            decimal v = GetVolume();

            if (jar.CurrentVolume + v < Jar.maxVolume)
            {
                jar.Coin025Counter++;
                jar.CurrentAmount += 0.25m;
                jar.CurrentVolume += v;
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

// Coin025.cs ends

// Coin050.cs begins

```
namespace CoinJar
{
    /// <summary>
    /// Coin class of 50-cent coin.
    /// </summary>
    public class Coin050 : Coin
    {
        /// <summary>
        /// Constructor to set coin diameter and thickness
        /// </summary>
        public Coin050()
        {
            diameter = 30.61m;
            thickness = 2.15m;
        }

        /// <summary>
        /// Add the coin to a jar
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public override bool AddToJar(Jar jar)
        {
            decimal v = GetVolume();

            if (jar.CurrentVolume + v < Jar.maxVolume)
            {
                jar.Coin050Counter++;
                jar.CurrentAmount += 0.50m;
                jar.CurrentVolume += v;
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

// Coin050.cs ends

// Coin100.cs begins

```
namespace CoinJar
{
    /// <summary>
    /// Coin class of $1 coin
    /// </summary>
    public class Coin100 : Coin
    {
        /// <summary>
        /// Constructor to set coin diameter and thickness
        /// </summary>
        public Coin100()
        {
            diameter = 26.50m;
            thickness = 2.00m;
        }



        /// <summary>
        /// Add the coin to a jar
        /// </summary>
        /// <param name="jar"></param>
        /// <returns>true: succeeded adding the coin to the jar</returns>
        public override bool AddToJar(Jar jar)
        {
            decimal v = GetVolume();

            if (jar.CurrentVolume + v < Jar.maxVolume)
            {
                jar.Coin100Counter++;
                jar.CurrentAmount += 1.00m;
                jar.CurrentVolume += v;
                return true;
```

```
            }
            else
            {
                return false;
            }
        }
    }
}
```

// Coin100.cs ends

// Program.cs begins

```
using System;

namespace CoinJar
{
    class Program
    {
        /// <summary>
        /// demo
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            Jar jar = new Jar();
            Console.WriteLine("Initially:\n");
            jar.PrintJar();

            var coin001 = new Coin001();
            coin001.AddToJar(jar);

            Console.WriteLine("\r\n\r\n\r\n\r\n\r\nAfter added 1 cent to the jar:\n");
            jar.PrintJar();


        }
```

```
    }
}
```

// Program.cs ends

//ConJarTest Project

// JarTest.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Test jar
    /// </summary>
    [TestClass]
    public class JarTest
    {
        /// <summary>
        /// Test jar resetting
        /// </summary>
        [TestMethod]
        public void ResetJarTest()
        {
            Jar jar = new Jar();
            Coin001 coin001 = new Coin001();
            coin001.AddToJar(jar);

            jar.ResetJar();

            // After calling jar.ResetJar() all properties of jar should be 0.
            Assert.AreEqual(0.0m, jar.CurrentAmount);
```

```
            Assert.AreEqual(0.0m, jar.CurrentVolume);
            Assert.AreEqual(0, jar.Coin001Counter);
            Assert.AreEqual(0, jar.Coin005Counter);
            Assert.AreEqual(0, jar.Coin010Counter);
            Assert.AreEqual(0, jar.Coin025Counter);
            Assert.AreEqual(0, jar.Coin050Counter);
            Assert.AreEqual(0, jar.Coin100Counter);
        }
    }
}
```

// JarTest.cs ends

// Coin001Test.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;


namespace CoinJarTest
{
    /// <summary>
    /// Test Coin001 class
    /// </summary>
    [TestClass]
    public class Coin001Test
    {
        /// <summary>
        /// Test Coin001 constructor
        /// </summary>
        [TestMethod]
        public void Coin001ConstructorTest()
        {
            Coin001 coin001 = new Coin001();

            //assert coin of 1 cent has diameter of 19.00mm, and thickness of 1.55mm
            Assert.AreEqual(19.00m, coin001.diameter);
```

```
            Assert.AreEqual(1.55m, coin001.thickness);

        }


        /// <summary>
        /// Test Coin001.AddJar method
        /// </summary>
        [TestMethod]
        public void Coin001AddJarTest()
        {
            Coin001 coin001 = new Coin001();
            Jar jar = new Jar();

