Object-Oriented Programming – Practical Exam

Problem 1 – War Machines

In a war machine virtual factory there are two types of **machines**: **tanks** and **fighters**. Each machine has **name**, **pilot**, **health points**, **attack points**, **defense points** and **attack targets**. Each pilot has **name** and **machines** he **engages**. Pilots make status **reports** on all machines they engage. One machine can be **engaged** by one pilot at a time. **Tanks** have **defense mode** which can be turned **on** and **off**. **Fighters** have stealth mode which can be turned **on** and **off**.

Design the Class Hierarchy

Your **task** is to **design an object-oriented class hierarchy** to model the war machines factory, machines and pilots **using the best practices for object-oriented design (OOD) and object-oriented programming (OOP)**. Avoid duplicated code though abstraction, inheritance, and polymorphism and encapsulate correctly all fields.

You are given few C# **interfaces** that you should **obligatory** implement and use as a basis of your code:

```
namespace WarMachines.Interfaces
{
  public interface IPilot
  {
        string Name { get; set; }
        void AddMachine(IMachine machine);
        string Report();
  }

  public interface IMachine
  {
        string Name { get; set; }
        IPilot Pilot { get; set; }
        double HealthPoints { get; set; }
        double AttackPoints { get; }
        double DefensePoints { get; }
        IList<string> Targets { get; }
        void Attack(string target);
        string ToString();
  }

  public interface ITank : IMachine
  {
        bool DefenseMode { get; }
        void ToggleDefenseMode();
  }

  public interface IFighter : IMachine
  {
        bool StealthMode { get; }
        void ToggleStealthMode();
  }
```

```
  public interface IMachineFactory
  {
        IPilot HirePilot(string name);
        ITank ManufactureTank(string name, double attackPoints, double defensePoints);
        IFighter ManufactureFighter(string name, double attackPoints,
               double defensePoints, bool stealthMode);
  }
}
```

All your machines should implement **IMachine**. Pilots should implement **IPilot**. Tanks and fighter courses should implement **ITank** and **IFighter** respectively. Machines and pilots should be created only through the **IMachineFactory** interface implemented by a class named **MachineFactory**. Tank's initial health points are always 100 and fighter's initial health points are always 200. Tank's defense mode adds 30 defense points to the initial ones and removes 40 attack points from the initial ones. By default tanks' defense mode is turned on. Fighters in stealth mode can be attacked only by other fighters.

The **IPilot.Report()** method returns the information about a pilot in the following form:

```
(pilot name) – (number of machines/"no") ("machine"/"machines")
- (machine name)
 *Type: ("Tank"/"Fighter")
 *Health: (machine health points)
 *Attack: (machine attack points)
 *Defense: (machine defense points)
 *Targets: (machine target names/"None" – comma separated)
 *Defense: ("ON"/"OFF" – when applicable)
- (machine name)
 *Type: (machine type)
 *Health: (machine health points)
 *Attack: (machine attack points)
 *Defense: (machine defense points)
 *Targets: (machine target names/"None" – comma separated)
 *Stealth: ("ON"/"OFF" – when applicable)
```

Look into the example below to get better understanding of the printing format.

The listed machines added to a certain pilot (though the **AddMachine(…)** method) should be ordered by health points then by name. If the pilot has no machines added, print **"no machines"**. If the pilot has **1** machine, print **"1 machine"** and list it. It is allowed to add the same machine more than once. Machine's targets are separated by "**,** " (comma + space). If a machine does not have targets, print **"None"**. All double type fields should be printed "**as is**", without any formatting or rounding.

The **IMachine.ToString()** method returns the information about a course in the following form:

```
- (machine name)
 *Type: ("Tank"/"Fighter")
 *Health: (machine health points)
 *Attack: (machine attack points)
 *Defense: (machine defense points)
 *Targets: (machine target names/"None" – comma separated)
 *Defense: ("ON"/"OFF" – when applicable)
 *Stealth: ("ON"/"OFF" – when applicable)
```

The **Type** is either "**Tank**" or "**Fighter**". The defense and stealth are only shown when applicable and should print "**ON**" or "**OFF**" depending on their state. The attack targets added to a certain machine (though the

`Attack(…)` method) should appear in the order of their addition. It is allowed to attack one machine (thus adding it) more than once. If the machine has no targets added to it, print "**None**". Do not print **`,`** (`comma`) at the line end. All double type fields should be printed "**as is**", without any formatting or rounding.

All properties in the above interfaces are mandatory (cannot be **`null`** or empty). Keep in mind that a machine can be manufactured without a pilot but **cannot** be engaged with **null** pilot afterwards.

If a **`null`** value is passed to some mandatory property, your program should throw **`a proper exception`**.

