# C# OOP Retake Exam – 18 Dec 2024

# Cyber Security Defense System

A blue circuit board with a lock

Description automatically generated

1. **Overview**

*Welcome to the CyberSecurity Defense System! This application simulates a platform for managing and mitigating cyberattacks using defense strategies and cybersecurity tools. You will implement entities representing different types of attacks, defenses, and software, creating an engaging scenario for understanding cybersecurity principles while applying key OOP concepts.*

## Setup

* Upload **only the CyberSecurityDS** project in every task **except** **Unit Tests.**
* **Do not modify the interfaces or their packages.**
* Use **strong cohesion** and **loose coupling.**
* **Use inheritance and the provided interfaces wherever possible**:
  + This includes **constructors**, **method parameters,** and **return types.**
* **Do not** violate your **interface** **implementations** by adding **more public methods** in the concrete class than the interface has defined.
* Make sure you have **no public fields** anywhere.
* **Exception** and **Output messages** are in the **"Utilities"** folder.
* To solve this problem use **Visual Studio 2022** and **Net 6.0**
* **Do not use** "\r\n" **for a new line.**

## Task 1: Structure (50 points)

**For this task’s evaluation logic in the methods isn’t included.**

You are given some **interfaces** and must **implement** their functionality in the **correct classes**.

There are **3** types of entities: **CyberAttack**, **DefensiveSoftware**, and **SystemManager**. You should also implement two generic repositories: **CyberAttackRepository** and **DefensiveSoftwareRepository**.

### CyberAttack

The CyberAttack is a **base class** representing any type of cyberattack. It defines the fundamental attributes and behaviors common to all types of attacks. This class **cannot be instantiated directly**.

#### Data

* **AttackName** - **string**
  + Must NOT be **null or whitespace.** If **invalid** throw a new **ArgumentException** with the message:

"Attack name is required."

* **SeverityLevel –** **int**
  + Must be a **positive number in the range [1-10]**.
  + If **0** is passed, **set the severity level to 1**, if **greater than 10 is passed**, **set it to 10**
  + If a **negative value** is passed, throw a new **ArgumentException** with the message:   
    "Severity level cannot assign negative values."
* **Status – bool**
  + Indicates whether the attack has been mitigated.
  + Default value: false (not mitigated)

#### Constructor

A CyberAttack should take the following values upon initialization:

string attackName, int severityLevel

#### Behavior

##### void MarkAsMitigated()

Updates the **Status** property to **true**, indicating that the **attack has been mitigated**.

#### Child Classes

The CyberAttack class has two specific child classes: PhishingAttack and MalwareAttack. Each child class adds unique attributes and behaviors related to its specific attack type:

**PhishingAttack**

The PhishingAttack class represents an attack focused on phishing attempts, such as email spoofing.

#### Data

* **TargetMail** - **string**
  + Must NOT be **null or whitespace.** If **invalid** throw a new **ArgumentException** with the message:

"Target mail is required."

The Constructor of the **PhishingAttack** should take the following parameters upon initialization:

string attackName, int severityLevel, string targetMail

#### Override ToString() method:

Overrides the existing method ToString()and modify it, so the returned **string must be on a single line**, in the following format:

"Attack: {AttackName}, Severity: {SeverityLevel} (Target Mail: {TargetMail})"

**MalwareAttack**

The **MalwareAttack** class represents an attack involving malicious software, such as ransomware or viruses.

#### Data

* **InfectedFilesCount** - **string**
  + Must NOT be **null or whitespace.** If **invalid** throw a new **ArgumentException** with the message:

"Infected files count is required."

* *We will use the infected files count as a string*

The Constructor of the **MalwareAttack** should take the following parameters upon initialization:

string attackName, int severityLevel, string infectedFilesCount

#### Override ToString() method:

Overrides the existing method ToString()and modify it, so the returned **string must be on a single line**, in the following format:

"Attack: {AttackName}, Severity: {SeverityLevel} (Infected Files: {InfectedFilesCount})"

### DefensiveSoftware

The DefensiveSoftware is a **base class** representing any type of defensive software. It defines the fundamental attributes and behaviors common to all types of defensive software. This class **cannot be instantiated directly**.

