```
1  namespace Aufgabe_1;
2
3  class Program_1
4  {
5      static void Main()
6      {
7      }
8  }
9
10 // 1.1    4 Punkte
11 // Legen Sie eine neue Klasse "CAnimal" an,
12 // und geben Sie ihr die les- und schreibbaren Properties:
13 // - "NoOfLegs" (Anzahl Beine) vom Typ int und
14 // - "HasWings" (hat Flügel) vom Typ bool.
15
16
17 // 1.2    3 Punkte
18 // Leiten Sie von der Klasse "CAnimal" die Klasse "CBird" ab und geben
19 // Sie dieser einen Konstruktor, der die Anzahl der Beine fest auf 2 setzt
20
21
22 // 1.3    5 Punkte
23 // Geben Sie der Basisklasse "CAnimal" jetzt einen Konstruktor mit einem Parameter,
24 // mit dem man beim Erzeugen eines beliebigen Tieres setzen kann, ob es Flügel hat
25 // Die Klasse "CBird" erzeugt jetzt einen Fehler beim Übersetzen. Fügen Sie deshalb
26 // der Klasse "CAnimal" den jetzt fehlenden Konstruktor auch noch hinzu.
27
28
29 // 1.4    2 Punkte
30 // Erweitern Sie den Konstruktur der Klasse CBirds so, dass alle erzeugten Objekte
31 // durch Aufruf des entsprechenden Konstruktors der Basisklasse Flügel bekommen
32
33
34 // 1.5    6 Punkte
35 // Überlegen Sie, in welcher Klasse Sie die Methoden "Hatch()" (das Schlüpfen aus dem Ei)
36 // und "Move()" (Bewegen) ansiedeln würden. Codieren Sie beide Methoden.
37 // Kommentieren Sie bei den Methoden den Grund Ihrer Entscheidung.
38
39
40 // 1.6    2 Punkte
41 // Erzeugen Sie jetzt in Main() einen Vogel, und bewegen Sie ihn.
```

```
1  namespace Aufgabe_2;
2
3  class Program_2
4  {
5      static void Main()
6      {
7      }
8  }
9
10 // 2.1  6 Punkte
11 // Erzeugen Sie die Klasse "AnyStudent", und sorgen Sie dafür, dass man kein Objekt dieser
12 // Klasse erzeugen kann.
13 // Leiten Sie die instantiierbaren Klassen "DHBWStudent" und "FHStudent" von dieser Klasse
14 // ab.
15 // Die Klasse "AnyStudent" soll dafür sorgen, dass alle abgeleiteten Klassen die Methode
16 // "int GetHolidays()" implementieren, die die Anzahl der Ferien-/ Urlaubstage zurückgibt.
17 // Fügen Sie diese Methode(n) entsprechend hinzu.
18
19
20 // 2.2   7 Punkte
21 // Leiten Sie die neuen Klassen DHBWIStudent und FHIStudent von einem adäquat benannten
22 // Interface  ab, das die Aufgabe der abstrakten Basisklasse übernimmt.
```

```
1  namespace Aufgabe_3_Generics;
2
3  class Program_3
4  {
5      static void Main()
6      {
7      }
8  }
9
10 // 3.1     6 Punkte
11 // Stellen Sie die folgende Klasse und ihr Feld auf generische (Valuetype-) Typen um,
12 // um die Property "Value" dieses Typs typunabhängig mit der Methode Print() auszugeben.
13
14 public class CMath
15 {
16     public int Value { get; set; }
17
18     public void Print()
19     {
20         Console.WriteLine($"The value is: {Value}");
21     }
22 }
23
24 // 3.2.   4  Punkte
25 // Erzeugen Sie in "Main()" ein Objekt der unter 3.1 geänderten Klasse "CMath"
26 // vom Typ "double", geben Sie ihm den Wert 5.7 und geben Sie seinen Wert aus.
```

```
1  namespace Aufgabe_4_Stack;
2
3  class Program_4
4  {
5      static void Main()
6      {
7      }
8  }
9
10 // 4.1  6  Punkte
11 // Verwenden Sie die Klasse Stack<T> aus dem Namensraum "System.Collections.Generic",
12 // um aus einem String einen neuen String aufzubauen, der die Buchstaben in umgekehrter
13 // Reihenfolge enthält (ergänzen/ korrieren Sie die fehlenden Zeilen)
14
15 public class Strings
16 {
17     public string ReverseString(string input)
18     {
19         string newString=new string("");
20
21         return newString;
22     }
23 }
24
25 // 4.2  6  Punkte
26 // Nutzen Sie die Funktion "ReverseString()" in Main(), um die Buchstaben des Strings
27 // "Otto liebt Lotto" umzudrehen und geben Sie den neuen String auf die Konsole aus
```