# Lab: Reflection and Attributes

# Part I: Reflection

## Stealer

**NOTE**: You need a public StartUp class with the namespace Stealer.

Add the **Hacker** class from the box below to your project.

|  |
| --- |
| **Hacker.cs** |
| public class Hacker  {  public string username = "securityGod82";  private string password = "mySuperSecretPassw0rd";  public string Password  {  get => this.password;  set => this.password = value;  }  private int Id { get; set; }  public double BankAccountBalance { get; private set; }  public void DownloadAllBankAccountsInTheWorld()  {  }  } |

There is one really nasty hacker, but not so wise though. He is trying to steal a big amount of money and transfer it to his own account. The police is after him but they need a professional… Correct - this is you!

You have the information that this hacker is keeping some of his info in private fields. Create a new class named **Spy** and add inside a method called - **StealFieldInfo,** which receives:

* **string** - name of the class to investigate
* **array of string** - names of the fields to investigate

After finding the fields, you must print on the console:

"**Class under investigation: {nameOfTheClass}**"

On the next lines, print info about each field in the following format:

"**{filedName} = {fieldValue}**"

Use **StringBuilder** to concatenate the answer**. Don’t change anything in Hacker class!**

In your **Main()** method, you should be able to check your program with the current piece of code.



### Example

|  |
| --- |
| **Output** |
| Class under investigation: Stealer.Hacker  username = securityGod82  password = mySuperSecretPassw0rd |

### Solution



## High Quality Mistakes

**NOTE**: You need a public StartUp class with the namespace Stealer.

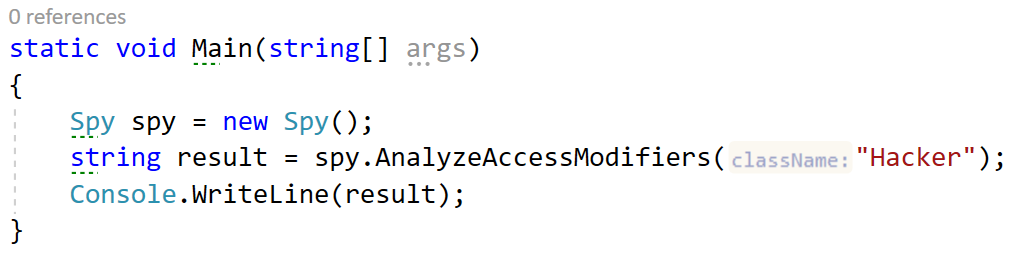
You are already an expert of **High Quality Code**, so you know what kind of **access modifiers** must be set to the members of a class. You should have noticed that our hacker is not familiar with these concepts.

Create a method inside your Spy class called - AnalyzeAccessModifiers(string className). Check all of the **fields and methods access modifiers**. Print on the console all of the **mistakes** in format:

* Fields
  + **{fieldName} must be private!**
* Getters
  + **{methodName} have to be public!**
* Setters
  + **{methodName} have to be private!**

Use **StringBuilder** to concatenate the answer**. Don’t change anything in Hacker class!**

In your **Main()** method you should be able to check your program with the current piece of code.



### Example

**NOTE**: **The order of your output** may **differ** based on your **solution logic**.

|  |
| --- |
| **Output** |
| username must be private!  get\_Id have to be public!  set\_Password have to be private! |

### Solution



## Mission Private Impossible

**NOTE**: You need a public StartUp class with the namespace Stealer.

It’s time to see what this hacker you are dealing with aims to do. Create a method inside your Spy class called - RevealPrivateMethods(stirng className). Print all private methods in the following format:

All Private Methods of Class: **{className}**  
Base Class: **{baseClassName}**  
On the next lines, print found method’s names each on a new line. Use **StringBuilder** to concatenate the answer**. Don’t change anything in Hacker class!** In your **Main()** method, you should be able to check your program with the current piece of code.



### Example

**NOTE**: **The order of your output** may **differ** based on your **solution logic**.

|  |
| --- |
| **Output** |
| All Private Methods of Class: Stealer.Hacker  Base Class: Object  get\_Id  set\_Id  set\_BankAccountBalance  MemberwiseClone  Finalize |

### Solution



## Collector

**NOTE**: You need a public StartUp class with the namespace Stealer.

Use reflection to get all **Hacker** methods. Then prepare an algorithm that will recognize which methods are getters and setters.

Print to console each getter on a new line in the format:  
"**{name} will return {Return Type}**"

Then print all of the setters in the format:  
"**{name}** **will set field of {Parameter Type}**"

Use **StringBuilder** to concatenate the answer**. Don’t change anything in Hacker class!**

In your **Main()** method you should be able to check your program with the current piece of code.



### Example

|  |
| --- |
| **Output** |
| get\_Password will return System.String  get\_Id will return System.Int32  get\_BankAccountBalance will return System.Double  set\_Password will set field of System.String  set\_Id will set field of System.Int32  set\_BankAccountBalance will set field of System.Double |

### Solution



# Part II: Attributes

## Create Attribute

**NOTE**: You need a public **StartUp** class with the namespace **AuthorProblem**.

Create attribute Author with a string element called **name**, that**:**

* Can be used over classes and methods
* Allow multiple attributes of same type

### Examples

|  |
| --- |
| StartUp.cs |
| [Author("Ventsi")]  class StartUp  {  [Author("Gosho")]  static void Main(string[] args)  {  }  } |



## Code Tracker

**NOTE**: You need a public **StartUp** class with the namespace **AuthorProblem**.

Create a class **Tracker** with a method:

* void PrintMethodsByAuthor()

The method above should print on the console information about each method that is written by someone. You should print the information about the method and its creator in the following format: "{methodName} is written by {authorName}". You should be able to call your method and see the output of it as in the example below.

### Examples

|  |
| --- |
| StartUp.cs |
| [Author("Ventsi")]  class StartUp  {  [Author("Gosho")]  static void Main(string[] args)  {  var tracker = new Tracker();  tracker.PrintMethodsByAuthor();  }  } |