Additional Notes

To simplify your work you are given a virtual war machines factory execution engine that executes a sequence of commands read from the console using the classes and interfaces in your project (see the file **`WarMachinesSkeleton.rar`**). Please put your classes in namespace **`WarMachines.Machines.`** Implement the **MachineFactory** class in the namespace **`WarMachines.Engine.`**

You are only **allowed to write classes**. You are **not allowed to modify the existing interfaces and classes except the MachineFactory class**.

Current implemented commands the engine supports are:

- **HirePilot (name)** – adds a pilot with given name. Duplicate names are not allowed. As a result the command returns "**Pilot (name) hired**".
- **Report (name)** – searches for a hired pilot with given name and returns the **IPilot.Report()** method result.
- **ManufactureTank (name) (attack) (defense)** – creates a tank with given name, attack and defense points. Duplicate names are not allowed. Initial health points are always 100. Initial defense mode is turned on. As a result the command returns "**Tank (name) manufactured - attack: (attack); defense: (defense)**".
- **DefenseMode (name)** – searches for tank with given name and toggles its defense mode. As a result the command returns "**Tank (name) toggled defense mode**".
- **ManufactureFighter (name) (attack) (defense) (stealth)** – creates a fighter with given name, attack and defense points. Duplicate names are not allowed. Initial health points are always 200. As a result the command returns **"Fighter (name) manufactured - attack: (attack); defense: (defense); stealth: (ON/OFF)"**
- **StealthMode (name)** - searches for fighter with given name and toggles its stealth mode. As a result the command returns "**Fighter (name) toggled stealth mode**".
- **Engage (pilot-name) (machine-name)** – searches for a pilot and machine by given names, adds the machine to the pilot's list of machines and initialises the machine's pilot. As a result the command returns "**Pilot (pilot-name) engaged machine (machine-name)**".
- **Attack (attacking-machine-name) (defending-machine-name)** – searches for two machines by given names and the first one attacks the second one if it is possible. As a result the command returns "**Machine (defending-machine-name) was attacked by machine (attacking-machine-name) - current health: (defending-machine-health).**

In case of invalid operation or error, the engine returns appropriate text messages.

Sample Input

```
HirePilot John
HirePilot Nelson
Report Bender
ManufactureTank T-72 100 100
ManufactureFighter Kingcobra 150 90 StealthON
Report John
Engage John T-72
Engage John Kingcobra
Report John
Report Nelson
Engage Nelson T-72
Engage Nelson Kingcobra
ManufactureFighter Boeing 180 90 StealthOFF
Engage Nelson Boeing
Attack T-72 Kingcobra
Attack T-72 Boeing
DefenseMode T-72
DefenseMode Kingcobra
DefenseMode Boeing
Attack T-72 Kingcobra
Attack T-72 Boeing
StealthMode Kingcobra
StealthMode Boeing
StealthMode T-72
Attack Kingcobra T-72
Attack Boeing T-72
Report Nelson
Report John
```

Sample Output

```
Pilot John hired
Pilot Nelson hired
Pilot Bender could not be found
Tank T-72 manufactured - attack: 100; defense: 100
Fighter Kingcobra manufactured - attack: 150; defense: 90; stealth: ON
John - no machines
Pilot John engaged machine T-72
Pilot John engaged machine Kingcobra
John - 2 machines
- T-72
 *Type: Tank
 *Health: 100
 *Attack: 60
 *Defense: 130
 *Targets: None
 *Defense: ON
- Kingcobra
 *Type: Fighter
 *Health: 200
 *Attack: 150
```

```
 *Defense: 90
 *Targets: None
 *Stealth: ON
Nelson - no machines
Machine T-72 is already occupied
Machine Kingcobra is already occupied
Fighter Boeing manufactured - attack: 180; defense: 90; stealth: OFF
Pilot Nelson engaged machine Boeing
Tank T-72 cannot attack stealth fighter Kingcobra
Machine Boeing was attacked by machine T-72 - current health: 200
Tank T-72 toggled defense mode
Machine Kingcobra does not support this operation
Machine Boeing does not support this operation
Tank T-72 cannot attack stealth fighter Kingcobra
Machine Boeing was attacked by machine T-72 - current health: 190
Fighter Kingcobra toggled stealth mode
Fighter Boeing toggled stealth mode
Machine T-72 does not support this operation
Machine T-72 was attacked by machine Kingcobra - current health: 50
Machine T-72 was attacked by machine Boeing - current health: 0
Nelson - 1 machine
- Boeing
 *Type: Fighter
 *Health: 190
 *Attack: 180
 *Defense: 90
 *Targets: T-72
 *Stealth: ON
John - 2 machines
- T-72
 *Type: Tank
 *Health: 0
 *Attack: 100
 *Defense: 100
 *Targets: Boeing, Boeing
 *Defense: OFF
- Kingcobra
 *Type: Fighter
 *Health: 200
 *Attack: 150
 *Defense: 90
 *Targets: T-72
 *Stealth: OFF
```