#### Data

* **Name** - **string**
  + If the **name** is **null or whitespace,** throw a new **ArgumentException** with the message:

"Software name is required."

* Effectiveness – int
  + Must be a **positive number in the range [1-10]**.
  + If **0** is passed, **set the effectiveness to 1**, if **greater than 10 is passed**, **set it to 10**

If a **negative value** is passed, throw a new **ArgumentException** with the message:   
"Effectiveness cannot assign negative values."

* **AssignedAttacks – IReadOnlyCollection<string>**
  + A collection of attack names that this defensive software is assigned to
  + Allows a single defensive software to handle multiple attacks

#### Override ToString() method:

Overrides the existing method ToString()and modifies it, so the returned string **must be on a single line**, in the following format:

**"**Defensive Software: **{Name}**, Effectiveness: **{Effectiveness}**, Assigned Attacks: **{attackName1,attackName2,…, attackNamen}"**

**If no attacks are assigned, the string "**[None]**" must appear in place of the attack names.**

**Example:  
 "**Defensive Software: **{Name}**, Effectiveness: **{Effectiveness}**, Assigned Attacks: [None]**"**

#### Constructor

A **DefensiveSoftware** should take the following values upon initialization:

stringname, inteffectiveness

* Also initializes a new instance of the AssignedAttacks property

#### Behavior

##### void AssignAttack(string attackName)

Associates a defensive software with a cyberattack by **adding the attack name** to the collection of **assigned attacks**.

#### Child Classes

There are **two** concrete **types** of **DefensiveSoftware** in our project:

##### Firewall

Represents firewalls used to block or monitor traffic. It can be effectively used only against MalwareAttack

The Constructor of the **Firewall** should take the following parameters upon initialization:

stringname, inteffectiveness

##### Antivirus

Represents antivirus software that scans for malicious files. It can be effectively used only against PhishingAttack

The Constructor of the **Antivirus** should take the following parameters upon initialization:

stringname, inteffectiveness

## CyberAttackRepository

The **CyberAttackRepository** is an **IRepository<ICyberAttack>. Collection** of all existing **cyber attacks in the application**.

### Data

* **Models – a collection of cyber attacks (unmodifiable)**

### Behavior

**void AddNew(ICyberAttack model)**

* **Adds** a **cyber attack** to the repository

**ICyberAttack GetByName(string attackName)**

* Returns a cyber attack with an **attackName value, equal to the given parameter**. If **no such attack is found** in the repository, **returns null**

**bool Exists(string attackName)**

* Returns **true** if a cyber attack with an **attackName matching the given parameter exists** in the repository,   
  **otherwise returns false**

## DefensiveSoftwareRepository

The **DefensiveSoftwareRepository** is an **IRepository<IDefensiveSoftware>. Collection** of all existing **defensive software in the application**.

### Data

* **Models – a collection of defensive software programs (unmodifiable)**

### Behavior

**void AddNew(IDefensiveSoftware model)**

* **Adds** a **defensive software** program to the repository.

**IDefensiveSoftware GetByName(string name)**

* Returns a defensive software with a **name value, equal to the given parameter**. If no such software is found, it **returns null**.

**bool Exists(string name)**

* Returns **true** if a defensive software with a **name matching the given parameter exists** in the repository,   
  **otherwise returns false**.

## SystemManager

### Data

* **CyberAttacks – IRepository<ICyberAttack>**
* **DefensiveSoftwares – IRepository<IDefensiveSoftware>**

### Constructor

A **SystemManager** should take **no values** upon initialization and should **initialize new instances** of the collections.

## Task 2: Business Logic (150 points)

## The Controller Class

The business logic of the program should be concentrated around several **commands**. You are given interfaces, which you have to implement in the correct classes.

**NOTE: Do not use** "\r\n" **for a new line.**

The first interface is the **IController**. You must create a **Controller** class, which implements the interface and implements all of its methods. The constructor of the **Controller** does not take any arguments. The given methods should have the logic described for each in the Commands section. When you create the **Controller** class, go into the **Engine** class constructor and uncomment the "this.controller = new Controller();" line.

