Wie bereits aus der Presse bekannt wurde, werden Kennzeichen von Fahrzeugen auf Bundesautobahnen automatisch erfasst. Hier sind die Datenbanken durcheinandergeraten und die Datensätze müssen wieder in einem Array `alleKennzeichen` zusammengesetzt werden. Um die Daten später übersichtlich sichten zu können, müssen die Datensätze zudem nach unterschiedlichen Kriterien sortierbar sein. Entsprechend ist die folgende Aufgabe umzusetzen:

Aufgabe 1

Ein Kennzeichen setzt sich drei Kennzeichenteilen (abgeleitete Klassen) zusammen: Während `KennzeichenOrt` und `KennzeichenBuchstabe` jeweils einen `String` speichern werden, speichert `KennzeichenZahl` einen `int` im Property `Data`.

```
abstract class KennzeichenTeile<T> : IComparer {
       public abstract T Data { get; set; }
       …
}
```

Die Schnittstelle `IComparer` ist entsprechend zu implementieren[1], damit diese im folgenden Code (darf nicht verändert oder ergänzt werden) funktioniert. Dieser Code kann in der Main() implementiert werden.

```
KennzeichenOrt[] dbOrte = { new KennzeichenOrt("S"), new KennzeichenOrt("N"), new
KennzeichenOrt("HAC") };
KennzeichenBuchstabe[] dbBuchstaben = { new KennzeichenBuchstabe("IN"), new
KennzeichenBuchstabe("IT"), new KennzeichenBuchstabe("K") };
KennzeichenZahl[] dbZahlen = { new KennzeichenZahl(1337), new KennzeichenZahl(2019), new
KennzeichenZahl(512) };

Kennzeichen[] alleKennzeichen = new Kennzeichen[3];
for (int i = 0; i < alleKennzeichen.Length; i++) {
    alleKennzeichen[i] = new Kennzeichen(dbOrte[i], dbBuchstaben[i], dbZahlen[i]);
}

Array.Sort(alleKennzeichen, (IComparer)new KennzeichenOrt(" "));
for (int i = 0; i < alleKennzeichen.Length; i++) Console.WriteLine(alleKennzeichen[i]);
Console.WriteLine();
Array.Sort(alleKennzeichen, (IComparer)new KennzeichenBuchstabe(" "));
for (int i = 0; i < alleKennzeichen.Length; i++) Console.WriteLine(alleKennzeichen[i]);
Console.WriteLine();
Array.Sort(alleKennzeichen, (IComparer)new KennzeichenZahl(1));
for (int i = 0; i < alleKennzeichen.Length; i++) Console.WriteLine(alleKennzeichen[i]);
```

Folgende Ausgabe soll dabei entstehen:

```
HAC K 512
N IT 2019
S IN 1337

S IN 1337
N IT 2019
HAC K 512

HAC K 512
S IN 1337
N IT 2019
```

[1] https://docs.microsoft.com/de-de/dotnet/api/system.collections.icomparer?view=netframework-4.8

Aufgabe 2

Erweitern Sie die drei Klassen zu den Kennzeichenteilen mit entsprechenden Exceptions, falls Data auf falsche Werte gesetzt werden soll. Diese erben von

```
abstract class InvalidKennzeichenException : Exception {
       String kennzeichenNachricht = "Kennzeichen ungültig: ";
       …
}
```

Beim Abfangen der Fehlernachrichten, wie hier am Beispiel einer falschen KennzeichenZahl

```
try {
    KennzeichenZahl invalid = new KennzeichenZahl(12345);
}
catch (InvalidKennzeichenException e) {
    Console.WriteLine(e.KennzeichenNachricht);
}
```

werden entsprechende Fehlernachrichten generiert:

```
Kennzeichen ungültig: Bestandteil Zahl darf nur eine Zahl zwischen [1,9999] enthalten. War aber: 12345

Kennzeichen ungültig: Bestandteil Buchstabe darf nur 2 Buchstaben enthalten. War aber: ABC

Kennzeichen ungültig: Bestandteil Ort darf nur 3 Buchstaben enthalten. War aber: ABCD
```

Allgemeine Hinweise für beide Aufgaben:

- Der Namespace System.Collections darf ausschließlich für IComparer verwendet werden.
- Es dürfen keine Delegates/Events verwendet werden.