            //add the 1-cent coin to the jar
            coin001.AddToJar(jar);

            //assert jar.Coin001Counter is 1 and jar.CurrentAmount is $0.01
            Assert.AreEqual(1, jar.Coin001Counter);
            Assert.AreEqual(0.01m, jar.CurrentAmount);
        }

    }
}
```

// Coin001Test.cs ends

// Coin005Test.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;


namespace CoinJarTest
{
```

```
    /// <summary>
    /// Test Coin005 class
    /// </summary>
    [TestClass]
    public class Coin005Test
    {
        /// <summary>
        /// Test Coin005 constructor
        /// </summary>
        [TestMethod]
        public void Coin005ConstructorTest()
        {
            Coin005 coin005 = new Coin005();

            //assert coin of 5 cent has diameter of 21.21mm, and thickness of 1.95mm
            Assert.AreEqual(21.21m, coin005.diameter);
            Assert.AreEqual(1.95m, coin005.thickness);
        }


        /// <summary>
        /// Test Coin005.AddJar method
        /// </summary>
        [TestMethod]
        public void Coin005AddJarTest()
        {
            Coin005 coin005 = new Coin005();
            Jar jar = new Jar();

            //add the 5-cent coin to the jar
            coin005.AddToJar(jar);

            //assert jar.Coin005Counter is 1 and jar.CurrentAmount is $0.05
            Assert.AreEqual(1, jar.Coin005Counter);
            Assert.AreEqual(0.05m, jar.CurrentAmount);

        }

    }
}
```

// Coin005Test.cs ends

// Coin010Test.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;


namespace CoinJarTest
{
    /// <summary>
    /// Test Coin010 class
    /// </summary>
    [TestClass]
    public class Coin010Test
    {
        /// <summary>
        /// Test Coin010 constructor
        /// </summary>
        [TestMethod]
        public void Coin010ConstructorTest()
        {
            Coin010 coin010 = new Coin010();

            //assert coin of 10 cent has diameter of 17.91mm, and thickness of 1.35mm
            Assert.AreEqual(17.91m, coin010.diameter);
            Assert.AreEqual(1.35m, coin010.thickness);

        }


        /// <summary>
        /// Test Coin010AddJar method
        /// </summary>
```

```
        [TestMethod]
        public void Coin010AddJarTest()
        {
            Coin010 coin010 = new Coin010();
            Jar jar = new Jar();

            //add the 10-cent coin to the jar
            coin010.AddToJar(jar);

            //assert jar.Coin010Counter is 1 and jar.CurrentAmount is $0.10
            Assert.AreEqual(1, jar.Coin010Counter);
            Assert.AreEqual(0.10m, jar.CurrentAmount);

        }

    }
}
```

// Coin010Test.cs ends

// Coin025Test.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;


namespace CoinJarTest
{
    /// <summary>
    /// Test Coin025 class
    /// </summary>
    [TestClass]
    public class Coin025Test
    {
        /// <summary>
```

```
        /// Test Coin025 constructor
        /// </summary>
        [TestMethod]
        public void Coin025ConstructorTest()
        {
            Coin025 coin025 = new Coin025();
            //assert coin of 25 cent has diameter of 24.26mm, and thickness of 1.75mm
            Assert.AreEqual(24.26m, coin025.diameter);
            Assert.AreEqual(1.75m, coin025.thickness);

        }


        /// <summary>
        /// Test Coin025.AddJar method
        /// </summary>
        [TestMethod]
        public void Coin025AddJarTest()
        {
            Coin025 coin025 = new Coin025();
            Jar jar = new Jar();

            //add the 25-cent coin to the jar
            coin025.AddToJar(jar);

            //assert jar.Coin025Counter is 1 and jar.CurrentAmount is $0.25
            Assert.AreEqual(1, jar.Coin025Counter);
            Assert.AreEqual(0.25m, jar.CurrentAmount);

        }

    }
}
```

// Coin025Test.cs ends

// Coin050Test.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;


namespace CoinJarTest
{
    /// <summary>
    /// Test Coin050 class
    /// </summary>
    [TestClass]
    public class Coin050Test
    {
        /// <summary>
        /// Test Coin050 constructor
        /// </summary>
        [TestMethod]
        public void Coin050ConstructorTest()
        {
            Coin050 coin050 = new Coin050();

            //assert coin of 50 cent has diameter of 30.61mm, and thickness of 2.15mm
            Assert.AreEqual(30.61m, coin050.diameter);
            Assert.AreEqual(2.15m, coin050.thickness);

        }


        /// <summary>
        /// Test Coin050.AddJar method
        /// </summary>
        [TestMethod]
        public void Coin050AddJarTest()
        {
            Coin050 coin050 = new Coin050();
            Jar jar = new Jar();

            //add the 50-cent coin to the jar
            coin050.AddToJar(jar);
```

```
            //assert jar.Coin050Counter is 1 and jar.CurrentAmount is $0.50
            Assert.AreEqual(1, jar.Coin050Counter);
            Assert.AreEqual(0.50m, jar.CurrentAmount);

        }

    }
}
```

// Coin050Test.cs ends

// Coin100Test.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;


namespace CoinJarTest
{
    /// <summary>
    /// Test Coin100 class
    /// </summary>
    [TestClass]
    public class Coin100Test
    {
        /// <summary>
        /// Test Coin100 constructor
        /// </summary>
        [TestMethod]
        public void Coin100ConstructorTest()
        {
            Coin100 coin100 = new Coin100();

            //assert coin of $1 has diameter of 26.50mm, and thickness of 2.00mm
            Assert.AreEqual(26.50m, coin100.diameter);
```

```
            Assert.AreEqual(2.00m, coin100.thickness);

        }


        /// <summary>
        /// Test Coin100.AddJar method
        /// </summary>
        [TestMethod]
        public void Coin100AddJarTest()
        {
            Coin100 coin100 = new Coin100();
            Jar jar = new Jar();

            //add the $1 coin to the jar
            coin100.AddToJar(jar);

            //assert jar.Coin100Counter is 1 and jar.CurrentAmount is $1.00
            Assert.AreEqual(1, jar.Coin100Counter);
            Assert.AreEqual(1.00m, jar.CurrentAmount);

        }

    }
}
```

// Coin100Test.cs ends

// IntegrationTest_FillUpWith100CentCoins.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
```

```
    /// <summary>
    /// Integration test to fill up jar with $1 coins.
    /// </summary>
    [TestClass]
    public class IntegrationTest_FillUpWith100CentCoins
    {
        /// <summary>
        /// Fill up with $1 coins.
        /// When the jar is full, it has $857 as amount, and 857 coins of $1
        /// </summary>
        [TestMethod]
        public void FillUpwith100CentCoinTest()
        {
            Jar jar = new Jar();
            Coin100 coin100 = new Coin100();

            //fill up with $1 coins
            while (coin100.AddToJar(jar));

            jar.PrintJar();
        }
    }
}
```

// IntegrationTest_FillUpWith100CentCoins.cs ends

// IntegrationTest_FillUpWith10CentCoins.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Integration test to fill up jar with 10-cent coins.
```

```
    /// </summary>
    [TestClass]
    public class IntegrationTest_FillUpWith10CentCoins
    {
        /// <summary>
        /// Fill up with 10-cent coins.
        /// When the jar is full, it has $278.20 as amount, and 2782 coins of 10-cent
        /// </summary>
        [TestMethod]
        public void FillUpwith10CentCoinTest()
        {
            Jar jar = new Jar();
            Coin010 coin010 = new Coin010();

            //fill up with 10-cent coins
            while (coin010.AddToJar(jar));

            jar.PrintJar();
        }
    }
}
```

// IntegrationTest_FillUpWith10CentCoins.cs ends

// IntegrationTest_FillUpWith1CentCoins.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Integration test to fill up jar with 1-cent coins.
    /// </summary>
    [TestClass]
```

```
    public class IntegrationTest_FillUpWith1CentCoins
    {
        /// <summary>
        /// Fill up with 1-cent coins.
        /// When the jar is full, it has $21.53 as amount, and 2153 coins of 1-cent
        /// </summary>
        [TestMethod]
        public void FillUpwith1CentCoinTest()
        {
            Jar jar = new Jar();
            Coin001 coin001 = new Coin001();