**Data**

You need to keep track of some collections, this is why you need a private field in your controller class:

**Example:**

* **systemManager – ISystemManager** (you can name it as you like, **Judge will not test this property**)

**Commands**

There are several commands, which control the business logic. They are stated below.

#### AddCyberAttack Command

##### Parameters

* **attackType – string**
* **attackName – string**
* **severityLevel – int**
* **extraParam – string**
  + Used to set the TargetMail (for PhishingAttack) or InfectedFilesCount (for MalwareAttack)

##### Functionality

The method should **create and add** a new **ICyberAttack** to the appropriate repository in the systemManager.

* Check if the given attackType matches a valid child class of CyberAttack (PhishingAttack or MalwareAttack), if not valid skip the creation process and return the following message: "{attackType**}** is not a valid type for the system."
* If a cyber attack with the same name already exists in the system, skip the creation process and return the following message: "{**attackName}** already exists in the system."
* If none of the above cases is reached, create an instance of the appropriate child class (PhishingAttack or MalwareAttack) based on the attackType. Add the newly created ICyberAttack to the CyberAttackRepository.
  + The following message should be returned:   
    "{**attackType}**: {**attackName}** is added to the system."

#### AddDefensiveSoftware Command

##### Parameters

* **softwareType - string**
* **softwareName – string**
* **effectiveness - int**

##### Functionality

The method should **create and add** a new **IDefensiveSoftware** to the appropriate repository  
in the systemManager.

* Check if the given softwareType matches a valid child class of DefensiveSoftware (Firewall or Antivirus), if not valid skip the creation process and return the following message: "{softwareType**}** is not a valid type for the system."
* If a defensive software with the same name already exists in the system, skip the creation process and return the following message: "{**softwareName}** already exists in the system."
* The validation of effectiveness should **occur within the respective class implementations**
* If none of the above cases is reached, create an instance of the appropriate child class (Firewall or Antivirus) based on the softwareType. Add the newly created IDefensiveSoftware to the DefensiveSoftwareRepository.
  + The following message should be returned:   
    "{**softwareType}**: {**softwareName}** is added to the system."

#### AssignDefense Command

##### Parameters

* **cyberAttackName – string**
* **defensiveSoftwareName - string**

##### Functionality

The method **assigns a cyberattack** **to defensive software**.

* If a cyber attack with the same name is not found in the systemManager, return the following message:   
  "{**cyberAttackName}** does not exist in the system."
* If defensive software with the same name is not found in the **systemManager**, return the   
  following message:   
  **"{**defensiveSoftwareName} does not exist in the system.**"**
* **Extract** a cyber attack with the given cyberAttackName **and a** defensive software with the givendefensiveSoftwareNamefrom thesystemManager'srepositories
* If the **cyberattack is already assigned to a defensive software** (the attack's name exists in the AssignedAttacks collection of **any** software), **skip the process and return the following message:   
  "**{cyberAttackName} is already assigned to {assignedSoftwareName}.**"**
* If none of the above cases is reached, you should add (use the AssignAttack method for DefensiveSoftware) the cyber attack's AttackName to the AssignedAttacks collection of the defensive software.
  + The following message is returned:   
    "{cyberAttackName} is assigned to {**defensiveSoftwareName}**."

#### MitigateAttack Command

##### Parameters

* **cyberAttackName - string**

##### Functionality

The method mitigates a cyberattack using the specified defensive software and applies relevant actions based on their types and properties.

* **If a cyber attack with the same name is not found in the** systemManager**, return the following message:   
  "{**cyberAttackName}does not exist in the system."
* Extract a cyber attack with the given cyberAttackName
* If the Status of the cyber attack is already true (mitigated), skip the process and return: "{**cyberAttackName}** is already mitigated."
* **Check Assignment - Verify if the** cyberAttackName **is in the** AssignedAttacks **collection of any defensive software, if not, return the following message:**"{cyberAttackName}is not assigned yet."
* **Extract the defensive software** for which the **cyberattack is assigned**.
* **Check Compatibility -** Ensure the **type of defensive software** is **compatible** with the **type of** **cyberattack**
  + Firewall: Can mitigate only MalwareAttack, Antivirus: Can mitigate only PhishingAttack
  + **If not, return the following message:**"{defensiveSoftware.TypeName}cannot mitigate {cyberAttack.TypeName}."
* Retrieve the Effectiveness of the defensive software and the SeverityLevel of the cyberattack and compare these properties:
  + If Effectiveness >= SeverityLevel, the cyberattack can be mitigated successfully.
    - Change the Status of the cyberattack to true (mitigated),
    - The following message is returned:   
      "{cyberAttackName} is mitigated successfully."
  + Otherwise return the following message:   
    "{cyberAttackName} could not be mitigated by {**defensiveSoftwareName}**."

