public class User

{

    private readonly DateTime \_refDate = new DateTime(2015, 1, 1);

    // SUT

    public string GetUserId()

    {

        if (DateTime.Today < \_refDate)

            return "1";

        else

            return "00001";

    }

}

[TestClass]

public class UserTest

{

    [TestMethod]

    public void Should\_Return\_New\_UserId()

    {

        var expected = "00001";

        // ARRANGE

        // Datetime.Today should return a date after the 01/01/2015

        var todayDate = new DateTime(2015,2,1);

        Mock.Arrange(() => DateTime.Today).Returns(todayDate);

        // ACT

        var user = new User();

        var actual = user.GetUserId();

        // ASSERT

        Assert.AreEqual(expected, actual);

    }

}

    [TestClass]

public class UserTest

{

    public class User

    {

        // SUT

        public int GetUserId(string userName)

        {

            return UserStaticRepository.GetUserId(userName);

        }

    }

    //Static Dependency

    public static class UserStaticRepository

    {

        static UserStaticRepository()

        {

            throw new **NotImplementedException**();

        }

        public static int GetUserId(string userName)

        {

            throw new **NotImplementedException**();

        }

    }

    [TestMethod]

    public void GetUserIdTest()

    {

        var expected = 10;

        //ARRANGE

        Mock.SetupStatic(typeof (UserStaticRepository), StaticConstructor.Mocked);

        Mock.Arrange(() => UserStaticRepository.GetUserId(Arg.AnyString)).Returns(expected);

        //ACT

        var user = new User();

        var actual = user.GetUserId("Fathi");

        //ASSERT

        Assert.AreEqual(expected, actual);

    }

}

public class User

 {

     // SUT

     public int GetUserId(string userName)

     {

         // dependency created and used inside the method

         var repository = new UserRepository();

         var id = repository.GetUserId(userName);

         return id;

     }

 }

 // Dependency

 public class UserRepository

 {

     public int GetUserId(string userName)

     {

         throw new **NotImplementedException**();

     }

 }

 [TestClass]

 public class UserTest

 {

     [TestMethod]

     public void GetUserIdTest()

     {

         var expected = 10;

         // ARRANGE

         //object used inside the SUT

         var futureMock = Mock.Create<UserRepository>();

         Mock.Arrange(() => futureMock.GetUserId(Arg.AnyString)).IgnoreInstance().Returns(10);

         //ACT

         var user = new User();

         var actual = user.GetUserId("Fathi");

         // ASSERT

         Assert.AreEqual(expected, actual);

     }

 }

public class User

    {

        // Dependency

        public User()

        {

            //call db to fill User properties

            throw new **NotImplementedException**();

        }

        // SUT

        public int GetUserId()

        {

            return 1;

        }

    }

    [TestClass]

    public class UserTest

    {

        [TestMethod]

        public void GetUserIdTest()

        {

            var expected = 1;

            // ARRANGE

            var user = Mock.Create<User>(Constructor.Mocked);

            Mock.Arrange(() => user.GetUserId()).CallOriginal();

            //ACT

            var actual = user.GetUserId();

            // ASSERT

            Assert.AreEqual(expected, actual);

        }

    }

public class User

 {

     // Dependency

     private int GetUserIdFromDb()

     {

         throw new **NotImplementedException**();

     }

     // SUT

     public int GetUserId()

     {

         var id = GetUserIdFromDb();

         return id;

     }

 }

 [TestClass]

 public class UserTest

 {

     [TestMethod]

     public void GetUserIdTest()

     {

         var expected = 10;

         var user = new User();

         // ARRANGE - When the non-public function GetUserIdFromDb() is called from the user instance,

         //  it should return expected integer.

         Mock.NonPublic.Arrange<int>(user, "GetUserIdFromDb").Returns(expected);

         // ACT

         int actual = user.GetUserId();

         // ASSERT

         Assert.AreEqual(expected, actual);

     }

 }