<u>للّذين يحلُّون بلغة C#</u>

في حديقة الحيوانات "نعيش بسلام"، طُوِّرَت منظومة مُحَوْسَبة يحفظون فيها معطيات عن الحيوانات التي في الحديقة.

تحوي المنظومة الفئات التالية: حيوان – **Animal**، عصفور – **Bird**، ببّغاء – **Parrot**، سمكة – **Fish**، أفعى – **Snake**.

معطى قسم من مبنى الفئات في المنظومة المُحَوْسَبة:

```
public class Animal {
    private string color;
    private int weight;
    public Animal (string color, int weight)
    {
        this.color = color;
        this.weight = weight;
    }
    public override string ToString() {
        return "My color is " + this.color + "! And I weigh " + weight + " kilos!";
    }
}
public class Bird {
    private string color;
    private int weight;
    private bool isFlying;
    public void Zoo () {
        Console.WriteLine ("Hello");
    }
}
public class Parrot {
    private string color;
    private int weight;
    private bool isFlying;
    private bool isTalking;
    public void Zoo () {
        Console.WriteLine ("Hello");
    }
}
public class Fish {
    private string color;
    private int weight;
    private string waterType;     // سمكة بحر / سمكة مياه عذبة
}
public class Snake {
    private string color;
    private int weight;
    private int length;
    private bool isVenomous;     // هل الأفعى سامّة؟
}
```

افترِضوا أنّه توجد عمليّات Get وَ Set لصفات الفئة.

أ. اكتبوا من جديد عناوين وصفات وعمليّات الفئات: **Bird** وَ **Parrot** وَ **Fish** وَ **Snake** ، حسب مبادئ البرمجة الموجّهة كائنات (لا تُغيِّروا الفئة Animal) . في هذا البند، لا حاجة لكتابة عمليّات بنائيّة . افترِضوا أنّ العمليّتَيْن Get وَ Set موجودتان . إضافةً إلى ذلك، ارسموا مخطَّطًا هرميًّا بين جميع الفئات . يجب الإشارة إلى توريث بواسطة السهم ⟵ .

ب. معطاة الفئة Tester :

```
public class Tester {
    public static void Main (string[] args) {
        Animal[] animals = new Animal [5];
        animals [0] = new Bird ("white", 4, false);
        animals [1] = new Fish ("blue", 3, "sweet water");
        animals [2] = new Parrot ("brown", 12, true, true);
        animals [3] = new Snake ("gray", 2, 6, true);
        animals [4] = new Snake ("black", 3, 4, false);
        //***
    }
}
```

أضيفوا عمليّات بنائيّة إلى الفئات **Bird** وَ **Parrot** وَ **Fish** وَ **Snake** ، تبتدئ صفات الفئات (بحيث تعمل العمليّة Main بدون أخطاء) .

ملاحظة: لا تُغيِّروا العمليّة البنائيّة المعطاة لِـ **Animal** .

جـ. **(1)** في العمليّة Main ، أُضيفَت قطعة الكود التي أمامكم (في المكان المشار إليه بِـ ***) :

```
for (int i = 0; i < animals.Length; i++) {
    Console.WriteLine (animals[i]); }
```

قطعة الكود سليمة، وتطبع المُخرَج الذي أمامكم:

I'm a bird! My color is white! And I weigh 4 kilos!
My color is blue! And I weigh 3 kilos!
I'm a parrot! My color is brown!
I'm a snake! I'm venomous, be careful! My color is gray! And I weigh 2 kilos!
I'm a snake! I'm not venomous! My color is black! And I weigh 3 kilos!

أضيفوا العمليّة ToString إلى الفئات (فقط أين تدعو الحاجة) حسب مبادئ البرمجة الموجّهة كائنات، بحيث يكون المُخرَج ملائمًا .

ملاحظة: لا تُغيِّروا العمليّة ToString في الفئة **Animal** . الإجابة التي تتضمّن تغييرًا لن تحصل على درجات .

(2) في الفئة Tester ، أُضيفَت العمليّة التي أمامكم:

```
public static void Hello (Animal [] arr) {
    for (int i = 0; i < arr.Length; i++) {
        arr[i].Zoo(); }
}
```

إذا كانت العمليّة سليمة، اكتبوا مُخرَجها بالنسبة للمصفوفة animals (المعطاة أعلاه)، وإذا لم تكن العمليّة سليمة – صحِّحوها (بحيث يكون استدعاء Zoo داخل الحلقة سليمًا)، واكتبوا المُخرَج بالنسبة للمصفوفة animals .

ملاحظتان: – لا تُغيِّروا الفئات نفسها . الإجابة التي تتضمّن تغييرًا في الفئات لن تحصل على درجات .

– يجب أن تعمل العمليّة بدون أخطاء بالنسبة لكلّ مصفوفة من نمط Animal .