#### GenerateReport Command

##### Functionality

This method generates a **detailed report for all defensive software and cyberattacks registered in the system manager**.

##### Structure

The report is structured as follows:

* Defensive softwares appear first, **ordered by** their **names**, **alphabetically**
* **Be careful with the** Status **of each cyber attack**. **Mitigated** and **Pending**(Not Mitigated) attacks should be reported separately.
* Mitigated and pending attacks **should be sorted alphabetically** within their sections

"Security:

**{software1}**

**{software2}**

**…**

**{softwaren}**

Threads:

-Mitigated:

**{attack1}**

**{attack2}**

**…**

**{attackn}**

-Pending:

**{attack1}**

**{attack2}**

**…**

**{attackn}**"

**NOTE: Do not use** "\r\n" **for a new line.**

#### Exit Command

##### Functionality

Ends the program.

### Input / Output

You are provided with one interface, which will help you with the correct execution process of your program. The interface is Engine, and the class implementing this interface should read the input, and when the program finishes, this class should print the output.

#### Input

Below, you can see the **format** in which **each command** will be given in the input:

* **AddCyberAttack {attackType} {attackName} {severityLevel} {extraParam}**
* **AddDefensiveSoftware {softwareType} {softwareName} {effectiveness}**
* **AssignDefense {cyberAttackName} {defensiveSoftwareName}**
* **MitigateAttack {cyberAttackName}**
* **GenerateReport**
* **Exit**

#### Output

Print the output from each command when issued. Print the exception message if an exception is thrown during any of the commands' execution.