            //fill up with 1-cent coins
            while (coin001.AddToJar(jar));

            jar.PrintJar();
        }
    }
}
```

// IntegrationTest_FillUpWith1CentCoins.cs ends

// IntegrationTest_FillUpWith25CentCoins.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Integration test to fill up jar with 25-cent coins.
    /// </summary>
    [TestClass]
    public class IntegrationTest_FillUpWith25CentCoins
    {
```

```
        /// <summary>
        /// Fill up with 25-cent coins.
        /// When the jar is full, it has $292.25 as amount, and 1169 coins of 25-cent
        /// </summary>
        [TestMethod]
        public void FillUpwith25CentCoinTest()
        {
            Jar jar = new Jar();
            Coin025 coin025 = new Coin025();

            //fill up with 25-cent coins
            while (coin025.AddToJar(jar));

            jar.PrintJar();
        }
    }
}
```

// IntegrationTest_FillUpWith25CentCoins.cs ends

// IntegrationTest_FillUpWith50CentCoins.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Integration test to fill up jar with 50-cent coins.
    /// </summary>
    [TestClass]
    public class IntegrationTest_FillUpWith50CentCoins
    {
        /// <summary>
        /// Fill up with 50-cent coins.
```

```
        /// When the jar is full, it has $299 as amount, and 598 coins of 50-cent
        /// </summary>
        [TestMethod]
        public void FillUpwith50CentCoinTest()
        {
            Jar jar = new Jar();
            Coin050 coin050 = new Coin050();

            //fill up with 50-cent coins
            while (coin050.AddToJar(jar));

            jar.PrintJar();
        }
    }
}
```

// IntegrationTest_FillUpWith50CentCoins.cs ends

// IntegrationTest_FillUpWith5CentCoins.cs begins

```
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Integration test to fill up jar with 5-cent coins.
    /// </summary>
    [TestClass]
    public class IntegrationTest_FillUpWith5CentCoins
    {
        /// <summary>
        /// Fill up with 5-cent coins.
        /// When the jar is full, it has $68.65 as amount, and 1373 coins of 5-cent
        /// </summary>
```

```
        [TestMethod]
        public void FillUpwith5CentCoinTest()
        {
            Jar jar = new Jar();
            Coin005 coin005 = new Coin005();

            //fill up with 5-cent coins
            while (coin005.AddToJar(jar));

            jar.PrintJar();
        }
    }
}
```

// IntegrationTest_FillUpWith5CentCoins.cs ends

// IntegrationTest_FillUpWithRandomCoins.cs begins

```
using System;
using CoinJar;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoinJarTest
{
    /// <summary>
    /// Integration test to fill up jar with random coins.
    /// </summary>
    [TestClass]
    public class IntegrationTest_FillUpWithRandomCoins
    {
        /// <summary>
        /// Fill up with random coins.
        /// </summary>
        [TestMethod]
        public void FillUpwithRandomCoinTest()
```

```
{
    Jar jar = new Jar();

    // The Random() constructor uses the system clock to provide a seed value.
    Random rnd = new Random();

    // coin.AddToJar failure indicates jar full.
    var addJarSucceeded=false;

    do
    {
        // Generate random int with range of [1,6]
        // 1-- 1 cent coin
        // 2-- 5 cent coin
        // 3-- 10 cent coin
        // 4-- 25 cent coin
        // 5-- 50 cent coin
        // 6-- $1 coin
        var coinType = rnd.Next(1, 7);
        switch (coinType)
        {
            case 1:
                addJarSucceeded = new Coin001().AddToJar(jar);
                break;
            case 2:
                addJarSucceeded = new Coin005().AddToJar(jar);
                break;
            case 3:
                addJarSucceeded = new Coin010().AddToJar(jar);
                break;
            case 4:
                addJarSucceeded = new Coin025().AddToJar(jar);
                break;
            case 5:
                addJarSucceeded = new Coin050().AddToJar(jar);
                break;
            default:
                addJarSucceeded = new Coin100().AddToJar(jar);
                break;
        }
    }
    //loop while last filling succeeded
```

```
            while (addJarSucceeded);

            jar.PrintJar();
        }
    }
}
// IntegrationTest_ FillUpWithRandomCoins.cs ends
```