#### Examples

|  |
| --- |
| **Input** |
| **AddCyberAttack SQLInjection DataStealer 7 user@examplemail.com**  **AddCyberAttack MalwareAttack RansomLock 9 250**  **AddCyberAttack PhishingAttack RansomLock 7 victim@example.com**  **AddCyberAttack PhishingAttack CredentialHarvesting 6 ceo@bigcompany.com**  **AddCyberAttack MalwareAttack SpywareInfiltration 10 500**  **AddCyberAttack PhishingAttack SocialEngineered 5 hr@companyxyz.com**  **AddCyberAttack MalwareAttack KeyLoggerDeployment 6 120**  **AddCyberAttack PhishingAttack ITCompromise 7 sysadmin@techcompany.com**  **AddDefensiveSoftware Firewall NetShield 8**  **AddDefensiveSoftware InvalidSoftwareType FakeShield 7**  **AddDefensiveSoftware Firewall NetShield 9**  **AddDefensiveSoftware Antivirus VirusHunter 9**  **AddDefensiveSoftware Firewall BasicBlocker 0**  **AddDefensiveSoftware Antivirus UltimateDefender 12**  **AddDefensiveSoftware Firewall SecureWall 7**  **AssignDefense RansomLock NetShield**  **AssignDefense NonExistentAttack NetShield**  **AssignDefense RansomLock FakeDefense**  **AssignDefense RansomLock NetShield**  **AssignDefense CredentialHarvesting NetShield**  **AssignDefense RansomLock SecureWall**  **AssignDefense SpywareInfiltration NetShield**  **AssignDefense ITCompromise VirusHunter**  **AssignDefense KeyLoggerDeployment SecureWall**  **AssignDefense SocialEngineered VirusHunter**  **AssignDefense CredentialHarvesting VirusHunter**  **AssignDefense SpywareInfiltration NonExistentDefense**  **AssignDefense ITCompromise VirusHunter**  **MitigateAttack SpywareInfiltration**  **MitigateAttack NonExistentAttack**  **MitigateAttack ITCompromise**  **MitigateAttack ITCompromise**  **MitigateAttack DataStealer**  **MitigateAttack CredentialHarvesting**  **MitigateAttack RansomLock**  **MitigateAttack KeyLoggerDeployment**  **GenerateReport**  **Exit** |
| **Output** |
| **SQLInjection is not a valid type for the system.**  **MalwareAttack: RansomLock is added to the system.**  **RansomLock already exists in the system.**  **PhishingAttack: CredentialHarvesting is added to the system.**  **MalwareAttack: SpywareInfiltration is added to the system.**  **PhishingAttack: SocialEngineered is added to the system.**  **MalwareAttack: KeyLoggerDeployment is added to the system.**  **PhishingAttack: ITCompromise is added to the system.**  **Firewall: NetShield is added to the system.**  **InvalidSoftwareType is not a valid type for the system.**  **NetShield already exists in the system.**  **Antivirus: VirusHunter is added to the system.**  **Firewall: BasicBlocker is added to the system.**  **Antivirus: UltimateDefender is added to the system.**  **Firewall: SecureWall is added to the system.**  **RansomLock is assigned to NetShield.**  **NonExistentAttack does not exist in the system.**  **FakeDefense does not exist in the system.**  **RansomLock is already assigned to NetShield.**  **CredentialHarvesting is assigned to NetShield.**  **RansomLock is already assigned to NetShield.**  **SpywareInfiltration is assigned to NetShield.**  **ITCompromise is assigned to VirusHunter.**  **KeyLoggerDeployment is assigned to SecureWall.**  **SocialEngineered is assigned to VirusHunter.**  **CredentialHarvesting is already assigned to NetShield.**  **NonExistentDefense does not exist in the system.**  **ITCompromise is already assigned to VirusHunter.**  **SpywareInfiltration could not be mitigated by NetShield.**  **NonExistentAttack does not exist in the system.**  **ITCompromise is mitigated successfully.**  **ITCompromise is already mitigated.**  **DataStealer does not exist in the system.**  **Firewall cannot mitigate PhishingAttack.**  **RansomLock could not be mitigated by NetShield.**  **KeyLoggerDeployment is mitigated successfully.**  **Security:**  **Defensive Software: BasicBlocker, Effectiveness: 1, Assigned Attacks: [None]**  **Defensive Software: NetShield, Effectiveness: 8, Assigned Attacks: RansomLock, CredentialHarvesting, SpywareInfiltration**  **Defensive Software: SecureWall, Effectiveness: 7, Assigned Attacks: KeyLoggerDeployment**  **Defensive Software: UltimateDefender, Effectiveness: 10, Assigned Attacks: [None]**  **Defensive Software: VirusHunter, Effectiveness: 9, Assigned Attacks: ITCompromise, SocialEngineered**  **Threads:**  **-Mitigated:**  **Attack: ITCompromise, Severity: 7 (Target Mail: sysadmin@techcompany.com)**  **Attack: KeyLoggerDeployment, Severity: 6 (Infected Files: 120)**  **-Pending:**  **Attack: CredentialHarvesting, Severity: 6 (Target Mail: ceo@bigcompany.com)**  **Attack: RansomLock, Severity: 9 (Infected Files: 250)**  **Attack: SocialEngineered, Severity: 5 (Target Mail: hr@companyxyz.com)**  **Attack: SpywareInfiltration, Severity: 10 (Infected Files: 500)** |

## Task 3: Unit Tests (100 points)

You will receive a **SecureOpsSystem** skeleton with **SecurityTool** and **SecureHub** classes inside. **SecureOpsSystem** class will have some methods, fields, and constructors, that are working properly You are **NOT ALLOWED** to change any class. **Cover the whole class** (only **SecureHub**) with unit tests to make sure that the class is working as intended. You are provided with a **unit test project** in the **project skeleton**.

* **Do NOT CHANGE OR REMOVE ANY namespaces or usings.**
* **Do not use** "\r\n" **for a new line.**
* Do **NOT** use **Mocking** in your unit tests!
* In Judge, you upload a **.zip** of the **unit test project** from the **skeleton.**

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated